Stony Brook University

OFFICIAL COPY

The official electronic file of this thesis or dissertation is maintained by the University Libraries on behalf of The Graduate School at Stony Brook University.

© All Rights Reserved by Author.

Reasoning about Uncertainty and Correlated Beliefs

A DISSERTATION PRESENTED

BY

Hui Wan

TO

THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

COMPUTER SCIENCE

STONY BROOK UNIVERSITY

August 2010

Copyright by
Hui Wan
2010

Stony Brook University

The Graduate School

Hui Wan

We, the dissertation committee for the above candidate for the degree of Doctor of Philosophy, hereby recommend acceptance of this dissertation.

Michael Kifer - Dissertation Advisor
Professor, Computer Science Department

C. R. Ramakrishnan - Chairperson of Defense
Professor, Computer Science Department

I. V. Ramakrishnan
Professor, Computer Science Department

Yanhong Annie Liu
Professor, Computer Science Department

V. S. Subrahmanian
Professor, Computer Science Department
University of Maryland

This dissertation is accepted by the Graduate School.

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Reasoning about Uncertainty and Correlated Beliefs

by

Hui Wan

Doctor of Philosophy

in

Computer Science

Stony Brook University

2010

Reasoning about uncertainty has been an active area of research for over thirty years. With the advent of networked systems, especially the Web, it became important to be able to combine uncertain information that comes from different sources. One of the hard problems in this area is combining evidence obtained from correlated, possibly conflicting, information sources.

This dissertation develops Belief Logic Programming (BLP), a novel form of quantitative logic programming that deals with uncertain and inconsistent information and is able to combine and correlate evidence obtained from non-independent information sources. BLP was inspired by Dempster-Shafer theory of evidence and belief combination functions. Our approach does not depend on a particular method of combining evidence and, in fact, different combination methods can be used simultaneously for different types of uncertain information. Most importantly, unlike previous efforts to integrate uncertainty and logic programming, BLP can correlate structural information contained in rules and provides more accurate estimates for certainty factors.

Together with declarative and fixpoint semantics for BLP, the dissertation develops optimized query evaluation algorithms. In addition, the monotonicity and non-monotonicity properties of BLP as well as the relationship to defeasible reasoning, paraconsistent reasoning, and Dempster-Shafer theory of evidence are discussed.

After developing the basic framework, the dissertation develops several extensions. One extension allows cyclic dependencies in BLP rules. Another extension captures quantitative correlation among the base facts. Since full correlation information might not always be available, we develop an approach that allows to approximate correlation information using only partial information. Under certain conditions, this approximate method yields the same results as the method that is based on full information. The query evaluation algorithms are then extended to cover the enhancements.

Dedicated to my dear husband, Haifeng.

Contents

List of Figures

Acknowledgments

Thanks above all to my advisor, Professor Michael Kifer, for the invaluable guidance and patient help he has given to me throughout my Ph.D. study. He spent countless time and effort to teach me how to be meticulous in research and how to have clarity in thought and presentation, and his broad knowledge and keen perception of research directions have greatly benefitted my research towards this dissertation. He set a wonderful example for me as a researcher and as a person. It has truly been a pleasure and an honor working with him.

Thanks to members of my dissertation committee: Professor V. S. Subrahmanian, Professor I. V. Ramakrishnan, Professor Yanhong Annie Liu, and Professor C. R. Ramakrishnan for reviewing my dissertation and giving valuable feedbacks.

Thanks to Dr. Benjamin Grosof for the helpful discussions and encouragements, and for his collaborative work with me.

Thanks to my fellow graduate students: Chang Zhao, Paul Fodor, Senlin Liang, Anu Singh, Jalal Mahmud, Ping Yang, Diptikalyan Saha, Zan Sun, Saikat Mukherjee, Pei Ye, and many others, for their help and friendship.

I would like to mention that this work was supported, in part, by the Halo project sponsored by Vulcan, Inc. and by the NSF grant CCF-0530363.

Chapter 1

Introduction

Knowledge representation and reasoning is an important part of artificial intelligence. Automated reasoning systems have been employed in many areas, such as science, business, health care, computer games, etc., to imitate the reasoning process of human beings.

Classical reasoning systems assume information to be certain and consistent, but in the real world such assumptions seldom hold. For example, when a medical expert system tries to diagnose a patient based on certain symptoms, evidence is often insufficient to make a definite diagnosis, and the outcome typically includes several possible diagnoses, each with its likelihood. For another example, when a stock trading system tries to make a prediction of the next-day market trend, the available evidence (e.g. past market data and recent news) is often not only insufficient but also contradictory. Again, several outcomes are possible and each has its likelihood.

In order to address such problems, numerous studies have been done to extend classical logic to reasoning with uncertain information. In this dissertation, these works will be referred to as *quantitative logic*. For a given statement A, a classical logic may allow to draw a conclusion that A is true or

false, while a quantitative logic may warrant a conclusion that, based on the available evidence, A is true to a certain extent, that it is false to a certain extent, and sometimes even that A's truth is unknown to a certain extent. In other words, in addition to the discrete truth values of "true" or "false", numerical values can be calculated for statements in a quantitative logic framework, to describe the degrees to which they are true, false, or unknown.

Despite the huge amount of works on quantitative logic, less explored is the issue of combining correlated pieces of uncertain information. Most works disregard correlation or assume that all the information sources are independent. Others make an effort to take some forms of correlation into account, but only in an ad hoc manner.

This dissertation develops Belief Logic Programming (BLP), a novel form of quantitative logic programming that deals with uncertain and inconsistent information and is able to combine and correlate evidence obtained from non-independent information sources. Although inspired by Dempster-Shafer theory of evidence and belief combination functions, BLP does not depend on a particular method of combining evidence and, in fact, different combination methods can be used simultaneously for different types of uncertain information. Most importantly, unlike previous efforts to integrate uncertainty and logic programming, BLP can correlate structural information contained in rules and provides more accurate estimates for certainty factors.

The remainder of this dissertation is organized as follows. Chapter 2 gives a survey of prior works on quantitative logic programming and on reasoning with uncertainty, and introduces fundamentals of logic programming and Dempster-Shafer theory. Chapter 3 motivates and defines the basic framework of BLP, including its syntax, declarative semantics and fixpoint semantics. Several

properties of BLP are discussed, including its relationship to some other theories of non-monotonic reasoning. The results in Chapter 3 were published in [85]. Chapter 4 provides algorithms for query evaluation in belief logic programs (blps). The results in Chapter 4 were published in [86]. Chapter 5 and Chapter 6 develop extensions of BLP to cyclic blps and correlated facts, respectively. For clarity of presentation, we present these extensions separately, and provide their integrations into the query evaluation algorithms, as well as a discussion on the overall extended framework, in Section 5.6, Section 6.2, and Section 6.3. The results in Chapter 5 were published in [83]. Following Chapter 6, Chapter 7 provides a method to approximate belief logic programs with correlated facts. Finally, Chapter 8 concludes the dissertation.

Chapter 2

Background: Uncertainty and Logic Programming

As the name suggests, Belief Logic Programming (BLP) builds on the formalism of logic programming. In this chapter, we start by reviewing fundamentals of logic programming, which is followed by a brief history of quantitative logic programming and a survey of existing works on rule-based reasoning with uncertainty. Finally we review several key concepts from Dempster-Shafer theory, to lay the foundation for future chapters.

2.1 Logic Programming Fundamentals

The first programs based on logic were developed by Colmerauer, et al. at the University of Marseilles in 1972. Then van Emden and Kowalski [81] laid down the theoretical foundation of logic programming. Logic programming is now a mature field in artificial intelligence and has numerous applications. One of the most popular logic programming language is PROLOG.

A logic program is built from constants, variables, function symbols, predicate symbols, and connectives: $\wedge$ for conjunction, `:-` for implication, and $\neg$ for negation. A **term** is either a constant, a variable, or of the form $f(t_1, ..., t_n)$ where f is a function symbol and t_i, $1 \leq i \leq n$, are terms. An **atom** is of the form $p(t_1, ..., t_n)$ where p is a predicate symbol and t_i, $1 \leq i \leq n$, are terms. A **clause** is either a fact which is an atom, or is a rule of the form

$$a \ \texttt{:-} \ b_1 \wedge ... \wedge b_n$$

where a is an atom and b_i, $1 \leq i \leq n$, is an atom or negation of an atom. A program is a set of clauses.

A **ground** atom is an atom that does not contain any variable. A **ground instance** of a clause is obtained by universally replacing the variables in the clause by ground terms. The set of all ground atoms that can be constructed from the symbols is called the **Herbrand base**.

Logic programs have model-theoretic semantics as follows. A **Herbrand interpretation** I is a function that maps every atom in the Herbrand base to truth value **t** (true) or **f** (false). Logical operations are used to compute the truth value of a formula under a Herbrand interpretation. A Herbrand interpretation I **satisfies** a ground rule unless the body of the rule is true but the head of the rule is false under I. A fact can be looked on as a rule whose body being a special proposition `true` which is always true. A Herbrand interpretation I is a **Herbrand model** of a program if I satisfies every ground instance of every clause in the program. Let I_1 and I_2 be a pair of Herbrand interpretations, $I_1 \preceq I_2$ iff there does not exist an atom A such that $I_1(A) = \mathbf{t}, I_1(A) = \mathbf{f}$.

Every logic program that does not contain negation has a unique least Herbrand model with respect to $\preceq$. But for logic programs with negation

("negation as failure") in rule body, there is often no such unique least Herbrand model. To provide semantics for such programs, Gelfond and Lifschitz proposed stable model semantics [20]; van Gelder, et al. and Przymusinski proposed well-founded semantics [19, 59].

Extensions of classical Logic programming include quantitative logic programming [72, 2, 17, 48, 31, 39, 65], paraconsistent logic programming [6, 60, 18, 51, 26], defeasible reasoning [64, 50, 23, 84], etc.

2.2 Rule-based Reasoning with Uncertainty

2.2.1 A Brief History of Quantitative Logic Programming

The first well-known quantitative logic programming framework was proposed by Shapiro [72], which is a general framework where the certainty factor of every atom is a real number between 0 and 1, and every clause is associated with a function to compute the certainty factor of head atom from the certainty factors of body atoms. Following that, several works [1, 2, 17, 28] adopted some special cases of Shapiro's syntax where the certainty function takes special forms. Among them, Baldwin's support logic programming [1, 2] made further departure from [72] in that it used a subinterval of [0,1] instead of a single number as certainty factor of each atom. It is also worth noting that Baldwin's work [1, 2] contains multiple ways of computing the certainty factor of conclusion from premises, one of which is based on the Dempster-Shafer theory [10, 71] and another based on fuzzy logic [89].

Subrahmanian and Blair [4, 5, 78] laid down the framework of annotated logic programming where each atom in a clause is annotated with a truth

value. For the body of a rule to hold, the certainty factor of each body atom must satisfy a condition specified by the annotated truth value. Kifer and Subrahmanian [31] generalized this framework to a Generalized Annotated Program which supports more complex annotations and greatly extends the expressive power.

In the above works, although certainty factors are often constrained to be between 0 and 1, they are not interpreted as probabilities. Many follow-up works gave probabilistic meanings to certainty factors [9, 35, 38, 39, 47, 48, 49]. In this dissertation, such quantitative logic frameworks in which certainty factors have probabilistic significance will be referred to as *probabilistic logic*; they are discussed in details in Section 2.2.3. One popular approach is to use probability intervals $[v, w]$ as certainty factors, where v and w are interpreted as lower and upper probabilities [9, 38, 39, 47, 48, 49]. The probability of each statement A, is constrained by $v \leq P[A] \leq w$. Specifically, Ng and Subrahmanian [47, 48, 49] were the first to give probabilistic meaning to certainty factors. They introduced probability intervals into annotated logic programming [4, 5, 31, 78] and used these intervals as annotations. Dekhtyar and Subrahmanian presented another annotation-based probabilistic program [9], which allows choices of conjunction and disjunction operators based on known dependencies between events, and in which probability intervals from multiple rules are combined with operator $\cap$. Lukasiewicz [38, 39] used intervals to constrain conditional probability in an implication-based framework. He also presented an approximation technique which uses van Emdens quantitative deduction [17] to compute a candidate model of a probabilistic logic program. There are other approaches to integrating logic programs with probabilistic graphical models [68, 29, 65].

Lakshmanan and Shiri proposed a generic implication-based framework [35]

which is not based on any particular uncertainty formalism. Let T be a certainty lattice and $B(T)$ be the set of finite multisets over T, FF is defined as a family of functions $f : B(T) \rightarrow T$ where f satisfies certain properties such as monotonicity, continuity, commutativity, associativity, etc. Functions are chosen from FF to compute the certainty of conjunctions, the certainty of disjunctions, and the certainty of rule head given the certainty of rule body, respectively. When an atom is supported through multiple rules, the certainty propagated from these rules are combined with a combination function, which is also from FF. However, correlations between overlapping derivation paths are ignored.

Besides probability theory, fuzzy set theory and Dempster-Shafer theory are also popular formalisms which are used to model uncertainty in quantitative logic programming. Sections 2.2.2, 2.2.3, and 2.2.4 give more detailed reviews of quantitative logic works based on the uncertainty formalisms they use.

2.2.2 Frameworks Based on Fuzzy Set Theory

Fuzzy sets were introduced by Lofti A. Zadeh [89] in 1965. A *fuzzy set* α is defined as a pair (S, t) where S is a (classical) set of elements, and $t : S \rightarrow [0, 1]$ is called the *membership function* of α. For each $x \in S$, $t(x) = 0$ means that x is not a member of the fuzzy set α, $t(x) = 1$ means that x is fully a member of α, $0 \leq t(x) \leq 1$ means that x is a fuzzy member of α and $t(x)$ is the degree of membership of x in α.

Fuzzy logic is a form of quantitative logic that is based on fuzzy set theory. It has been a very active research area and resulted in numerous formal frameworks [28, 33, 37, 80, 94]. However, the basic ideas are relatively invariant. In

fuzzy logic, each logic statement has a degree of truth between 0 and 1, which is interpreted as a degree of membership in a fuzzy set [89]. Note that these degrees of truth values can not be interpreted as probabilities [92, 93, 94]. A formula is composed from simple statements and connectives $\wedge$ (which means "and", conjunction), $\vee$ (which means "or", disjunction), and $\neg$ (which means "not", negation). The degree of truth assigned to a formula is defined as follows.

Definition 2.1 *[80] Let A and B be arbitrary formulas. Then*

$$\begin{aligned} t(A \wedge B) &= min\{t(A), t(B)\} \\ t(A \vee B) &= max\{t(A), t(B)\} \\ t(\neg A) &= 1 - t(A) \\ t(A) &= t(B) \quad \textit{if } A \textit{ and } B \textit{ are logically equivalent} \end{aligned}$$

□

Various logic programming semantics based on fuzzy logic have been proposed. One of the four models for conjunction and disjunction computation proposed by Baldwin [1] is based on fuzzy logic. Other examples include Vojtás' Fuzzy Logic Programming [82].

Although fuzzy logic has been successfully applied in many applications, such as linguistics and control systems [76], it remains controversial due to a number of weaknesses [16, 25]. For example, let S be a fuzzy set and S' be the complement of S. Then according to the definitions of the complement and intersection in the fuzzy set theory [89], it is possible to have $S \cap S' \neq \varnothing$. In other words, it is possible that there exists some statement A such that the degree of truth of $A \wedge \neg A$ is greater than 0. Such properties are counter-intuitive and hard to justify in many real life applications of reasoning with uncertainty.

Zadeh introduced possibility theory [90] in 1978 as an extension of fuzzy set theory. Dubois and Prade further developed it and introduced possibilistic logic [12, 14]. In possibility theory, the possibility measure is a function $pos : Pow(\Omega) \rightarrow [0, 1]$ where Ω is the universal set and $Pow(\Omega)$ is the powerset of Ω, such that

$$pos(\varnothing) = 0, \quad pos(\Omega) = 1, \quad pos(U \cup V) = max\{pos(U), pos(V)\}$$

It follows that $pos(U) = max_{x \in U}\ pos(\{x\})$, and $\exists x_0 \in \Omega,\ pos(\{x_0\}) = 1$. This, although applicable in certain application domains, in many other applications, makes it inappropriate to use possibility theory as a formalism to model uncertainty.

2.2.3 Frameworks Based on Probability Theory

Another popular track of quantitative logic is based on probability theory. There have been numerous works [38, 39, 47, 48, 49, 67, 68, 69] in this track, some of which are based on probabilistic graphical models, while others based on probabilities in general.

Poole proposed the Independent Choice Logic (ICL) [56, 57, 58]. The ICL framework can be seen as adding independent stochastic inputs to a logic program. An *alternative* is defined to be a set of atoms. Atoms in an alternative are called *atomic choices*. A *choice space* is defined to be a set of alternatives. An ICL program consists of an acyclic logic program F, a choice space C, and a probability distribution P_0 which assigns probability to each atomic choice in alternatives in C, such that $\forall \chi \in C \sum_{\alpha \in \chi} P_0(\alpha) = 1$. Atomic choices in the same or different alternatives cannot unify with each other. Atomic choices cannot unify with the head of any clause in F. A *total choice* for C is a selection of exactly one atomic choice from each grounding of each alternative

in C. Each total choice decides a possible world. ICL imposes an independence assumption: different alternatives are independent, and different ground instances of alternatives are independent. Under this assumption, the probability of a possible world, i.e., a total choice, is the product of the probabilities of the atomic choices that make up this total choice. Finally, the probability of a formula is the sum of the probabilities of all the possible worlds in which this formula is true. It is claimed in [58] that ICL is expressive enough to "represent anything that is representable by a belief network".

Sato and Kameya proposed parameterized logic programs [67, 68, 69], which are implemented in a system called PRISM. A parameterized logic program is a definite clause program [1] $DB = F \cup R$ where F is a set of facts and R is a set of rules, such that no rule head in R is unifiable with any fact in F. Rules are non-probabilistic as in classical logic, but facts are probabilistic. There is a probability distribution P_F over Ω_F where Ω_F is defined to be the set of all interpretations of F. Each interpretation of $J \in \Omega_F$ is viewed as a possible world. The probability distribution can be extended from F to the whole Herbrand base as follows. Ω_{DB} is defined to be the set of all Herbrand interpretations of DB. The probability of a Herbrand interpretation I is the sum of the probabilities of all the possible worlds in which I is true, i.e., the sum of $P_F(J)$ of all the interpretations $J \in \Omega_F$ such that I can be entailed from J according to the rules in R. Let B be an atom, an *explanation* for B is a conjunction S such that 1) S is composed of facts in F, 2) $S \cup R$ entails B, and 3) removal of any one or more conjuncts from S violates 2). The probability of B is the sum of the probabilities of all the possible worlds where some explanation for B is true.

Suppose P_F contains a set of parameters Θ, Sato and Kameya developed

[1] A definite clause is a clause that contains no negation in the head.

EM algorithms in [68] to learn Θ from a set of observations on atoms in the Herbrand base. Some strong assumptions are imposed, such as 1) explanations must be mutually exclusive, and 2) facts are of the form $msw(N, I, v)$ and facts with different N and I must be independent of each other. These assumptions greatly limit the expressive power and flexibility of parameterized logic programs. However, if these assumptions are removed, the computational complexity would be very high.

Though independently developed, parameterized logic programs and ICL have much in common. In both, the rules are always non-probabilistic, the atoms in the probabilistic part of the program can not be the head of any rule, and therefore conditional probabilities can only be represented indirectly. One notable difference between parameterized logic programs and ICL is: In parameterized logic programs, the probability distribution over Ω_F can be used to represent correlation among facts in F, while in ICL, alternatives must be independent of each other.

Muggleton proposed stochastic logic programs [42], which are inspired by Hidden Markov Model [62]. *Stochastic clauses* in stochastic logic programs are of the form $p : C$ where $p \in [0, 1]$ and C is a range-restricted clause [2] of the standard form $A \leftarrow B_1, ..., B_n$. A set of stochastic clauses is called a *stochastic logic program* if and only if for each predicate symbol q the probabilities labeled on the clauses with q in the head sum to 1. This restriction, together with the restriction that every clause must be range-restricted, greatly limits the expressive power of stochastic logic programs. Each SLD-resolution for a query is assigned a probability value which is the product of the probabilities of the clauses chosen in this resolution. With the assumption that derivations

[2] A clause C is said to be range-restricted if every variable in the head is found in the body of C.

(represented by SLD-resolutions) are mutually exclusive, the probability of a query is then defined to be the sum of the probabilities of the SLD-resolutions for the query.

Some other works are based on Bayesian networks [53, 54]. Bayesian networks are directed acyclic graphs in which nodes represent random variables, and edges represent conditional dependencies between the variables. Nodes which are not connected are conditionally independent of each other. A node X is conditionally independent of its ancestors given X's parents. The inference on a Bayesian network produces a probability distribution for a logical variable being queried or joint probability distribution for multiple variables [53, 54]. Bayesian networks have been successfully implemented in many expert systems, such as the SMILE system from the University of Pittsburgh [11], and applied in many applications, such as traffic accident reconstruction [8].

Kersting and Raedt's Bayesian logic programs [29] are an attempt of integrating logic programming with Bayesian networks. In Bayesian logic programs, atoms are not logical variables which are true or false. Instead, they are random variables which can be assigned values in any associated domain. Rules in Bayesian logic programs describe probabilistic dependencies instead of implications. Indeed, each rule is associated with a conditional probability distribution (cpd). From this perspective, Bayesian logic programs are close to representations of Bayesian networks *in the form of* logic programs. The main difference between Bayesian networks and Bayesian logic programs is that in the latter there may be multiple rules with the same head, where each of the rules is assigned a cpd of the head given the atoms in the rule body. [29] proposed certain combination methods to combine these cpds to get the cpd

of the head given all its parents. However, there is no justification in probability theory for these combination methods, e.g., the methods which combine $P(a \mid b, c)$ and $P(a \mid d, e)$ to produce $P(a \mid b, c, d, e)$.

Markov logic networks (MLNs) [65] are based on the graphical model of Markov networks [32, 55]. An MLN is a set of pairs (F_i, w_i) where F_i is a first-order formula that consists of atoms and connectives $\wedge$, $\vee$, $\neg$, and w_i is a real number signifying the weight attached to F_i. Just like that a Markov network represents a joint probability distribution of all the random variables in it, an MLN specifies a probability distribution over all interpretations of the Herbrand base:

$$P(X = x) = \frac{1}{Z} exp\Big(\sum_i w_i n_i(x)\Big), \quad \text{where } Z = \sum_x exp\Big(\sum_i w_i n_i(x)\Big)$$

where X is the Herbrand base, x is an interpretation of X, $n_i(x)$ is the number of true groundings of F_i under x. A Markov network is an undirected graphical model, so it can represent cyclic dependencies that a Bayesian network can not. On the other hand, a Bayesian network can represent induced dependencies while a Markov network can not. Such differences also exist between MLN and other frameworks based on directed graphical models which we mentioned previously. A drawback of Markov logic networks is the high computational complexity to calculate the exact probabilities. Consequently, the authors of [65] proposed to use Markov Chain Monte Carlo (MCMC) [21] to approximate inference. Weights on clauses can be learned from relational databases by iteratively optimizing a pseudo-likelihood measure.

Ng and Subrahmanian introduced the first probabilistic characterization of logic programming semantics in [48]. The language in [48] is syntactically similar to annotated logic programming [4, 5, 31, 78]. A clause is of the form $A : \mu \leftarrow F_1 : \mu_1 \wedge ... \wedge F_n : \mu_n$ where A is an atom, $F_1, ..., F_n$ are

formulas consisting of atoms and logic connectives $\wedge$ and $\vee$, and $\mu, \mu_1, ..., \mu_n$ are probability intervals such that $\mu, \mu_1, ..., \mu_n \subseteq [0, 1]$. Such a clause means that if the probability interval of F_i is in μ_i then the probability interval of A is in μ. For an atom A, the probability interval of A is given by $\bigcap M_A$ where M_A is the set of intervals given by the applicable rules that have A in head. Operators $\otimes$ and $\oplus$ are defined to compute the probability intervals of $F_1 \wedge F_2$ and $F_1 \vee F_2$, respectively, from the probability intervals of F_1 and F_2. Later Dekhtyar and Subrahmanian presented another annotation-based probabilistic program [9], which allows choices of conjunction and disjunction operators based on known dependencies between events.

In [38, 39] by Lukasiewicz, a program KB is a set of conditional constraints $(H \mid B)[c_1, c_2]$ where H and B are conjunctions of atoms and $0 \leq c_1 < c_2 \leq 1$. Such a constraint means that the conditional probability $P(H \mid B)$ lies in the interval $[c_1, c_2]$. If the program contains two conditional constraints $(H \mid B)[c_1, c_2]$ and $(H \mid B)[c_3, c_4]$ such that $[c_1, c_2] \cap [c_3, c_4] = \varnothing$ then the program collapses. Queries on a program can be on 1) whether a certain conditional constraint is entailed by the program; 2) what are the tight bounds of a conditional probability; 3) what variable substitutions satisfy a conditional constraint. An approximation technique which uses van Emden's quantitative deduction [17] was also presented to compute a candidate model of a program.

The works reviewed in this section all fall in the category of probabilistic logic programs because they also use probabilities to model uncertainty. As a consequence, they share a number of shortcomings:

1. They can not model the lack of knowledge. For example, let us consider a medical expert system for diagnosing a patient based on certain symptoms, and suppose that the available evidence is scarce. Suppose also that the system concludes that the certainty of diarrhea is 10%.

In the probabilistic context, one would claim that this person doesn't have diarrhea with 90% chance; however, the actual situation might be that the system supports the diarrhea diagnosis with 10% confidence, and the "inconclusive" diagnosis with confidence of 90%. The usage of probability intervals [47, 48, 49] does remedy this problem to a certain degree, but these interval models are not as powerful as the models that represent the lack of knowledge explicitly as we will see in the case of belief functions [10, 71].

2. When there are conflicting sources of information, it is difficult to resolve such a conflict while staying within a probabilistic context. For example, suppose a person has two sources of information, one saying that the probability of the stock market going up is in the interval $[55\%, 60\%]$, while the other saying that this probability is in the interval $[20\%, 30\%]$. What should this person infer from such sources? Various methods have been proposed: for example, calculating the new lower probability as $max(55\%, 20\%)$ and the upper probability as $min(1, 60\% + 30\%)$ [45]. However, none of these methods has a justification in the probabilistic context.

3. When the application does not involve a random event, probabilistic inference becomes questionable. For example, suppose that a logic program represents a decision process of judging a suspect based on criminal evidence, and that the conclusion is 70% certainty for the suspect being guilty and 30% certainty for the suspect being innocent. This does not mean that the suspect's past behavior is a random event with two possible outcomes. Thus, the 70% and 30% certainty factors cannot be viewed as probabilities.

2.2.4 Frameworks Based on Dempster-Shafer Theory

Another popular category of quantitative logic is based on Dempster-Shafer theory of evidence [10, 71] (also called the theory of belief functions). In works in this category, certainty factors are interpreted as degrees of belief instead of probabilities. From a mathematical perspective, belief functions are a generalization of probability functions [71], in that all probability functions satisfy the definition of belief functions and in fact represent a special case of complete knowledge. However, the meaning of belief functions is fundamentally non-probabilistic, as we will discuss later in Section 2.3.

There has been great progress in applying the belief function methodology to reasoning [1, 2, 3, 30, 36, 45, 66, 75, 74]. Shennoy and Shafer proposed an abstract framework [75] in which the relations among the variables are represented by joint belief functions. Their framework requires that the relations among variables form a hypertree, and enables local computation of marginals of joint belief functions. Based on [75] Shennoy later constructed a framework called valuation-based system (VBS) [74] for managing uncertainty in expert systems. Lee [36] proposed an extended relational database model which can model uncertainty with Dempster-Shafer theory. Bergsten and Schubert [3] proposed an efficient algorithm for evidential reasoning in transitions between states, which can be modeled by *complete* directed acyclic graphs. Xu and Smets [87] proposed a framework in which variables are connected with directed edges and each edge is annotated with a conditional belief function. Then they presented an algorithm to propagate belief functions on the graph. However, their framework lacks the ability to model the causality represented by rules with conjunctions in body. [66] applied Dempster-Shafer theory to combine evidence in relevance feedback in the area of information retrieval.

The syntax in [3, 36, 66, 75, 74] do not allow rules, so these works could not model implication or causality of uncertainty. Since we are mainly interested in rule-based reasoning with uncertainty, we will focus on the review of rule-based frameworks [1, 2, 30, 45].

In [30] by Kifer and Li, the syntax falls in the category of annotated logic programming [4, 5, 31, 78]. A rule is of the form $H : f(\mu_1, ..., \mu_n) \leftarrow B_1 : \mu_1 \wedge ... \wedge B_n : \mu_n$, where $H, B_1, ..., B_n$ are atoms, $\mu_1, ..., \mu_n$ are certainty variables or constants, and f is a function which maps a set of certainty value in $[0, 1]$ to a single certainty value in $[0, 1]$. A Herbrand interpretation of a program assigns a certainty value to each atom in the Herbrand base. An annotated atom $A : \mu$ is true under a Herbrand interpretation I iff $A : \nu \in I$ such that $\nu \geq \mu$. I satisfies a rule iff whenever all the annotated atoms in the rule body are true, the rule head is also true. I is a model of a program iff 1) I satisfies every ground instance of every rule, and 2) for each set R of rules with the same atom A in head, I satisfies $A : f_c(\{\alpha \mid A : \alpha \leftarrow Body \in R,\ Body \text{ is true under } I\})$ where f_c is a combination function that maps a set of certainty value to a single certainty value. The semantics has a strong restriction that, to be combinable, supporting evidences should be independent of each other. For example, the semantics can not handle rules $p \leftarrow a, b$ and $p \leftarrow b, c$.

In Baldwin's support logic programming[1, 2], each rule is of the form $Head$:- $Body : [w, v]$ or the form $Head : [w, v]$ where $w, v \in [0, 1]$. Given $A : [w_A, v_A]$ and $B : [w_B, v_B]$, it is inferred that $A \wedge B : [w_A \cdot w_B, v_A \cdot v_B]$, $A \vee B : [w_A + w_B - w_A \cdot w_B, v_A + v_B - v_A \cdot v_B]$ and $NOT A : [1 - v_A, 1 - w_A]$. Given A :- $B : [w, v]$ and $B : [w_B, v_B]$, it is inferred that $A : [w \cdot w_B, 1 - (1 - v_B) \cdot w_B]$. When there are two (or more) rules with the same head A, after getting $A : [w_1, v_1]$ and $A : [w_2, v_2]$ separately from the supporting rules, Dempster's combination rule is used to combine the two evidences into some

$A : [w_A, v_A]$. The above inference mechanism computes a certainty factor of the form $[w, v]$ for each atom. Unlike [30], Baldwin's work does not restrict the bodies of supporting rules for the same atom to be independent. However, in the case that the bodies of supporting rules overlap or correlate, correlations are not handled properly in the combination procedure, as will be discussed in Section 3.1.

In Section 6 of [45], Ng proposed an annotated logic programming framework based on Dempster-Shafer theory. Annotations in [45] are in the form of $[w, v]$ where w and v can be 1) variables, 2) constants in $[0, 1]$, or 3) a term $f(\delta_1, ..., \delta_n)$ where f is a function and $\delta_1, ..., \delta_n$ are variables or constants in $[0, 1]$. A clause is in the form of $A : \mu \leftarrow A_1 : \mu_1 \wedge ... \wedge A_n : \mu_n$ where A, $A_1, ..., A_n$ are atoms, and $\mu, \mu_1, ..., \mu_n$ are annotations. Each such clause serves as an evidence for A to the degree of μ when the body of this clause is satisfied. When there are multiple clauses supporting A, Dempster's combination rule is used to combine the evidences. [45] has the same shortcoming as [1, 2] in handling correlations between overlapping supporting rules.

As discussed in this section, to our best knowledge, the existing rule-based frameworks on belief functions all have certain limitations that prevent them from performing large-scale reasoning in realistic settings. Some of them require restrictions on logic dependencies to avoid combining sources that are not independent [30, 75]. Others have difficulty in combining different evidence sources which are correlated to a certain degree due to overlapping belief propagation paths. For example, consider two rules $[0.5, 1]A \text{ :- } B \wedge C$ and $[0.8, 1]A \text{ :- } B \wedge D$. The previous works [1, 2] and [45, 35] [3] would

[3] A similar example in the syntax of [45] is a pair of rules $A : [0.5 \cdot v_1 \cdot v_2, 1 \cdot v_1 \cdot v_2] \text{ :- } B : [v_1, w_1] \wedge C : [v_2, w_2]$ and $A : [0.8 \cdot v_1 \cdot v_3, 1 \cdot v_1 \cdot v_3] \text{ :- } B : [v_1, w_1] \wedge D : [v_3, w_3]$. A similar example in the syntax of [35] is a pair of rules $A \text{ :- } B \wedge C$ with propagation function $fp_1([v_1, w_1], [v_2, w_2]) = [0.5 \cdot v_1 \cdot v_2, 1 \cdot v_1 \cdot v_2]$ and $A \text{ :- } B \wedge D$ with propagation function $fp_2([v_1, w_1], [v_2, w_2]) = [0.8 \cdot v_1 \cdot v_2, 1 \cdot v_1 \cdot v_2]$.

directly combine the certainty factors of A propagated from the two rules, assuming the two sources are independent. Obviously this assumption does not hold. For a more extreme example, consider a logic program with 1000 rules $[1, 1]A$:- B_i for $i = 1, 2, \cdots, 1000$ and another 1000 rules $[1, 1]B_i$:- C for $i = 1, 2, \cdots, 1000$. If given the fact that the certainty of C is $[0.5, 1]$, [1, 2] would conclude that A is true with a certainty value that is almost 1, which is far from the reasonable conclusion that the certainty of A should be $[0.5, 1]$. As will be demonstrated in Section 3.1 and Section 3.5, the above problems are addressed by our new approach.

2.3 Dempster-Shafer Theory

In BLP, uncertainty is represented with belief functions from Dempster-Shafer Theory [10, 71]. In this section we review some basic concepts.

Dempster-Shafer Theory, also known as Theory of Evidence or Theory of Belief Function, was first proposed by Dempster [10] and then extended by Shafer [71]. It has two components: belief representation and combination of evidence.

2.3.1 Representing Belief

In Probability Theory, probabilities are assigned to mutually exclusive states. In Dempster-Shafer Theory, evidence (also known as degree of belief, certainty, or support) is associated with *sets of* mutually exclusive states. For example, the set of states {a , b , c } may have an associated degree of belief 0.4, which means that either a or b or c is true with certainty 0.4. This statement does *not* imply anything about the individual truth of a , b , or c , or about any of the sets {a , b }, {a , c }, or {b , c }.

There are three important functions in Dempster-Shafer theory: the basic probability assignment function (bpa function or mass function), the belief function, and the plausibility function.

Let $\mathcal{U}$ be the universal set—a set of all possible mutually exclusive states under consideration. The power set $\mathbb{P}(\mathcal{U})$ is the set of all possible sub-sets of $\mathcal{U}$, including the empty set, $\varnothing$.

A **mass function** is a mapping $\texttt{mass}: \ \mathbb{P}(\mathcal{U}) \ \longrightarrow \ [0,1]$ such that

$$\begin{aligned} \texttt{mass}(\varnothing) &= 0 \\ \sum_{S \in \ \mathbb{P}(\mathcal{U})} \texttt{mass}(S) &= 1 \end{aligned}$$

$\texttt{mass}(S)$ expresses the proportion of all relevant and available evidence that supports the claim that the actual true state belongs to the set $S \subseteq \mathcal{U}$ and to no proper subset of S. If there also exists evidence to support that the actual state belongs to a subset S' of S, then $\texttt{mass}(S')$ can also be non-zero.

The **belief** associated with the set S is defined as the sum of all the masses of S's subsets:

$$\texttt{belief}(S) = \sum_{S' \subseteq S} \texttt{mass}(S')$$

The **plausibility** $\texttt{plausibility}(S)$ is the sum of all the masses of the sets S' that intersect the set S:

$$\texttt{plausibility}(S) = \sum_{S' \cap S \neq \varnothing} \texttt{mass}(S')$$

By the above definitions we have $\texttt{plausibility}(S) = 1 - \texttt{belief}(S^c)$ and $\texttt{belief}(S) \leq \texttt{plausibility}(S)$, where $S^c = \mathcal{U} - S$.

It also follows from the above definitions that with one of the three (mass function, belief function and plausibility function) the other two can be deduced.

2.3.2 Dempster's Combination Rule

Dempster's combination rule [10, 71] addresses the issue of how to combine two independent sets of mass assignments.

Given two independent mass functions $\texttt{mass}_1$ and $\texttt{mass}_2$, the combination of them, $\texttt{mass}_3$, is calculated as follows:

$$\begin{aligned} \texttt{mass}_3(\varnothing) &= 0 \\ \texttt{mass}_3(A) &= \frac{1}{1-K} \sum_{A=B\cap C\neq\varnothing} \texttt{mass}_1(B)\texttt{mass}_2(C) \end{aligned}$$

where $K = \sum_{B\cap C=\varnothing} \texttt{mass}_1(B)\texttt{mass}_2(C)$.

In the above formula, K is the amount of mass associated with the conflict between the two mass sets. By applying the normalization factor, $1-K$, the mass associated with conflict is distributed to other states.

Dempster's combination rule emphasizes the agreement between multiple sources and ignores correlation and conflict through a normalization factor. Some argue that ignoring these aspects may lead to questionable results in situations that different rules derive information with significant conflict. In Lotfi Zadeh's review [91] of Shafer's book [71], Zadeh provided the following example of questionable results from applying Dempster's combination rule. Suppose that a patient is seen by two physicians regarding the patient's neurological symptoms. The first doctor believes that the patient has either meningitis with probability of 0.99 or a brain tumor, with probability of 0.01. The second physician believes the patient actually suffers from a concussion with probability of 0.99 but admits the possibility of a brain tumor with probability of 0.01. Using the values to calculate the $\texttt{mass}(\textit{brain tumor})$ with Dempster's rule, we find that $\texttt{mass}(\textit{brain tumor}) = \texttt{belief}(\textit{brain tumor}) = 1$. This diagnosis is considered to be very unlikely by both physicians.

To avoid this problem, alternatives to the normalization process has been proposed, including Yager's modified Dempster's rule [88], Dubois and Prade's disjunctive consensus rule [13, 15], Zhang's center combination rule [95], Inagaki's unified combination rule [27], etc.

BLP supports a family of combination methods, including the combination rules in [88, 44, 70], and does not commit to any particular one. As a special case, Dempster's combination rule and its extensions can be used when appropriate.

Chapter 3

Belief Logic Programming – Syntax and Semantics

3.1 Motivating Example: Traffic Advisory

Let us motivate with a simple traffic advisory example.

A group of Stony Brook students is planning a trip to see a Broadway musical. Normally, it takes 1.5 hours by car to get to Manhattan, but the students know that Long Island Expressway is not called by the locals "the longest parking lot in America" for nothing. The students consult a traffic service, which integrates information from several independent information sources to provide traffic advisory along various travel routes. Let us assume that these sources are:

- weather forecast (rain, snow, fog)
- social activity (parades, motorcades, marathons)
- police activity (accidents, emergencies)

- roadwork

The service uses the following rules (which are simplified for this example) to generate advisories:

1. If the weather is bad, and there is roadwork along the route, the likelihood of a delay is 0.9.

2. If there is roadwork and social activities along the route, the likelihood of a delay is 0.8.

3. If there is roadwork and police activity along the route, the likelihood of a delay is 0.99.

These rules are expressed in BLP as shown below, where $?r$ is the variable that represents the travel route.

$$
\begin{aligned}
&[0.9, 1] \quad delay(?r) \ \texttt{:-}\ roadwork(?r) \wedge bad\_weather(?r) \\
&[0.8, 1] \quad delay(?r) \ \texttt{:-}\ roadwork(?r) \wedge social\_act(?r) \\
&[0.99, 1] \quad delay(?r) \ \texttt{:-}\ roadwork(?r) \wedge police\_act(?r)
\end{aligned}
$$

The service generates advisories expressed as the likelihood of delays along the routes of interest. The students do not want to miss the show due to traffic, but they also have conference deadlines and so do not want to leave too early. They decide that if the advisory says that the likelihood of delays is between 0.2 and 0.4 then they add one extra hour to the trip time. If the likelihood is between 0.4 and 0.6, then they add two hours, and if the likelihood is over 0.6 then they take a train.

The key observation here is that the three rules used in generating the advisory are not independent—they all rely on the roadwork information from Department of Transportation. Our intuition suggests that predictions based

on the independence assumption might cost our student a Broadway show, a few hours of sleep, or a conference paper.

As mentioned in the introduction, the novelty of our approach is that it does not assume that the information sources are independent and instead properly correlates inferences obtained using the rules that represent these sources. In Section 3.5, we will return to this example and show that, in contrast to the approaches which are unable to correlate information sources, our approach improves the quality of the advisory and could have helped the students avoid unnecessary grief.

3.2 Syntax

Let $\mathcal{L}$ be a first order language containing an infinite set of variables, denoted by alphanumeric symbols prefixed with the question mark ?; a finite set of constant and predicate symbols, both denoted by alphanumeric symbols that are not prefixed with a "?"; and the connectives: $\wedge$ for conjunction, $\vee$ for disjunction, `:-` for rule implication, and $\overline{A}$ for negation of A. $\mathcal{L}$ also contains two special propositions: `true` which is always interpreted as true and `false` which is always interpreted as false. A **term** is a variable or a constant. An **atomic formula**, also called **atom**, is composed of the form $p(t_1, t_2, ..., t_n)$, where $n \geq 0$, p is an n-ary predicate symbol in $\mathcal{L}$ and t_i, $1 \leq i \leq n$, are all terms in $\mathcal{L}$. If A is an atom then A is said to be a **positive literal** and $\overline{A}$ is said to be a **negative literal**.

A **belief logic program** (or a **blp**, for short) is a set of annotated rules. Each **annotated rule** has the following format:

$$[v, w]\ X\ \texttt{:-}\ Body$$

where the rule head X is an atom and $Body$ is a Boolean combination of

atoms, i.e., a formula composed out of atoms by conjunction, disjunction, and negation. The annotation $[v, w]$ is called a **belief factor**, where v and w are real numbers such that $0 \leq v \leq w \leq 1$.

The informal meaning of the above rule is that if *Body* is true, then this rule supports X to the degree v and $\overline{X}$ to the degree $1 - w$. The difference, $w - v$, is the *information gap* (or the degree of ignorance) with regard to X.

Note that, in keeping with the theory of evidence, BLP uses what is known as explicit negation (or, strong negation) [52] rather than negation as failure. That is, if nothing is known about A, it only means that there is no evidence that A holds; it does not mean that the negation of A holds.

An annotated rule of the form $[v, w]$ X `:- true` is called an **annotated fact**; it is often written simply as $[v, w]$ X. In the remainder of this dissertation we will deal only with annotated rules and facts and refer to them simply as rules and facts.

Example 3.1 The following blp P_1 consists of three rules, two of which are facts.

P_1:

$$\begin{array}{rlll} [0.5, 1] & a & & \\ [0.8, 1] & b & \texttt{:-} & a \\ [0.6, 0.8] & b & & \end{array}$$

□

An expression is called **ground** if it does not contain any variables. The set of all ground terms in $\boldsymbol{P}$, which is the set of constant symbols here, is called the **Herbrand Universe** of $\boldsymbol{P}$ (denoted by $U_{\boldsymbol{P}}$). The set of all ground atoms constructed from the constant and predicate symbols in $\boldsymbol{P}$ is called the **Herbrand Base** of $\boldsymbol{P}$ (denoted by $B_{\boldsymbol{P}}$). If F is a formula (atom, literal,

rule, respectively), then by a **ground instance** of F we mean any ground formula (atom, literal, rule, respectively) obtained from F by uniformly substituting ground terms for all variables. The **ground form** of a blp $\boldsymbol{P}$, denoted $ground(\boldsymbol{P})$, is the set of all ground instances of all the rules in $\boldsymbol{P}$.

Definition 3.1 *Given a blp $\boldsymbol{P}$, an atom X is said to **depend** on an atom Y*

- *directly, if X is the head of a rule R in $\boldsymbol{P}$and Y occurs in the body of R;*
- *indirectly, if X depends on Z, and Z depends on Y.* □

A blp $\boldsymbol{P}$ is said to be **cyclic** if in $ground(\boldsymbol{P})$ there is an atom that depends on itself directly or indirectly, and **acyclic** otherwise.

Example 3.2 P_1 in Example 3.1 is acyclic. The following blps P_2 and P_3 are cyclic.

P_2:

$$\begin{aligned} [0.5, 1] \quad & a \\ [0.8, 1] \quad & b \ \texttt{:-} \ a \\ [0.6, 0.8] \quad & b \ \texttt{:-} \ b \end{aligned}$$

P_3:

$$\begin{aligned} [0.8, 1] \quad & disease(?X) \ \texttt{:-} \ test\_pos(?X). \\ [0.6, 1] \quad & disease(?X) \ \texttt{:-} \ contact(?X, ?Y) \wedge disease(?Y). \\ [1, 1] \quad & test\_pos(a). \\ [1, 1] \quad & test\_pos(b). \\ [1, 1] \quad & contact(a, b). \\ [1, 1] \quad & contact(b, a). \end{aligned}$$

□

In the remainder of the dissertation, except in Chapter 5, we talk about only acyclic blps. That is to say, there can be no circular dependency among

ground atoms. Most other works in this area, such as [1, 3, 74, 63], make the same assumption. The difficulties in allowing cycles, and an extension of BLP that drops this restriction are described in Chapter 5.

3.3 Combination Functions

We first define the notion of combination function, which will help determine the *combined* belief from multiple sources of evidence.

Definition 3.2 *Let D be the set of all sub-intervals of $[0,1]$, and $\Phi : D \times D \rightarrow D$ be a function. We say that Φ is a* ***belief combination function*** *if Φ is associative and commutative.* □

Let us represent $\Phi([v_1, w_1], [v_2, w_2])$ as $[V(v_1, w_1, v_2, w_2), W(v_1, w_1, v_2, w_2)]$. A useful common-sense restriction on combination functions is that the functions V and W above are monotonically increasing in each of their four arguments, v_1, w_1, v_2, w_2, but this is not required for our results.

Other useful common-sense restrictions on combination functions include: $\Phi([0,0],[0,0]) = [0,0]$; $\Phi([1,1],[1,1]) = [1,1]$; $\Phi([0,1],[V,W]) = [V,W]$; $\Phi([0,0],[1,1]) = [0,1]$. Among them the third and fourth are not required for our results. It is worth noting that the special case of combining $[0,0]$ and $[1,1]$ corresponds to the scenario when two pieces of evidence irreconcilably contradict each other.

Due to the associativity of Φ, we can extend it from two arguments to three and more arguments as follows: for any $k > 2$,

$$\Phi\big([v_1, w_1], ..., [v_k, w_k]\big) \;=\; \Phi\Big(\Phi\big([v_1, w_1], ..., [v_{k-1}, w_{k-1}]\big), [v_k, w_k]\Big)$$

For convenience, we also extend Φ to the nullary case and the case of a single argument as follows: $\Phi() = [0,1]$ and $\Phi([v,w]) = [v,w]$.

Note that the order of arguments in a belief combination function is immaterial, since such functions are commutative, so we often write such functions as functions on multisets of intervals, e.g., $\Phi(\{[v_1, w_1], ..., [v_k, w_k]\})$.

As mentioned earlier, there are many ways to combine evidence and so there are many useful belief combination functions. Different functions can be used for different application domains and even for different types of data within the same domain.

3.3.1 Dempster's Combination Function

Given two independent sources of evidence, the first source gives support of amount v_1 to A, $1 - w_1$ to $\overline{A}$, the second source gives support of amount v_2 to A, $1 - w_2$ to $\overline{A}$. As discussed in Section 2.3.2, Dempster's combination rule combines the two sources of evidence as follows:

	A v_2	$\overline{A}$ $1 - w_2$	*ignorance* $w_2 - v_2$
A v_1	A $v_1 \cdot v_2$	A *and* $\overline{A}$ $v_1 \cdot (1 - w_2)$	A $v_1 \cdot (w_2 - v_2)$
$\overline{A}$ $1 - w_1$	$\overline{A}$ *and* A $(1 - w_1) \cdot v_2$	$\overline{A}$ $(1 - w_1) \cdot (1 - w_2)$	$\overline{A}$ $(1 - w_1) \cdot (w_2 - v_2)$
ignorance $w_1 - v_1$	A $(w_1 - v_1) \cdot v_2$	$\overline{A}$ $(w_1 - v_1) \cdot (1 - w_2)$	*ignorance* $(w_1 - v_1) \cdot (w_2 - v_2)$

"A *and* $\overline{A}$" and "$\overline{A}$ *and* A" are two cases of contradiction, so the support

assigned to them need to be discarded. After normalization we get

$$\begin{aligned} v &= \text{the support to } A \\ &= \frac{1}{K} \cdot \Big(v_1 \cdot v_2 + v_1 \cdot (w_2 - v_2) + v_2 \cdot (w_1 - v_1) \Big) \\ w &= 1 - \text{the support to } \overline{A} \\ &= 1 - \frac{1}{K} \cdot \Big((1 - w_1) \cdot (1 - w_2) + (w_1 - v_1) \cdot (1 - w_2) \\ &\qquad + (1 - w_1) \cdot (w_2 - v_2) \Big) \end{aligned}$$

where $K = 1 - v_1 \cdot (1 - w_2) - v_2 \cdot (1 - w_1)$.

In the special case of $\{[v_1, w_1], [v_2, w_2]\} = \{[0, 0], [1, 1]\}$, which is, when two pieces of evidence irreconcilably contradict each other, the above K equals 0. In this case, we let the combined support to A be 0, and the combined support to $\overline{A}$ also be 0, in other words, we let v be 0 and w be 1.

From the above, we get **Dempster's combination function,** Φ^{DS}, as follows:

$$\begin{aligned} \Phi^{DS}([0, 0], [1, 1]) &= [0, 1] \\ \Phi^{DS}([v_1, w_1], [v_2, w_2]) &= [v, w] \qquad \text{if } \{[v_1, w_1], [v_2, w_2]\} \neq \{[0, 0], [1, 1]\} \end{aligned}$$

where

$$v = \frac{v_1 \cdot w_2 + v_2 \cdot w_1 - v_1 \cdot v_2}{1 + v_1 \cdot w_2 + v_2 \cdot w_1 - v_1 - v_2}$$

$$w = \frac{w_1 \cdot w_2}{1 + v_1 \cdot w_2 + v_2 \cdot w_1 - v_1 - v_2}$$

Dempster's combination rule discussed in Section 2.3.2 is under an assumption that the two mass functions to be combined are independent to each other. Similarly, Dempster's combination function in BLP is under an assumption that all the rules with the same head are considered to be independent sources of evidence with respect to each other.

3.3.2 Other Combination Functions

Some other popular combination functions include

Maximum:

$$\Phi^{max}([v_1, w_1], [v_2, w_2]) = [max(v_1, v_2), max(w_1, w_2)]$$

Minimum:

$$\Phi^{min}([v_1, w_1], [v_2, w_2]) = [min(v_1, v_2), min(w_1, w_2)]$$

Average:

$$\Phi^{avg}([v_1, w_1], [v_2, w_2]) = [avg(v_1, v_2), avg(w_1, w_2)]$$

Independent:

$$\Phi^{ind}([v_1, w_1], [v_2, w_2]) = [min(1, v_1 + v_2 - v_1 \cdot v_2), min(1, w_1 + w_2 - w_1 \cdot w_2)]$$

and many more. We can also use combination functions obtained from the alternatives [88, 44, 70] of Dempster's rule.

3.4 Declarative Semantics

Given a blp $\boldsymbol{P}$, the definitions of Herbrand Universe $U_{\boldsymbol{P}}$ and Herbrand Base $B_{\boldsymbol{P}}$ of $\boldsymbol{P}$ are the same as in the classical case. As usual in logic programming, the easiest way to define a semantics is by considering *ground* (i.e., variable-free) rules. We assume that each atom $X \in B_{\boldsymbol{P}}$ has an associated belief combination function, denoted Φ_X. Intuitively, Φ_X is used to help determine the *combined* belief in X accorded by the rules in $\boldsymbol{P}$ that support X.

Definition 3.3 *A **truth valuation** over a set of atoms α is a mapping from α to $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$. The set of all possible valuations over α is denoted as $\mathcal{TV}al(\alpha)$.*

*A **truth valuation** I for a blp $\boldsymbol{P}$ is a truth valuation over $B_{\boldsymbol{P}}$. Let $\mathcal{TV}al(\boldsymbol{P})$ denote the set of all the truth valuations for $\boldsymbol{P}$, $\mathcal{TV}al(\boldsymbol{P}) = \mathcal{TV}al(B_{\boldsymbol{P}})$.* □

It is easy to see that $\mathcal{TV}al(\boldsymbol{P})$ has $3^{|B_P|}$ truth valuations.

Example 3.3 Consider P_1 from Example 3.1:

$$\begin{array}{rlll} [0.5, 1] & a & & \\ [0.8, 1] & b & \texttt{:-} & a \\ [0.6, 0.8] & b & & \end{array}$$

There are 9 truth valuations in $\mathcal{TV}al(P_1)$, namely $I_1, \ldots, I_9$:

$$I_1(a) = t, \quad I_1(b) = t$$
$$I_2(a) = t, \quad I_2(b) = f$$
$$I_3(a) = f, \quad I_3(b) = t$$
$$I_4(a) = f, \quad I_4(b) = f$$
$$I_5(a) = u, \quad I_5(b) = t$$
$$I_6(a) = t, \quad I_6(b) = u$$
$$I_7(a) = u, \quad I_7(b) = f$$
$$I_8(a) = f, \quad I_8(b) = u$$
$$I_9(a) = u, \quad I_9(b) = u$$

□

Let α be a set of atoms, we use $\mathcal{B}ool(\alpha)$ to denote the set of all Boolean formulas constructed out of these atoms (i.e., using $\wedge$, $\vee$, and negation).

Definition 3.4 *Let α be a set of atoms, I a truth valuation over α, and F a boolean formula in $\mathcal{B}ool(\alpha)$. $I(F)$ is defined as in Lukasiewicz's three-valued logic: $I(A \vee B) = max\big(I(A), I(B)\big)$, $I(A \wedge B) = min\big(I(A), I(B)\big)$, and*

$I(\overline{A}) = \neg I(A)$, *where* $\mathbf{f} < \mathbf{u} < \mathbf{t}$ *and* $\neg\mathbf{t} = \mathbf{f}$, $\neg\mathbf{f} = \mathbf{t}$, $\neg\mathbf{u} = \mathbf{u}$. *We say that* $I \models F$ *if* $I(F) = \mathbf{t}$. □

Definition 3.5 *Let* α *and* β *be two sets of atoms such that* $\alpha \subseteq \beta$. *Also let* I *be a truth valuation over* β. *The* ***restriction*** *of* I *to* α, *denoted* $I\mid_\alpha$, *is defined to be the truth valuation over* α *such that* $\forall X \in \alpha,\ I\mid_\alpha (X) = I(X)$.

□

Definition 3.6 *Let* I_1 *be a truth valuation over a set of atoms* α *and* I_2 *be a truth valuation over a set of atoms* β, *where* $\alpha \cap \beta = \varnothing$. *The* ***union*** *of* I_1 *and* I_2, *denoted* $I_1 \uplus I_2$, *is the truth valuation over* $\alpha \cup \beta$ *such that* $(I_1 \uplus I_2)\mid_\alpha = I_1$ *and* $(I_1 \uplus I_2)\mid_\beta = I_2$. *It is easy to see that* $I_1 \uplus I_2 = I_2 \uplus I_1$. □

In the remainder of this dissertation, if α is a set of atoms and $\tau \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ is a truth value, we will use $\{\alpha \rightarrow \tau\}$, to denote the *constant truth valuation* over α, which maps every atom $X \in \alpha$ to τ. If $\alpha = \{A\}$ is a singleton set, we will simply write $\{A \rightarrow \tau\}$ instead of $\{\{A\} \rightarrow \tau\}$.

Definition 3.7 *A* ***support function*** *for a set of atoms* α *is a mapping* m_α *from* $\mathcal{TV}al(\alpha)$ *to* $[0, 1]$ *such that*

$$\sum_{I \in \mathcal{TV}al(\alpha)} m_\alpha(I) = 1$$

The atom-set α *is called the* ***base*** *of* m_α. *A* ***support function*** *for a blp* $\boldsymbol{P}$ *is a mapping* m *from* $\mathcal{TV}al(\boldsymbol{P})$ *to* $[0, 1]$ *such that*

$$\sum_{I \in \mathcal{TV}al(\boldsymbol{P})} m(I) = 1$$

□

Support functions, defined above, are always associated with mass functions of Dempster-Shafer theory, as will be discussed in Section 3.6. In

Dempster-Shafer theory, every mass function has a corresponding belief function and, similarly, BLP support functions are associated with belief functions, defined next.

Definition 3.8 *Recall that $\mathcal{B}ool(B_{\boldsymbol{P}})$ denotes the set of all Boolean formulas composed out of the atoms in $B_{\boldsymbol{P}}$. A mapping $\texttt{bel} : \mathcal{B}ool(B_{\boldsymbol{P}}) \longrightarrow [0,1]$ is said to be a **belief function** for $\boldsymbol{P}$ if there exists a support function m for $\boldsymbol{P}$, such that for all $F \in \mathcal{B}ool(B_{\boldsymbol{P}})$*

$$\texttt{bel}(F) = \sum_{I \in \mathcal{TV}al(\boldsymbol{P}) \text{ such that } I \models F} m(I)$$

□

Belief functions can be thought of as *interpretations* of belief logic programs. However, as usual in deductive database and logic programming, we are interested not just in interpretations, but in models. We define this next.

Definition 3.9 *Let $\boldsymbol{P}$ be a blp and I a truth valuation, we define **$\boldsymbol{P}$'s reduct under** I to be $\boldsymbol{P}_I = \{R \mid R \in \boldsymbol{P}, I \models Body(R)\}$, where $Body(R)$ denotes the body of the rule R.*

*Let $\boldsymbol{P}(X)$ denote the set of rules in $\boldsymbol{P}$ with the atom X in the head. **$\boldsymbol{P}$'s reduct under I with X as head** is defined as $\boldsymbol{P}_I(X) = \boldsymbol{P}_I \cap \boldsymbol{P}(X)$. Thus, $\boldsymbol{P}_I(X)$ is simply that part of the reduct $\boldsymbol{P}_I$, which consists of the rules that have X as their head.* □

We now define a measure for the degree by which I is supported by $\boldsymbol{P}(X)$.

Definition 3.10 *Let $\boldsymbol{P}$ be a blp, I a truth valuation for $\boldsymbol{P}$, and X an atom in $B_{\boldsymbol{P}}$, we define $s_{\boldsymbol{P}}(I, X)$, called the **$\boldsymbol{P}$-support for** X **in** I, as follows.*

1. *If $\boldsymbol{P}_I(X) = \phi$, then*

- *If* $I(X) = \mathbf{t}$ *or* $I(X) = \mathbf{f}$*, then* $s_{\boldsymbol{P}}(I, X) = 0$*;*
- *If* $I(X) = \mathbf{u}$*, then* $s_{\boldsymbol{P}}(I, X) = 1$*.*

2. *If* $\boldsymbol{P}_I(X) = \{R_1, \ldots, R_n\}$*,* $n > 0$*, let* $[v, w]$ *be the result of applying* Φ_X *to the belief factors of the rules* $R_1, \ldots, R_n$*. Then*

   - *If* $I(X) = \mathbf{t}$*, then* $s_{\boldsymbol{P}}(I, X) = v$*;*
   - *If* $I(X) = \mathbf{f}$*, then* $s_{\boldsymbol{P}}(I, X) = 1 - w$*;*
   - *If* $I(X) = \mathbf{u}$*, then* $s_{\boldsymbol{P}}(I, X) = w - v$*.* □

Informally, $I(X)$ represents what the possible world I believes about X. The above interval $[v, w]$ produced by Φ_X represents the combined support accorded by the rule set $\boldsymbol{P}_I(X)$ to that belief. $s_{\boldsymbol{P}}(I, X)$ measures the degree by which a truth valuation I is supported by $\boldsymbol{P}(X)$. If X is true in I, it is the combined belief in X supported by $\boldsymbol{P}$ given the truth valuation I. If X is false in I, $s_{\boldsymbol{P}}(I, X)$ is the combined disbelief in X. Otherwise, it represents the combined information gap about X.

It is easy to see that the case of $\boldsymbol{P}_I(X) = \varnothing$ in the above definition is just a special case of $\boldsymbol{P}_I(X) = \{R_1, \ldots, R_n\}$ when $n = 0$, since $\Phi_X(\varnothing)$ is $[0, 1]$ by Definition 3.2.

We now introduce the notion of $\boldsymbol{P}$-support for I *as a whole*. It is defined as a cumulative $\boldsymbol{P}$-support for all atoms in the Herbrand base.

Definition 3.11 *Let* I *be a truth valuation for a blp* $\boldsymbol{P}$*, we define*

$$\hat{m}_{\boldsymbol{P}}(I) = \prod_{X \in B_{\boldsymbol{P}}} s_{\boldsymbol{P}}(I, X)$$

□

Example 3.4 Continue with Example 3.3, suppose we use Φ^{DS} as combination function, $\Phi^{DS}([0.8,1],[0.6,0.8]) \approx [0.9,0.95]$

$$\begin{aligned}
\hat{m}_{P_1}(I_1) &= s_P(I_1,a) \cdot s_P(I_1,b) \approx 0.5 \cdot 0.9 = 0.45 \\
\hat{m}_{P_1}(I_2) &= s_P(I_2,a) \cdot s_P(I_2,b) \approx 0.5 \cdot 0.05 = 0.025 \\
\hat{m}_{P_1}(I_3) &= s_P(I_3,a) \cdot s_P(I_3,b) = 0 \cdot 0.6 = 0 \\
\hat{m}_{P_1}(I_4) &= s_P(I_4,a) \cdot s_P(I_4,b) = 0 \cdot 0.2 = 0 \\
\hat{m}_{P_1}(I_5) &= s_P(I_5,a) \cdot s_P(I_5,b) = 0.5 \cdot 0.6 = 0.3 \\
\hat{m}_{P_1}(I_6) &= s_P(I_6,a) \cdot s_P(I_6,b) \approx 0.5 \cdot 0.05 = 0.025 \\
\hat{m}_{P_1}(I_7) &= s_P(I_7,a) \cdot s_P(I_7,b) = 0.5 \cdot 0.2 = 0.1 \\
\hat{m}_{P_1}(I_8) &= s_P(I_8,a) \cdot s_P(I_8,b) = 0 \cdot 0.2 = 0 \\
\hat{m}_{P_1}(I_9) &= s_P(I_9,a) \cdot s_P(I_9,b) = 0.5 \cdot 0.2 = 0.1
\end{aligned}$$

□

Theorem 3.1 *For any blp* $\boldsymbol{P}$*,*

$$\sum_{I \in \mathcal{TV}al(\boldsymbol{P})} \hat{m}_{\boldsymbol{P}}(I) = 1$$

□

The proof of Theorem 3.1 can be found in Appendix A.1.

Theorem 3.1 shows that $\hat{m}_{\boldsymbol{P}}$ is a support function. This theorem is crucial, as it makes the following definition well-founded.

Definition 3.12 *The* ***model*** *of a blp* $\boldsymbol{P}$ *is the following belief function:*

$$\texttt{model}_{\boldsymbol{P}}(F) = \sum_{I \in \mathcal{TV}al(\boldsymbol{P}) \text{ such that } I \models F} \hat{m}_{\boldsymbol{P}}(I), \quad \text{where } F \in \mathcal{B}ool(B_{\boldsymbol{P}}).$$

□

The belief function $\texttt{model}_{\boldsymbol{P}}(F)$ measures the degree by which F is supported by $\boldsymbol{P}$. It is easy to see that every blp has a unique model. The rationale for the above definition is expressed by the following theorem:

Theorem 3.2 *Let $\boldsymbol{P}$ be a blp and A an atom. For any rule R, let $Body(R)$ denote its body. Let $\mathcal{S}$ be a subset of $\boldsymbol{P}(A)$ that satisfies (i) $\texttt{model}_{\boldsymbol{P}}\big(\bigwedge_{R\in\mathcal{S}} Body(R)\big) > 0$; and (ii) $\mathcal{S}$ is maximal: if $\mathcal{S}' \supseteq \mathcal{S}$ is another subset of $\boldsymbol{P}(A)$ that satisfies (i) then $\mathcal{S}' = \mathcal{S}$. Let $[v_R, w_R]$ denote the belief factor associated with the rule R and suppose $\Phi_A\big(\{[v_R, w_R]\}_{R\in\mathcal{S}}\big) = [v, w]$ ($\Phi_A\big(\{[v_R, w_R]\}_{R\in\mathcal{S}}\big)$ is the result of applying Φ_A to the belief factors in $\mathcal{S}$). Then*

$$\frac{\texttt{model}_{\boldsymbol{P}}\big(A \wedge \bigwedge_{R\in\mathcal{S}} Body(R)\big)}{\texttt{model}_{\boldsymbol{P}}\big(\bigwedge_{R\in\mathcal{S}} Body(R)\big)} = v \qquad \frac{\texttt{model}_{\boldsymbol{P}}\big(\overline{A} \wedge \bigwedge_{R\in\mathcal{S}} Body(R)\big)}{\texttt{model}_{\boldsymbol{P}}\big(\bigwedge_{R\in\mathcal{S}} Body(R)\big)} = 1 - w.$$

□

The above theorem shows that $\texttt{model}_{\boldsymbol{P}}$ is a "correct" (and unique) belief function that embodies the evidence that $\boldsymbol{P}$ provides for each atom. In other words, $\texttt{model}_{\boldsymbol{P}}$ supports each atom in the Herbrand base with precisely the expected amount of support. In contrast, all other works that we are aware of either do not account for combined support provided by multiple rules deriving the same atom or do not have a clear model-theoretic account for that phenomenon.

Theorem 3.2 can be proved by the following lemma.

Lemma 3.3 *Let $\boldsymbol{P}$ be a blp and A an atom. For any rule R, let $Body(R)$ denote its body and $[v_R, w_R]$ denote the belief factor associated with the rule R.*

For any $\mathcal{S} \subseteq \boldsymbol{P}(A)$,

$$\sum_{I \in \mathcal{T}_{\mathcal{S}} \text{ and } I \models A} \hat{m}_{\boldsymbol{P}}(I) = v \times \sum_{I \in \mathcal{T}_{\mathcal{S}}} \hat{m}_{\boldsymbol{P}}(I) \tag{3.1}$$

$$\sum_{I \in \mathcal{T}_{\mathcal{S}} \text{ and } I \models \overline{A}} \hat{m}_{\boldsymbol{P}}(I) = (1-w) \times \sum_{I \in \mathcal{T}_{\mathcal{S}}} \hat{m}_{\boldsymbol{P}}(I) \tag{3.2}$$

where $\mathcal{T}_{\mathcal{S}} = \big\{ I \in \mathcal{TV}al(B_{\boldsymbol{P}}) \mid \forall R \in \mathcal{S},\ I \models Body(R),\ \text{and } \forall R' \in \boldsymbol{P}(A) - \mathcal{S},\ I \not\models Body(R') \big\}$ *and* $[v, w] = \Phi_A\big(\{[v_R, w_R] \mid R \in \mathcal{S}\}\big)$. □

The proof of Lemma 3.3 can be found in Appendix A.2. With the help of Lemma 3.3, the proof of Theorem 3.2 is in Appendix A.3.

Example 3.5 Continue with Example 3.4,

$$\begin{aligned} \texttt{model}_{P_1}(a) &= \hat{m}_{P_1}(I_1) + \hat{m}_{P_1}(I_3) + \hat{m}_{P_1}(I_5) = 0.5 \\ \texttt{model}_{P_1}(\overline{b}) &= \hat{m}_{P_1}(I_2) + \hat{m}_{P_1}(I_4) + \hat{m}_{P_1}(I_7) = 0.125 \\ \texttt{model}_{P_1}(a \wedge b) &= \hat{m}_{P_1}(I_1) = 0.45 \end{aligned}$$

□

3.5 Traffic Advisory Example Revisited

Returning to the example in Section 3.1, suppose that our information sources predict 50% chance of bad weather, parades with 50% certainty, roadwork along the Long Island Expressway (henceforth LIE) with certainty 80%, and police activity due to accidents with the likelihood of 40%. This information is expressed in BLP as follows.

$$\begin{aligned} &[0.8, 0.8]\ \ roadwork(LIE) \\ &[0.5, 0.5]\ \ bad\_weather(LIE) \\ &[0.5, 0.5]\ \ social\_act(LIE) \\ &[0.4, 0.4]\ \ police\_act(LIE) \end{aligned} \tag{3.3}$$

The traffic service fetches the above information from four different information sources and integrates them using these rules: [1]

$$[0.9, 1] \quad delay(?r) \ \texttt{:-} \ roadwork(?r) \wedge bad\_weather(?r)$$
$$[0.8, 1] \quad delay(?r) \ \texttt{:-} \ roadwork(?r) \wedge social\_act(?r)$$
$$[0.99, 1] \quad delay(?r) \ \texttt{:-} \ roadwork(?r) \wedge police\_act(?r)$$

Suppose the atom $delay(?r)$ is associated with the combination function Φ^{max} defined in Section 3.3. When correlation is *not* taken into account, as in [1, 35], the certainty factor of $roadwork(LIE) \wedge bad\_weather(LIE)$ is computed to be $[0.4, 0.4]$, that of $roadwork(LIE) \wedge social\_act(LIE)$ also $[0.4, 0.4]$, and the certainty factor of $roadwork(LIE) \wedge police\_act(LIE)$ is computed to be $[0.32, 0.32]$. The first rule propagates to $delay(LIE)$ support of the degree $[0.36, 1]$; the second rule $[0.32, 1]$; the third rule $[0.3168, 1]$. Applying the combination function Φ^{max}, the certainty factor of $delay(LIE)$ is computed to be $[0.36, 1]$, which means that the available information predicts traffic delay with certainty 0.36 and smooth traffic with certainty 0. Based on this advisory, the students would decide to drive and leave one hour earlier than normal (see Section 3.1 for the explanation of how the students make their travel plans). It is not hard to see that this advisory might cost our students a show. The information from the weather forecast and Department of Transportation alone (the first rule) is enough to predict traffic delays with certainty 0.36. Taking into account the possibilities of parades and accidents, it is reasonable to up the expectation of delays.

In contrast, according to Definition 3.11, BLP computes first $\hat{m}_{\boldsymbol{P}}$, which is a support function for atom set $\{delay(LIE), roadwork(LIE), social\_act(LIE), bad\_weather(LIE), police\_act(LIE)\}$. (We omit the computation details

[1] Note that although the semantics is defined for ground rules, query evaluation algorithms, work with non-ground rules.

here.) Then by Definition 3.12, the belief in traffic delay is computed to be 0.63, which means that our students will shell out a little extra for the train but will make it to the show.

One may argue that the problem was the Φ^{max} combination function and a different function, such as the Dempster's combination rule Φ^{DS} (Section 3.3), may do just fine even without BLP. While this might be true for the traffic conditions in (3.3), a wrong advice will be given in the following scenario:

$$[0.2, 0.2]\ \ roadwork(LIE) \qquad [0.8, 0.8]\ \ bad\_weather(LIE)$$
$$[0.9, 0.9]\ \ social\_act(LIE) \qquad [0.3, 0.3]\ \ police\_act(LIE)$$

where we assume that $delay(?r)$ is associated with the combination function Φ^{DS}. Without taking correlation into account, as in [1, 35], the certainty factor of $roadwork(LIE) \wedge bad\_weather(LIE)$ is computed to be $[0.16, 0.16]$, that of $roadwork(LIE) \wedge social\_act(LIE)$ is computed to be $[0.18, 0.18]$, and that of $roadwork(LIE) \wedge police\_act(LIE)$ is computed to be $[0.06, 0.06]$. The first rule propagates to $delay(LIE)$ support of the degree $[0.144, 1]$; the second rule also $[0.144, 1]$; the third rule $[0.0594, 1]$. Applying the combination function Φ^{DS}, the certainty factor of $delay(LIE)$ is computed to be $[0.31, 1]$, — again suggesting to add one extra hour to the trip. However, this advisory errs on the cautious side. Here all three rules make their predictions building the same roadwork factor into their decision, so this factor is counted multiple times.

In contrast, BLP recognizes that the three predictions are not independent and computes the belief in traffic delay to be 0.18. (Computation details omitted.) Thus, the students will allocate no extra time for the eventualities and will get that badly needed extra hour of sleep before their conference deadlines.

We thus see that, by correlating rules, BLP is able to better predict certainty factors of the combined information.

3.6 Properties and Discussion

In this section, we have some discussions on BLP, including its non-monotonicity, conditions guaranteeing monotonicity, disjunctions in rule body, etc. We will then discuss the reason to choose Dempster-Shafer theory as the formalism to model uncertainty, and also the relationship of BLP to Dempster-Shafer belief functions and defeasible reasoning.

3.6.1 Non-monotonicity

First, generally speaking, the BLP semantics is *non-monotonic*. Consider rules r_1 and r_2:

$$\begin{array}{lll} r_1 & [0.4, 0.4] & X \\ r_2 & [0.8, 0.8] & X \end{array}$$

Let $\boldsymbol{P}_1$ be $\{r_1\}$, $\boldsymbol{P}_2$ be $\{r_2\}$, $\boldsymbol{P}_3$ be $\{r_1, r_2\}$, and $\texttt{bel}_i$ be the model of $\boldsymbol{P}_i$, $i = 1, 2, 3$. For any combination function Φ, let $[v, w] = \Phi([0.4, 0.4], [0.8, 0.8])$, since $v \leq w$, we have: either $v < 0.8$ or $w > 0.4$.

- If $v < 0.8$, then $\texttt{bel}_3(X) < 0.8 = \texttt{bel}_2(X)$. In other words, adding r_1 to $\boldsymbol{P}_2$ reduces the support for X.
- If $w > 0.4$, then $\texttt{bel}_3(\overline{X}) < 0.6 = \texttt{bel}_1(\overline{X})$. In other words, adding r_2 to $\boldsymbol{P}_1$ reduces the support for $\overline{X}$.

[46] discussed the non-monotonicity of Dempster-Shafer theory, which is closely related to the non-monotonicity of BLP.

With certain restrictions on belief factors and combination functions, BLP could become monotonic. One such condition that guarantees monotonicity is:

- Every belief factor is of the form $[v, 1]$; and
- Every combination function takes the form $\Phi([v_1, 1], [v_2, 1]) = [V(v_1, v_2), 1]$, where $V(v_1, v_2) \geq max(v_1, v_2)$.

Indeed, under this restriction, the belief in any negative literal is 0. Let $\boldsymbol{P}' = \boldsymbol{P} \cup \{r\}$, and let the head of r be A, for any truth valuation I, $\boldsymbol{P}'_I(A) = \boldsymbol{P}_I(A) \cup \{r\}$. Let $[v, 1]$ be the result of applying Φ to the belief factors of $\boldsymbol{P}_I(A)$, $[v', 1]$ that of $\boldsymbol{P}'_I(A)$, under the above restriction there must be $v' \geq v$. By Definition 3.10, if $I(A) = \mathbf{t}$, $s_{\boldsymbol{P}'}(I, A) = v', s_{\boldsymbol{P}}(I, A) = v$; if $I(A) = \mathbf{f}$, $s_{\boldsymbol{P}'}(I, A) = s_{\boldsymbol{P}}(I, A) = 0$; if $I(A) = \mathbf{u}$, $s_{\boldsymbol{P}'}(I, A) = 1 - v', s_{\boldsymbol{P}}(I, A) = 1 - v$. For any atom $X \neq A$, $s_{\boldsymbol{P}'}(I, X) = s_{\boldsymbol{P}}(I, X)$. By Definition 3.11 and Definition 3.12, $\texttt{model}_{\boldsymbol{P}'}(A) \geq \texttt{model}_{\boldsymbol{P}}(A)$, and for any atom $X \neq A$, $\texttt{model}_{\boldsymbol{P}'}(A) = \texttt{model}_{\boldsymbol{P}}(A)$.

3.6.2 Disjunction in Rule Body

Under the BLP semantics, the support for A provided by a rule of the form $[v, w]$ A `:-` $B_1 \vee B_2$ might differ from the support for A provided by the pair of rules $[v, w]$ A `:-` B_1 and $[v, w]$ A `:-` B_2, if $\Phi_A([v, w], [v, w]) \neq [v, w]$.

Example 3.6 In blp P_4:

$$\begin{array}{ll} [0.4, 0.9] & a \texttt{ :- } b_1 \\ [0.4, 0.9] & a \texttt{ :- } b_2 \\ [1, 1] & b_1 \\ [1, 1] & b_2 \end{array}$$

Suppose $\Phi_a([0.4, 0.9], [0.4, 0.9]) = [0.61, 0.88]$. We get $\texttt{model}_{P_4}(a) = 0.61$.

While in blp P_5:

$$\begin{array}{ll} [0.4, 0.9] & a \ \texttt{:-}\ b_1 \vee b_2 \\ [1,1] & b_1 \\ [1,1] & b_2 \end{array}$$

$\texttt{model}_{P_5}(a) = 0.4$. □

Nevertheless, every blp can still be transformed into an equivalent blp without body-disjunctions. First, boolean formulas can be transformed to equivalent formulas of the form $C_1 \vee ... \vee C_n$ where C_i, $1 \leq i \leq n$, are conjunctions of literals. Without loss of generality, from now on, by boolean formulas and disjunctions in rule body, we refer to such disjunctions of conjunctions of literals.

Definition 3.13 *For any blp* $\boldsymbol{P}$*, its* ***non-disjunctive equivalent****,* $nodisjunct(\boldsymbol{P})$*, is obtained from* $\boldsymbol{P}$ *by replacing every rule* R *of the form*

$$[v, w]\ H \ \texttt{:-}\ Body_1 \vee \cdots \vee Body_k.$$

where $k > 1$ *and* $Body_i$ $(1 \leq i \leq k)$ *are conjunctions of literals, with a set of rules of the form*

$$\begin{array}{lll} [v,w]\ H & \texttt{:-} & H_R. \\ [1,1]\ H_R & \texttt{:-} & Body_i. \quad 1 \leq i \leq k \end{array}$$

where H_R *is a new predicate that does not appear anywhere else in* $nodisjunct(\boldsymbol{P})$ *and* H_R *has the same arguments (variables) as* H *does.* □

Theorem 3.4 *Let* $\boldsymbol{P}$ *be a blp,* $\texttt{bel}_1$ *be the model of* $\boldsymbol{P}$*, and* $\texttt{bel}_2$ *be the model of* $nodisjunct(\boldsymbol{P})$*. If all combination functions in* $\boldsymbol{P}$ *have the property* $\Phi([1,1],[1,1]) = [1,1]$ *then* $\texttt{bel}_1(F) = \texttt{bel}_2(F)$ *for all* $F \in \mathcal{B}ool(B_{\boldsymbol{P}})$.

In other words, the models of $\boldsymbol{P}$ and $nodisjunct(\boldsymbol{P})$ coincide where it matters.

Proof: By Definition 3.12, we only need to prove that for any $I \in \mathcal{TV}al(\boldsymbol{P})$

$$\hat{m}_{\boldsymbol{P}}(I) = \sum_{I' \in \mathcal{TV}al(nodisjunct(\boldsymbol{P})),\ I' | B_{\boldsymbol{P}} = I} \hat{m}_{nodisjunct(\boldsymbol{P})}(I') \tag{3.4}$$

Let R_i, $1 \leq i \leq n$ be the rules in $\boldsymbol{P}$ with disjunctive bodies, and let $Body_{ij}$, $1 \leq j \leq m$ be the disjuncts in the body of R_i.

Given $I \in \mathcal{TV}al(\boldsymbol{P})$, for any $I' \in \mathcal{TV}al(nodisjunct(\boldsymbol{P}))$ such that $I' \mid B_{\boldsymbol{P}} = I$,

- If $I'(H_{R_i}) \neq max\{I(Body_{ij})\}$ for some $1 \leq i \leq n$, then $s_{nodisjunct(\boldsymbol{P})}(I', H_{R_i}) = 0$ and $\hat{m}_{nodisjunct(\boldsymbol{P})}(I') = 0$;
- If $I'(H_{R_i}) = max\{I(Body_{ij})\}$ for every $1 \leq i \leq n$, then for any atom $A \in B_{\boldsymbol{P}}$, $s_{nodisjunct(\boldsymbol{P})}(I', A) = s_{\boldsymbol{P}}(I, A)$ and $\hat{m}_{nodisjunct(\boldsymbol{P})}(I') = \hat{m}_{\boldsymbol{P}}(I)$.

Since there is exactly one $I' \in \mathcal{TV}al(nodisjunct(\boldsymbol{P}))$ that satisfies 1) $I' \mid B_{\boldsymbol{P}} = I$ and 2) for every $1 \leq i \leq n$, $I'(H_{R_i}) = max\{I(Body_{ij}\}$, we can conclude that for any $I \in \mathcal{TV}al(\boldsymbol{P})$, (3.4) holds. Consequently, by Definition 3.12, the theorem is proved. □

3.6.3 Combination Functions and Belief Factors

One might be wondering whether the combination functions in BLP are really necessary and whether the same result could not be achieved without the use of combination functions. The answer is that combination functions *can* be dispensed with. The main idea is to transform a blp $\boldsymbol{P}$ to another blp $\boldsymbol{P'}$ as follows.

For any atom X, suppose $\boldsymbol{P}(X)$ is the set of rules

$$\begin{array}{llll} [v_1, w_1] & X & \texttt{:-} & Body_1. \\ \ldots\ldots & & & \\ [v_n, w_n] & X & \texttt{:-} & Body_n. \end{array}$$

then let $\boldsymbol{P'}(X)$ be $\{R_S \mid S \subseteq \{1, ..., n\}\}$, where R_S is a rule of the form

$$\Phi(\{[v_k, w_k] \mid k \in S\}) \quad X \quad \texttt{:-} \quad \bigwedge_{i \in S} Body_i \ \wedge \bigwedge_{j \in \{1,\ldots,n\}-S} not\ Body_j$$

However, this requires an extension of BLP with default negation and, more importantly, causes an exponential blowup of the program (making it an unlikely tool for knowledge engineering). Consider the scenario of integrating a number of information sources, each of which has a rule supporting X. With combination functions, we only need to put these rules in a blp, and then directly make inferences on this blp; without combination functions, at the point of integration, we must first conduct an exponential blowup of the rules.

One might also be wondering where the belief factors, i.e. the uncertainty values that are attached to rules and facts, come from. The answer is that they can come from various sources, including data, device, literature, and human experts. For instance, weather forecasts and road work schedule in the traffic advisory example can bring in uncertainty. For another example, sensors have reliabilities, and medical tests have false positive and false negative. In data-rich application domains, large data sets are often utilized to get the uncertainty information.

3.6.4 Probability vs. Belief in Combination of Evidence

Probability theory has been widely used for reasoning with uncertainty. However, several aspects of the application of probability theory to modeling

uncertainty has been criticized [71, 93], especially when it comes to combining evidence obtained from different sources.

To illustrate, consider two mutually exclusive states A and $\overline{A}$. Suppose this distribution is provided by two different sources. Source 1 may assert that $prob(A) = 0.8$, $prob(\overline{A}) = 0.2$, meaning that the probability of A is 0.8. Source 2 may assert that $prob(A) = 0.6$, $prob(\overline{A}) = 0.4$, meaning that the probability of A is 0.6. There is no obvious way to combine information from these two sources because probability is objective. Some approaches take the maximum or the minimum of the two probability values; others take the average. However, none of these has any probabilistic justification.

In some frameworks, e.g. [9, 45], probability intervals are used to model uncertainty. Suppose source 1 asserts $0.8 \leq prob(A) \leq 1$ and source 2 asserts $0.6 \leq prob(A) \leq 0.7$. Some approaches [9] compute the intersection of the two intervals, yielding $\varnothing$ (thus concluding nothing). Some other approaches [45] simply combine the uncertainty ranges, for instance, $[min(0.8, 0.6), max(1, 0.7)]$. Again, no probabilistic justification exists for either of these rules of combination, so probability theory is used here in name but not in substance.

In contrast, Dempster-Shafer theory [10, 71] gives up certain postulates of the probability theory in order to provide an account for the phenomenon of combined evidence.

3.6.5 Relationship to Dempster-Shafer Theory of Evidence

We now relate our semantics to Dempster-Shafer theory.

Definition 3.14 *A **complete valuation** over a set of atoms α is a mapping*

from α *to* $\{\mathbf{t}, \mathbf{f}\}$. *The set of all complete valuations over* α *is denoted as* $\mathcal{U}(\alpha)$. *A* ***complete valuation*** I *for a blp* $\boldsymbol{P}$ *is a complete valuation over* $B_{\boldsymbol{P}}$. *Let* $\mathcal{U}(\boldsymbol{P})$ *denote the set of all the complete valuations for* $\boldsymbol{P}$, *so* $\mathcal{U}(\boldsymbol{P}) = \mathcal{U}(B_{\boldsymbol{P}})$.

□

Complete valuations correspond to interpretations in classical 2-values logic programs. A complete valuation J can be viewed as a state, and it is clear that two different complete valuations represent two mutually exclusive states. $\mathcal{U}(\alpha)$ is a universal set of mutually exclusive states.

A truth valuation I over α (see Definition 3.3) can be uniquely mapped to a set of complete valuations:

$$\Psi_I = \{J \in \mathcal{U}(\alpha) \mid \forall A \in \alpha,\ J(A) = \mathbf{t} \text{ if } I(A) = \mathbf{t}, J(A) = \mathbf{f} \text{ if } I(A) = \mathbf{f}\}$$

In other words, each truth valuation I represents a subset Ψ_I of $\mathcal{U}(\alpha)$, and hence an element Ψ_I of $\mathbb{P}(\mathcal{U}(\alpha))$, the power set of $\mathcal{U}(\alpha)$. Obviously, $\{\Psi_I | I \in \mathcal{TV}al(\alpha)\} \subset \mathbb{P}(\mathcal{U}(\alpha))$.

Dempster-Shafer's mass function, `mass`, is a mapping from $\mathbb{P}(\mathcal{U}(\alpha))$ to $[0, 1]$ such that $\sum_{S \in\ \mathbb{P}(\mathcal{U}(\alpha))} \texttt{mass}(S) = 1$. According to Definition 3.7, our support function m is a mapping from $\mathcal{TV}al(\alpha)$ to $[0, 1]$ such that $\sum_{I \in\ \mathcal{TV}al(\alpha)} m(I) = 1$. Given any support function m, it can be related to a unique mass function, `mass`, as follows.

$$\begin{aligned} &\texttt{mass}(\Psi_I) = m(I), && \forall I \in \mathcal{TV}al(\alpha) \\ &\texttt{mass}(S) = 0, && \text{if } S \neq \Psi_I, \forall I \in \mathcal{TV}al(\alpha) \end{aligned} \tag{3.5}$$

Based on the above correspondence between Dempster-Shafer's mass function and BLP support function introduced in Definition 3.7, we establish a correspondence between Dempster-Shafer's belief function and BLP's belief function of Definition 3.8.

Any BLP formula $F \in \mathcal{B}ool(B_{\boldsymbol{P}})$ (defined in Section 3.4) uniquely corresponds to a set of complete valuations: $\Theta_F = \{J \in \mathcal{U}(B_{\boldsymbol{P}}) \mid J \models F\}$. Clearly, $\Theta_F \in \mathbb{P}(\mathcal{U}(B_{\boldsymbol{P}}))$, and it can be shown that $\{\Theta_F \mid F \in \mathcal{B}ool(B_{\boldsymbol{P}})\} = \mathbb{P}(\mathcal{U}(B_{\boldsymbol{P}}))$.

The following theorem shows that any BLP belief function corresponds to a Dempster-Shafer's belief function.

Theorem 3.5 *Let m be a support function and* `mass` *its corresponding mass function as in (3.5). Also let* `bel` *be the belief function of Definition 3.8 constructed from m, and let* `belief` *be the Dempster-Shafer's belief function based on* `mass`*:* $\texttt{belief}(S) = \sum_{\forall S' \subseteq S} \texttt{mass}(S')$*. Then, for any $F \in \mathcal{B}ool(B_{\boldsymbol{P}})$, the following holds:* $\texttt{bel}(F) = \texttt{belief}(\Theta_F)$.

Proof: Indeed, according to Definition 3.8 and equation (3.5),

$$\texttt{bel}(F) = \sum_{I \in \mathcal{TV}al(\boldsymbol{P}) \text{ such that } I \models F} m(I) = \sum_{I \in \mathcal{TV}al(\boldsymbol{P}) \text{ such that } I \models F} \texttt{mass}(\Psi_I)$$

Because $\Psi_I = \{J \in \mathcal{U}(\alpha) \mid \forall A \in \alpha,\ J(A) = \mathbf{t} \text{ if } I(A) = \mathbf{t}, J(A) = \mathbf{f} \text{ if } I(A) = \mathbf{f}\}$ and $\Theta_F = \{J \in \mathcal{U}(B_{\boldsymbol{P}}) \mid J \models F\}$,

$$\texttt{bel}(F) = \sum_{\forall J \in S' \subseteq \mathcal{U}(B_{\boldsymbol{P}}),\ J \models F} \texttt{mass}(S') = \sum_{\forall S' \subseteq \Theta_F} \texttt{mass}(S') = belief(\Theta_F)$$

□

3.6.6 Relationship to Defeasible Reasoning and Explicit Negation

There is an interesting correspondence between the treatment of contradictory information in BLP and a form of defeasible reasoning called Courteous Logic Programming [23] and, more generally, Logic Programming with Courteous Argumentation Theories (LPDA) [84]. Here we consider only LPDA

without default negation. A defeasible LPDA rule has the form

$$@r \quad H \ \texttt{:-}\ B_1 \wedge \cdots \wedge B_n \tag{3.6}$$

where r is a label, H and B_i, $1 \leq i \leq n$, are atoms or explicitly negation of atoms. As before, we use $\overline{A}$ to represent explicit negation of A.

For any atom A, let $\lambda(A) = [1,1]\,A$ and $\lambda(\overline{A}) = [0,0]\,A$. We extend λ to rules of the form (3.6) so that $\lambda(@r\ H \ \texttt{:-}\ B_1 \wedge \cdots \wedge B_n)$ is $\lambda(H) \ \texttt{:-}\ B_1 \wedge \cdots \wedge B_n$. Finally, λ is extended to programs so that $\lambda(\Pi) = \{\lambda(R) \mid R \in \Pi\}$. Note that $\lambda(\Pi)$ is a blp.

Let $\Pi \models_{LPDA} F$ denote that Π entails F under the semantics of LPDA [84] with one of the courteous argumentation theories and let the combination function Φ_1 be such that $\Phi_1([0,0],[1,1]) = [0,1]$, $\Phi_1([0,0],[0,0]) = [0,0]$, $\Phi_1([1,1],[1,1]) = [1,1]$.

Theorem 3.6 *Let Π be an acyclic LPDA program that consists of the rules of the form (3.6) and let $\texttt{bel}_{\lambda(\Pi)}$ be the model of the blp $\lambda(\Pi)$ with the combination function Φ_1. Assume, in addition, that none of the rules in Π defines or uses the special predicates* `overrides` *and* `opposes`*, which in LPDA provide information about conflicting rules and their defeasibility properties. Then, for any formula $F \in \mathcal{B}ool(B_\Pi)$, $\Pi \models_{LPDA} F$ if and only if $\texttt{bel}_{\lambda(\Pi)}(F) = 1$.* □

The proof can be found in Appendix A.4.

In both theories, the presence of A and $\overline{A}$ (in LPDA) or of $[1,1]\,A$ and $[0,0]\,A$ (in BLP) implies that A's truth value is undefined. That is, inconsistent information is self-defeating. In contrast, Pearce and Wagner's logic programs with strong negation, as defined in [52], handle inconsistent information by explicitly declaring a contradiction. Thus, if Π in Theorem 3.6 is a program with strong negation then $\Pi \models_{LPSN} F$ is equivalent to $\texttt{model}_{\lambda(\Pi)}(F) = 1$ only

if Π is *consistent*. Here $\models_{LPSN}$ denotes the entailment in logic programing with strong negation [52].

3.7 Fixpoint Semantics

Belief functions form the basis for the model-theoretic semantics for blps introduced in Section 3.4. However, such semantics does not provide any effective way of computing the models or answering queries. In this section, we introduce a fixpoint semantics for BLP, which will serve as a basis for the development of a query evaluation algorithm in Chapter 4. As is customary in logic programming, we define the semantics using *ground* (i.e., variable-free) blps. Later this is lifted to the *non-ground* case.

Definition 3.15 *$I_\varnothing$ is defined to be an* empty *truth valuation $\varnothing \longrightarrow \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, i.e., a trivial valuation with an empty domain. Since $I_\varnothing$ is the only truth valuation over $\varnothing$, it follows that $\mathcal{TV}al(\varnothing) = \{I_\varnothing\}$ and $\forall\alpha\forall I \in \mathcal{TV}al(\alpha)\ I\ |_\varnothing = I_\varnothing$.*

We also define a special support function $m_\varnothing : \mathcal{TV}al(\varnothing) \rightarrow [0,1]$ to be $m_\varnothing(I_\varnothing) = 1$; it is the only support function for $\varnothing$. □

We define $\mathcal{SF}(\boldsymbol{P})$ to be the set of all support functions for $\boldsymbol{P}$: $\mathcal{SF}(\boldsymbol{P}) = \{m \mid m \text{ is a support function for } \alpha \in B_{\boldsymbol{P}}\}$. Associated with each blp $\boldsymbol{P}$, there is an operator $\hat{T}_{\boldsymbol{P}}$, which takes a support function in $\mathcal{SF}(\boldsymbol{P})$ as an argument and returns another support function in $\mathcal{SF}(\boldsymbol{P})$.

Definition 3.16 *Let $\boldsymbol{P}$ be a blp, $\hat{T}_{\boldsymbol{P}}$ is a mapping from $\mathcal{SF}(\boldsymbol{P})$ to $\mathcal{SF}(\boldsymbol{P})$, defined as follows. For any support function m in $\mathcal{SF}(\boldsymbol{P})$, let α be the base of m, let*

$$dep(\alpha) = \{X \mid X \in B_{\boldsymbol{P}} - \alpha, \text{ and every atom that } X \text{ depends on is in } \alpha\}.$$

$\hat{T}_{\boldsymbol{P}}(m)$ is defined to be a support function over the set of atoms $\alpha \cup dep(\alpha)$ such that for every $I_1 \in \mathcal{TV}al(\alpha)$ and $I_2 \in \mathcal{TV}al(dep(\alpha))$,

$$\hat{T}_{\boldsymbol{P}}(m)(I_1 \uplus I_2) = m(I_1) \cdot \prod_{X \in dep(\alpha)} Val\Big(\Phi_X\big(BF(X, \boldsymbol{P}, I_1)\big), I_2(X)\Big) \qquad (3.7)$$

where $BF(X, \boldsymbol{P}, I)$ is the multiset of the belief factors of all the rules in $\boldsymbol{P}$, which have atom X in head and whose bodies are true in I, and

$$Val([v, w], \tau) = \begin{cases} v, & \text{if} \quad \tau = \mathbf{t}; \\ 1 - w, & \text{if} \quad \tau = \mathbf{f}; \\ w - v, & \text{if} \quad \tau = \mathbf{u}. \end{cases}$$

In the special case when the base of m is $B_{\boldsymbol{P}}$, we have $\hat{T}_{\boldsymbol{P}}(m) = m$. □

It can be shown that $\hat{T}_{\boldsymbol{P}}(m)$ is a support function because, for every $I_1 \in \mathcal{TV}al(\alpha)$, it is the case that $\sum_{I_2 \in \mathcal{TV}al(dep(\alpha))} \hat{T}_{\boldsymbol{P}}(m)(I_1 \uplus I_2) = m(I_1)$ and, consequently, $\sum_{I_1 \in \mathcal{TV}al(\alpha),\ I_2 \in \mathcal{TV}al(dep(\alpha))} \hat{T}_{\boldsymbol{P}}(m)(I_1 \uplus I_2) = 1$.

If the equation (3.7) looks mysterious, it should not be. All it says is that when we construct $\hat{T}_{\boldsymbol{P}}(m)$ by expanding the base of the support function m from α to $\alpha \cup dep(\alpha)$, we split each support amount $m(I_1)$ according to the constituent components $\hat{T}_{\boldsymbol{P}}(m)(I_1 \uplus I_2)$. Each such component is assigned a fraction specified by the product $\prod_{X \in dep(\alpha)}$ in (3.7). Product is used here because under a *fixed* truth valuation I_1 over α, the rule-sets that fire in I_1 and have *different* heads are treated as independent and each member of the product represents the contribution of each particular head $X \in dep(\alpha)$. In the contribution of a particular head, X, $\Phi_X\big(BF(X, \boldsymbol{P}, I_1)$ is simply the combined belief factor $[v, w]$ of the rules that support X in I_1. Val examines $I_2(X)$ and, depending on whether $I_2(X)$ is true, false, or unknown, interprets the first argument (the combined belief factor $[v, w]$ for X) as the degree of belief in X (i.e., v), or in $\overline{X}$ (i.e., $1 - w$), or as the uncertainty gap for X (i.e., $w - v$).

Now we give a constructive definition of models of blp, based on iterated fixpoint of the operator $\hat{T}_{\boldsymbol{P}}$.

Theorem 3.7 *Let $\boldsymbol{P}$ be a blp. Starting with $m_0 = m_\varnothing$, let m_k be $\hat{T}_{\boldsymbol{P}}^{\uparrow k}(m_0)$, $k = 0, 1, \ldots$. There is an integer $0 \leq \omega \leq |B_{\boldsymbol{P}}|$ such that $m_\omega = m_{\omega+1}$. The support function m_ω coincides with the support function $\hat{m}_{\boldsymbol{P}}$ defined in purely model-theoretic terms in Section 3.4.* □

The proof of Theorem 3.7 can be found in Appendix A.5.

Theorem 3.7 shows that there is always a fixpoint for the $\hat{T}_{\boldsymbol{P}}$ operator. And the fixpoint semantics coincides with the model-theoretic semantics defined in Section 3.4. Consequently we have a variant of Definition 3.12 as follows.

Definition 3.17 *Let $\boldsymbol{P}$ be a blp and m_ω be the fixpoint of $\hat{T}_{\boldsymbol{P}}$ as described in Theorem 3.7. The **model** of $\boldsymbol{P}$ is the following belief function:*

$$\texttt{model}_{\boldsymbol{P}}(F) = \sum_{I \in \mathcal{TV}al(\boldsymbol{P}) \text{ such that } I \models F} m_\omega(I), \quad \text{where } F \in \mathcal{B}ool(B_{\boldsymbol{P}}).$$

□

In fixpoint semantics, starting from $m_\varnothing$, the support function is expanded to a bigger base in each iteration. The bottom-up inferencing algorithm, which we will develop in Chapter 4, is based on the same idea, with the modification that only one atom is added to the support function base each time the support function is expanded.

Chapter 4

Query Evaluation in Belief Logic Programming

The semantics defined in Chapter 3 didn't address the issues of efficient reasoning and query answering, especially for non-ground programs. In this chapter we introduce proof DAGs and present algorithms that utilize proof DAGs for query evaluation. First, we consider only ground blps and queries, then in Section 4.4 we generalize the algorithm to *non-ground* blps and queries.

A **ground query** to a blp $\boldsymbol{P}$ is a statement of the form $?- \ Goal$, where $Goal \in \mathcal{B}ool(B_{\boldsymbol{P}})$. The answer to ground query $?- \ Goal$ should be the belief by $\boldsymbol{P}$ in $Goal$, e.g. $\texttt{model}_{\boldsymbol{P}}(Goal)$ as defined in Definition 3.17.

Given a blp $\boldsymbol{P}$ and a ground query $?- \ Goal$, let $\boldsymbol{P}_g$ be a rule-set composed of all the rules in $\boldsymbol{P}$ plus the additional rule of the form $[1,1] \ g \ \texttt{:-} \ Goal$, where g is a new proposition. It is not hard to see that $\texttt{model}_{\boldsymbol{P}}(Goal)$ is equivalent to $\texttt{model}_{\boldsymbol{P}_g}(g)$. This simple standard trick simplifies the matters by allowing us to assume that *all ground queries are simply singleton atoms.*

Example 4.1 Consider a blp P_6:

$$
\begin{array}{ll}
[0.2, 0.4] & a \ \texttt{:-} \ b \ \wedge \ c \\
[1, 1] & e \ \texttt{:-} \ a \ \wedge \ c \\
[0, 0.5] & e \ \texttt{:-} \ b \ \wedge \ d \\
[0.3, 0.6] & b \\
[1, 1] & c \\
[0.4, 0.6] & d \\
[1, 1] & d \ \texttt{:-} \ h \\
[0.5, 0.5] & f \ \texttt{:-} \ c \ \vee \ e
\end{array}
$$

and a query $? - b \vee e$. To evaluate this query on P_6, we need to compute $\texttt{model}_{\boldsymbol{P}_6}(b \vee e)$. This can be done by computing $\texttt{model}_{\boldsymbol{P}_7}(g)$ where $P_7 = P_6 \cup \{[1, 1] \quad g \ \texttt{:-} \ b \ \vee \ e\}$. □

To simplify the description of algorithms in this chapter, we assume that each rule, R, has a unique identifier, denoted ID_R — a new propositional constant.

4.1 Dependency Graph and Proof Graph

We first introduce *dependency DAGs*, which represent derivation paths through which belief factors are propagated in a blp.

Definition 4.1 *The **dependency DAG**, $\mathcal{H}$, of a ground blp **P** is a directed acyclic bipartite graph whose nodes are partitioned into a set of **atom nodes** (*a-nodes*, for short) and a set of **rule nodes** (*r-nodes*, for short). The nodes and edges are defined as follows:*

- *For each atom A in **P**, $\mathcal{H}$ has an a-node labeled A.*

- *For each rule R in $\boldsymbol{P}$, $\mathcal{H}$ has an r-node labeled with* proposition ID_R.
- *For each rule R in $\boldsymbol{P}$, an edge goes from the r-node labeled ID_R to the a-node labeled with the atom in R's head.*
- *For each rule R in $\boldsymbol{P}$ and each atom A that appears in R's body, an edge goes from the a-node labeled A to the r-node labeled ID_R.*
- *Each edge that goes from an r-node, labeled ID_R, to an a-node is labeled with the belief factor associated with the rule R.* □

Example 4.2 Continuing Example 4.1, Figure 4.1 shows the dependency DAG of blp P_7. $rg, r1, ...r8$ are rule identifiers.
P_7:

rg	$[1,1]$	g	:-	b	$\vee$	e
$r1$	$[0.2,0.4]$	a	:-	b	$\wedge$	c
$r2$	$[1,1]$	e	:-	a	$\wedge$	c
$r3$	$[0,0.5]$	e	:-	b	$\wedge$	d
$r4$	$[0.3,0.6]$	b				
$r5$	$[1,1]$	c				
$r6$	$[0.4,0.6]$	d				
$r7$	$[1,1]$	d	:-	h		
$r8$	$[0.5,0.5]$	f	:-	c	$\vee$	e

□

Proof DAGs are defined next as subgraphs of dependency DAGs, which contain only the atoms and rules that are relevant to particular queries.

Definition 4.2 *Let $\mathcal{H}$ be the dependency DAG of a ground blp $\boldsymbol{P}$. The* ***proof DAG*** *of $\boldsymbol{P}$ for a ground query $? - \ g$, denoted $\mathcal{H}_g$, is a subgraph of $\mathcal{H}$ such that*

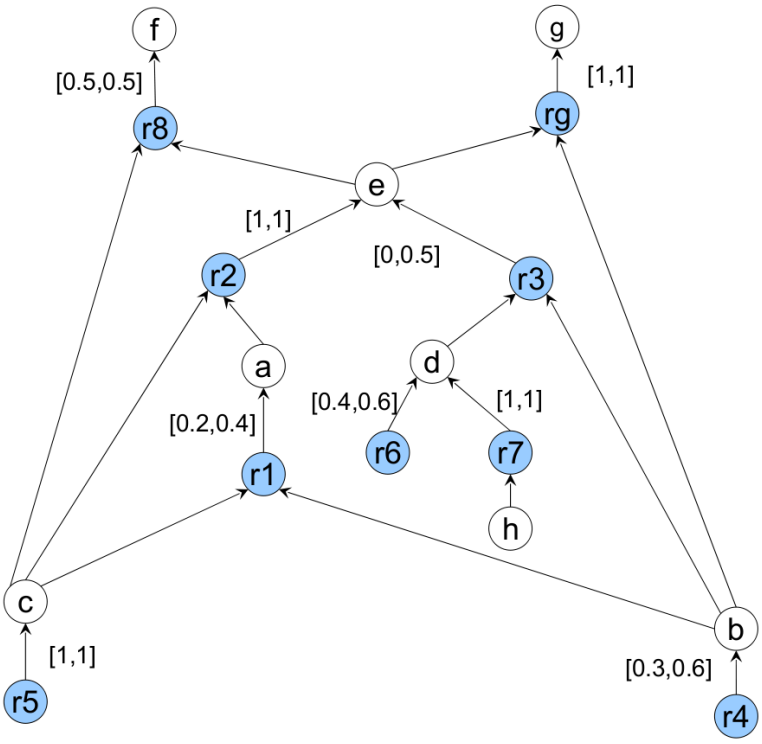


Figure 4.1: The dependency DAG for Example 4.2

- *Every node in $\mathcal{H}_g$ is connected to the a-node labeled g by a directed path; and*
- *$\mathcal{H}_g$ contains all the nodes that are so connected to the a-node labeled g.*

□

In dependency DAGs and proof DAGs, if there is an edge from node A to node B in a DAG, We call A a **child node** of B and B a **parent node** of A.

Example 4.3 Figure 4.1 shows the proof DAG of blp P_7 in Example 4.2 for the query $? - \ g$. □

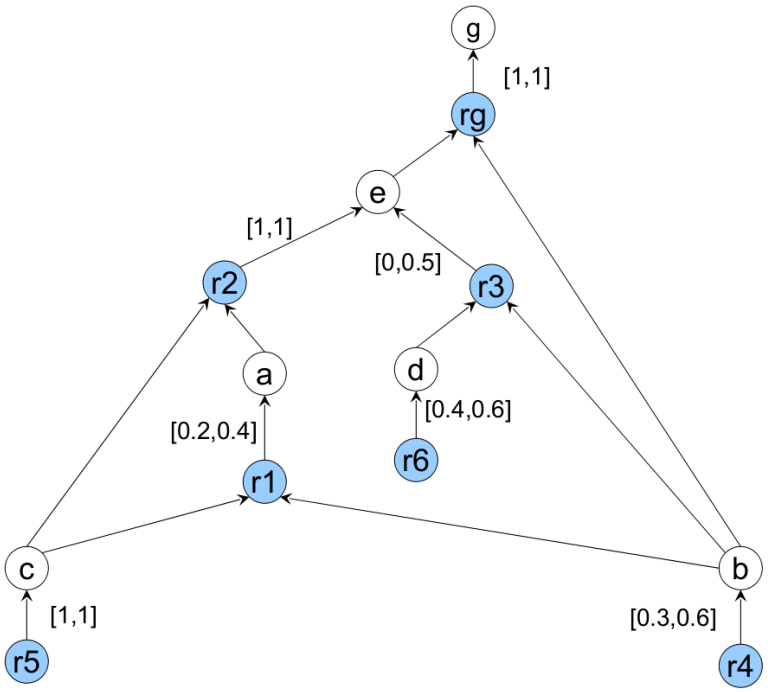


Figure 4.2: The proof DAG for Example 4.3

4.2 Bottom-up Inferencing Algorithm

We start with defining projections and expansions between support functions.

Definition 4.3 *Let m_α and m_β be support functions for the sets of atoms α and β, respectively, where $\alpha \subseteq \beta$. We say m_α is a* ***projection*** *of m_β on α if for every $I \in \mathcal{TV}al(\alpha)$, $m_\alpha(I) = \sum_{J \in \mathcal{TV}al(\beta) \text{ where } J|_\alpha = I} m_\beta(J)$. (Recall that $J|_\alpha$ is defined in Definition 3.5.)*

In the above case we will also say that m_β is an ***expansion*** *of m_α to β.* □

It is clear that for any support function m and any subset α of m's base, there exists one and only one projection of m on α. We denote it by $projection(m, \alpha)$.

The projection relation is transitive, as shown in the following lemma.

Lemma 4.1 *Consider a support function m_α for a set of atoms α, a support function m_β for a set of atoms β and a support function m_γ for a set of atoms γ, where $\alpha \subseteq \beta \subseteq \gamma$. If m_α is a projection of m_β on α and m_β is a projection of m_γ on β then m_α is a projection of m_γ on α.*

Proof: By Definition 4.3,

$$\begin{aligned} m_\alpha(I) &= \sum_{\substack{I' \in \mathcal{TV}al(\beta) \\ \text{where } I'|_\alpha = I}} m_\beta(I') = \sum_{\substack{I' \in \mathcal{TV}al(\beta) \\ \text{where } I'|_\alpha = I}} \sum_{\substack{I'' \in \mathcal{TV}al(\gamma) \\ \text{where } I''|_\beta = I'}} m_\gamma(I'') \\ &= \sum_{\substack{I'' \in \mathcal{TV}al(\gamma) \\ \text{where } (I''|_\beta)|_\alpha = I}} m_\gamma(I'') \end{aligned}$$

From Definition 3.5 we know that $(I''\ |_\beta)\ |_\alpha = I''\ |_\alpha$. Thus, according to Definition 4.3, m_α is a projection of m_γ on α. □

Lemma 4.2 *Let m_α be a support function for α and m_β on β, where $\alpha \subseteq \beta$, and let m_α be a projection of m_β on α. Let bel_α be the belief function constructed from m_α as in Definition 3.8, and bel_β be the belief function constructed from m_β. Then, for any formula $F \in \mathcal{B}ool(\alpha)$, we have $bel_\alpha(F) = bel_\beta(F)$.* □

The proof of Lemma 4.2 follows from Definition 3.8 and Definition 4.3.

4.2.1 Algorithm

Now we introduce an algorithm, Algorithm 1, to compute $\hat{m}_{\boldsymbol{P}}$ for blp $\boldsymbol{P}$. The algorithm is based on the same idea as the fixpoint semantics in Section 3.7. For the easiness of expression, and for the later optimization to Algorithm 2, instead of expanding the support function m from the base α

to $\alpha \cup dep(\alpha)$ in the fixpoint semantics, Algorithm 1 post-order traverses the dependency DAG, and at each step expands m from the base $Base$ to a larger base $Base \cup \{X\}$. Note that each r-node ID_R represents the statement that the body of rule R is true and is used to facilitate the computation.

Algorithm 1

(∗ Bottom-up inferencing ∗)

Input: A ground blp $\boldsymbol{P}$.

Output: $\hat{m}_{\boldsymbol{P}}$ as defined in Definition 3.11.

1. Construct the dependency DAG, $\mathcal{H}$, of $\boldsymbol{P}$
2. Let $< X_1, ..., X_n >$ be a post-order traversal of $\mathcal{H}$.
3. $m \leftarrow m_{\varnothing}$
4. $Visited \leftarrow \varnothing$
5. $Base \leftarrow \varnothing$
6. **for** $i \leftarrow 1$ **to** n **do**
7. **if** X is an a-node labeled with an atom A
8. **then** Initialize a new support function m' for $Base \cup \{A\}$.
9. Let $Id_1, \ldots, Id_k$ be the labels of the child r-nodes of X.
10. **forall** $I \in \mathcal{TV}al(Base)$ such that $m(I) \neq 0$ **do**
11. $\mathcal{S} \leftarrow \varnothing$
12. **for** $j \leftarrow 1$ **to** k **do**
13. **if** $I(Id_j) = \mathbf{t}$
14. **then** Let $[V_j, W_j]$ be the belief factor of the rule identified by Id_j.
15. $\mathcal{S} \leftarrow \mathcal{S} \cup \{[V_j, W_j]\}$
16. $[V, W] \leftarrow \Phi_A(\mathcal{S})$
17. $m'(I \uplus \{A \rightarrow \mathbf{t}\}) \leftarrow m(I) \cdot V$
18. $m'(I \uplus \{A \rightarrow \mathbf{f}\}) \leftarrow m(I) \cdot (1 - W)$
19. $m'(I \uplus \{A \rightarrow \mathbf{u}\}) \leftarrow m(I) \cdot (W - V)$

20. $m \leftarrow m'$
21. $Visited \leftarrow Visited \cup \{A\}$
22. $Base \leftarrow Base \cup \{A\}$
23. **else** ($*$ X is an r-node labeled with Id which is the identifier of rule R. $*$)
Initialize a new support function m' for $Base \cup \{Id\}$.
24. Let $A_1, \ldots, A_k$ be the labels of the child a-nodes of X.
25. Let F be the body of R.
26. **forall** $I \in \mathcal{TV}al(Base)$ such that $m(I) \neq 0$ **do**
27. **forall** $\tau \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ **do**
28. **if** $\tau = I(F)$
29. **then** $m'(I \uplus \{Id \rightarrow \tau\}) \leftarrow m(I)$
30. $m \leftarrow m'$
31. $Visited \leftarrow Visited \cup \{Id\}$
32. $Base \leftarrow Base \cup \{Id\}$
33. $m \leftarrow projection(m, B_{\boldsymbol{P}})$
34. **return** m

Now we prove that this algorithm indeed outputs $\hat{m}_{\boldsymbol{P}}$, in other words, this algorithm is consistent with the declarative semantics defined in Section 3.4.

Theorem 4.3 (Correctness of Algorithm 1) *Given a ground blp* $\boldsymbol{P}$*, let* $\hat{m}_{\boldsymbol{P}}$ *be the support function for* $\boldsymbol{P}$ *as defined in Definition 3.11 and let* m *be the outcome of Algorithm 1. The algorithm is correct in the sense that* $m = \hat{m}_{\boldsymbol{P}}$*.* □

To prove this theorem, we only need to prove that the support function we get at line 40 of Algorithm 1 is an expansion of $\hat{m}_{\boldsymbol{P}}$. The correctness of Algorithm 1 then follows from Lemma 4.1 and Lemma 4.2. The proof of Theorem 4.3 can be found in Appendix A.6.

4.2.2 Complexity

The space complexity for storing the support function of interest is bounded by the domain size of the final and largest support function, which is $O(3^{|B_P|})$.

For the time complexity, consider the operator $\hat{T}_{\boldsymbol{P}}$ from Definition 3.16. During the execution of the algorithm, each rule is processed exactly once by the operator $\hat{T}_{\boldsymbol{P}}$, and the computation is dominated by evaluating the rule body under each truth valuation over the base of the current support function. Therefore, the time complexity of processing a rule R is bounded by $O(|Body(R)| \cdot 3^{|B_P|})$, where $|Body(R)|$ denotes the size of the body of R, and the overall complexity of the algorithm is $O(\sum_{R \in \boldsymbol{P}} |Body(R)| \cdot 3^{|B_P|})$.

The computational complexity is high because this algorithm computes the entire model of a blp and stores the support function for the entire Herbrand base $B_{\boldsymbol{P}}$ regardless of the query. In the next section we will introduce a much more effective algorithm that answers queries for blps.

4.3 Query Evaluation for Ground Cases

In this section we introduce an algorithm that computes the answer to the query $? - \ g$, i.e., $\texttt{model}_{\boldsymbol{P}}(g)$, the degree of belief in g.

4.3.1 Algorithm

Like Algorithm 1, Algorithm 2 is based on the idea of propagating belief functions during post-order traversing. However, since we only care about the belief in g instead of the support function on the whole Herbrand base, a great amount of redundant information can be eliminated. First, instead of traversing the dependency DAG, we only need to traverse the proof DAG,

which contains only the atoms and rules that are relevant to the query. Second, when traversing every node X, after expanding m from the base $Base$ to a larger base $Base \cup \{X\}$, Algorithm 2 projects m to a smaller base by throwing out the nodes which are no longer needed for the rest of the computation.

- As shown in lines 19-22. If visiting an a-node labeled with an atom A, let $Id_1, \ldots, Id_k$ be the child r-nodes of A. At this point m is expanded from the base $Base$ to $Base \cup \{A\}$. Since no non-visited node depends on the r-nodes labeled $Id_1, \ldots, Id_k$, we can then project m from the base $Base \cup \{A\}$ to $Base \cup \{A\} - \{Id_1, \ldots, Id_k\}$.

- As shown in lines 30-37. If visiting an r-node labeled with a rule Id Id, let $A_1, \ldots, A_k$ be the labels of that r-node's children (which are a-nodes). At this point m is expanded from the base $Base$ to $Base \cup \{Id\}$. Let $\{A_{h1}, \ldots, A_{hl}\}$ be those a-nodes in $\{A_1, \ldots, A_k\}$ such that no non-visited node in the DAG depends on them. We can then project m from the base $Base \cup \{Id\}$ to $Base \cup \{Id\} - \{A_{h1}, \ldots, A_{hl}\}$.

Algorithm 2

(∗ Query evaluation ∗)

Input: A ground blp $\boldsymbol{P}$, a ground query $? - \ g$, and the proof DAG, $\mathcal{H}$, of $\boldsymbol{P}$ for the query $? - \ g$.

Output: The degree of belief in g, i.e., $\texttt{model}_{\boldsymbol{P}}(g)$ as defined in Definition 3.12.

1. Let $X_1, ..., X_n$ be a post-order traversal of $\mathcal{H}$.
2. $m \leftarrow m_{\varnothing}$
3. $Visited \leftarrow \varnothing$
4. $Base \leftarrow \varnothing$
5. **for** $i \leftarrow 1$ to n
6. **if** X is an a-node labeled with an atom A
7. **then** Initialize a new support function m' for $Base \cup \{A\}$.

8. Let $Id_1, \ldots, Id_k$ be the labels of the child r-nodes of X.
9. **forall** $I \in \mathcal{TV}al(Base)$ such that $m(I) \neq 0$
10. $\mathcal{S} \leftarrow \varnothing$
11. **for** $j \leftarrow 1$ to k
12. **if** $I(Id_j) = \mathbf{t}$
13. **then** Let $[V_j, W_j]$ be the belief factor of the rule identified by Id_j.
14. $\mathcal{S} \leftarrow \mathcal{S} \cup \{[V_j, W_j]\}$
15. $[V, W] \leftarrow \Phi_A(\mathcal{S})$
16. $m'(I \uplus \{A \rightarrow \mathbf{t}\}) \leftarrow m(I) \cdot V$
17. $m'(I \uplus \{A \rightarrow \mathbf{f}\}) \leftarrow m(I) \cdot (1 - W)$
18. $m'(I \uplus \{A \rightarrow \mathbf{u}\}) \leftarrow m(I) \cdot (W - V)$
19. $m' \leftarrow projection(m', Base \cup \{A\} - \{Id_1, \ldots, Id_k\})$
20. $m \leftarrow m'$
21. $Visited \leftarrow Visited \cup \{A\}$
22. $Base \leftarrow Base \cup \{A\} - \{Id_1, \ldots, Id_k\}$
23. **else** ($*$ X is an r-node labeled with Id which is the identifier of rule R. $*$) Initialize a new support function m' for $Base \cup \{Id\}$.
24. Let $A_1, \ldots, A_k$ be the labels of the child a-nodes of X.
25. Let F be the body of R.
26. **forall** $I \in \mathcal{TV}al(Base)$ such that $m(I) \neq 0$
27. **forall** $\tau \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$
28. **if** $\tau = I(F)$
29. **then** $m'(I \uplus \{Id \rightarrow \tau\}) \leftarrow m(I)$
30. $\alpha \leftarrow \varnothing$
31. **for** $i = 1$ to k
32. **if** every parent node of A_i is in $Visited$
33. **then** $\alpha \leftarrow \alpha \cup \{A_i\}$
34. $m' \leftarrow projection(m', Base \cup \{Id\} - \alpha)$

35. $m \leftarrow m'$

36. $Visited \leftarrow Visited \cup \{Id\}$

37. $Base \leftarrow Base \cup \{Id\} - \alpha$

38. **return** $\sum_{I \in \mathcal{TV}al(Base), I(g)=\mathbf{t}} m(I)$

Example 4.4 Algorithm 2 does not specify any concrete post-order traversal to use. One possible order of traversal for the proof DAG in Figure 4.1 is: $\langle r5,\ c,\ r4,\ b,\ r1,\ a,\ r2,\ r6,\ d,\ r3,\ e,\ rg,\ g\rangle$. With this traversal, the base of the support function m changes during the runtime of Algorithm 2 in the following order: $\{r5\}$, $\{c\}$, $\{c, r4\}$, $\{c, b\}$, $\{c, b, r1\}$, $\{c, b, a\}$, $\{b, r2\}$, $\{b, r2, r6\}$, $\{b, r2, d\}$, $\{b, r2, r3\}$, $\{b, e\}$, $\{rg\}$, $\{g\}$. Suppose $\Phi_e = \Phi^{DS}$, the output of Algorithm 2 is $\texttt{model}_{P_7}(g) = 0.3$. □

After introducing the query evaluation algorithm, we need to prove that this algorithm indeed outputs $\texttt{model}_{\boldsymbol{P}}(g)$, in other words, this algorithm is consistent with the semantics defined in Section 3.4.

Theorem 4.4 (Correctness of Algorithm 2) *Given a ground blp* $\boldsymbol{P}$ *and a ground query* $?-\ g$*, let* $\texttt{model}_{\boldsymbol{P}}$ *be the model of* $\boldsymbol{P}$ *and* $\texttt{bel}(g)$ *be the outcome of Algorithm 2. The algorithm is correct in the sense that* $\texttt{bel}(g) = \texttt{model}_{\boldsymbol{P}}(g)$.

□

The proof of Theorem 4.4 can be found in Appendix A.7.

4.3.2 Complexity

Recall that the complexity of Algorithm 1 is $O(\sum_{R \in \boldsymbol{P}} |Body(R)| \cdot 3^{|B_P|})$. The complexity of Algorithm 2 is much lower. First of all, the space complexity is bounded by the domain size of the largest support function. Let M be the maximum among the sizes of support function bases constructed during the

run of the algorithm. Then space complexity is $O(3^M)$. The time needed to construct the proof DAG $\mathcal{H}$ for a query is the same as the time needed to find all the proof paths in classical deductive databases (with certainty factors removed): it is polynomial in the size of the data. Then the algorithm performs a post-order traversal of $\mathcal{H}$. At each step, when visiting a node, X, it tries to expand the support function $m_{\mathcal{S}}$ from the base $\mathcal{S}$ to $\mathcal{S} \cup \{X\}$. By the same argument as in the previous paragraph, the time complexity for processing each rule R is bounded by $O(|Body(R)| \cdot 3^M)$. Therefore, since each rule is processed by exactly one step of the algorithm, the overall time complexity is bound by $O(\sum_{R \in \mathcal{T}} |Body(R)| \cdot 3^M)$, where $\mathcal{T}$ is the set of rules that appear in $\mathcal{H}$, and M is the maximum among the sizes of support function bases constructed during the run of the algorithm.

It is clear from the query evaluation algorithm that, compared to Algorithm 1, it requires less space in a typical run by "garbage-collecting" atoms in the base of the support function. It is less clear where the time savings come from. The two time complexity bounds estimates above indicate that the exponent has dominant effect. For instance, in Example 4.4, the Algorithm 1 computation yields the exponent $|B_{\boldsymbol{P}}| = 6$. In contrast, the exponent for the query evaluation algorithm is $M = 3$.

It is interesting to note that different traversals of the proof DAG might generate support function bases whose maximum sizes vary greatly. An important issue is, therefore, finding heuristics that produce smaller bases.

4.4 Query Evaluation for Non-Ground Cases

Now we turn to non-ground programs and non-ground queries.

A **non-ground query** to a blp $\boldsymbol{P}$ is a statement of the form $? - \ Goal$,

where *Goal* is a Boolean combination of non-ground atoms. A ground instance *goal* of *Goal* is called a **positive instance** of *Goal* if *goal* is entailed by $\boldsymbol{P}$ with a positive belief. An **answer to non-ground query** $? -$ *Goal* is a positive instance *goal* of *Goal*, together with $\texttt{model}_{\boldsymbol{P}}(goal)$. Using the same trick as for ground queries, we can make the assumption that all non-ground queries are simply singleton non-ground atoms.

The main problem in answering queries to non-ground blps is how to construct proof DAGs without grounding the whole program and constructing the whole dependency DAG. It turns out that a procedure in the style of SLD-resolution [34] can solve this problem and even construct a *pruned* proof DAG which does not contain atom that is not supported at all or rules that never fires.

We first define pruned proof DAGs.

Definition 4.4 *The* ***pruned proof DAG*** *of a ground blp* $\boldsymbol{P}$ *for a ground query* $? -$ *g is obtained from the proof DAG of* $\boldsymbol{P}$ *for* $? -$ *g by the following steps:*

- *Mark all leaf a-nodes as "failed" and all leaf r-nodes as "non-failed".*
- *Perform a post-order traversal to mark every node as follows:*
  - *If an a-node has at least one child r-node which is marked "non-failed," mark the a-node as "non-failed;" otherwise, mark the a-node as "failed."*
  - *If an r-node has at least one child a-node which is marked "failed," mark the r-node as "failed;" otherwise, mark the r-node as "non-failed."*
- *Delete all the nodes which are marked as "failed."*

- *Delete all the nodes which are not connected to the a-node g.* □

It is clear that pruned proof DAG contains all the necessary information to answer $? - \ g$, and executing Algorithm 2 on pruned proof DAGs eliminates unnecessary computation compared to running it without pruning the DAGs.

Recall that SLD-resolutions can only be applied to programs without body-disjunctions, while blps might have such disjunctions. Fortunately, every blp can be transformed into an equivalent blp without body-disjunctions, as defined in Definition 3.13. In this section we assume that blps do not have disjunctions in rule bodies.

Now we define SLD-trees on *non-ground* programs and queries in a constructive way. We will use SLD-trees in Algorithm 3 to build the pruned proof DAGs.

Definition 4.5 ***[Constructive definition of SLD-tree in BLP]*** *An* ***SLD-tree*** *for a blp* $\boldsymbol{P}$ *and a query* $? - \ G$ *(both* $\boldsymbol{P}$ *and* G *can be non-ground) is constructed by the following steps:*

- *Add a node labeled with* $? - \ G$ *as root.*
- *Repeat the following until no nodes can be added:*
  Let o *be a leaf node labeled with* $? - \ G_1 \wedge ... \wedge G_n$,
  for every rule R *in* $\boldsymbol{P}$ *of the form* $[v, w]$ A `:-` $B_1 \wedge ... \wedge B_m$ *such that* G_1 *and* A *or* G_1 *and* $\overline{A}$ *unify with the most general unifier (mgu)* θ,
  - *Add a node* o' *labeled with* $? - \ (B_1 \wedge ... \wedge B_m \wedge G_2 \wedge ... \wedge G_n)\theta$;
  - *Add an edge from* o *to* o', *then label this edge with a triple* $\langle \theta, ID_R, Vars\theta \rangle$, *where* ID_R *is the identifier of* R *and* $Vars$ *is the list of variables in* R.

In a special case when the node o above is labeled with $? - G_1$ *(has only one atom) and R above is a fact that unifies with* G_1 *or* $\overline{G_1}$*, o′ becomes a leaf node labeled with* $? - \{\}$*. We call it a* success node.

- *Delete the nodes and edges that are not connected to any success node.*

□

Using the SLD-tree for $? - G$, Algorithm 3 constructs the set of pruned proof DAGs for the ground form of $\boldsymbol{P}$ and each query $? - g$, where g is a positive instance of G. The query $? - G$ can then be answered by applying Algorithm 2 on the output of Algorithm 3.

Algorithm 3

(∗ Construct pruned proof DAGs from SLD-tree ∗)

Input: An SLD-tree, $\mathcal{T}$, of $\boldsymbol{P}$ and of the query $? - G$.

Output: The set of pruned proof DAGs for the grounding of $\boldsymbol{P}$ and for each query $? - g$, where g is a positive instance of G

1. $\mathcal{HS} \leftarrow \varnothing$
2. $\Theta \leftarrow \varnothing$
3. **forall** path Γ in $\mathcal{T}$ from root to a leaf node labeled $? - \{\}$ **do**
4. Let θ be the composition of all the mgu's labeled along Γ.
5. $\Theta \leftarrow \Theta \cup \{\theta\}$
6. **forall** edge e along Γ **do**
7. Let ID_R be the rule identifier labeled on e and let R be the corresponding rule.
8. Let $Vars$ be the variable list labeled on e.
9. $Insert(\mathcal{HS}$, an r-node labeled with $ID_R(Vars\theta))$ (∗ $Insert$ means insert a node or an edge if it is not already in the graph ∗)
10. $Insert(\mathcal{HS}$, an a-node labeled with $head(R)\theta)$
11. $Insert(\mathcal{HS}, edge(ID_R(Vars\theta), head(R)\theta))$

12. Label the new edge with the belief factor of R.
13. **forall** atom A in the body of R **do**
14. $Insert(\mathcal{HS}$, an a-node labeled with $A\theta)$
15. $Insert(\mathcal{HS}, edge(A\theta, ID_R(Vars\theta)))$
16. $Set \leftarrow \{\mathcal{H} \mid \exists\theta \in \Theta,\ \mathcal{H}$ is a subgraph of $\mathcal{HS}$ which contains all the nodes that are connected to a node labeled with $g\theta\}$
17. **return** Set

Example 4.5 Consider a system that assesses the risk for people getting a certain disease. The relevant inference rules and facts can be expressed as follows:

$r1$	$[0.2, 1]$	$disease(X)$:- $little\_sports(X)$.
$r2$	$[0.1, 1]$	$disease(X)$:- $favorite(X, Y), unhealthy(Y)$.
$r3$	$[1, 1]$	$little\_sports(p1)$.
$r4$	$[1, 1]$	$favorite(p1, a)$.
$r5$	$[1, 1]$	$favorite(p1, b)$.
$r6$	$[1, 1]$	$favorite(p1, c)$.
$r7$	$[1, 1]$	$favorite(p2, a)$.
$r8$	$[1, 1]$	$favorite(p2, d)$.
$r9$	$[1, 1]$	$favorite(p2, e)$.
$r10$	$[1, 1]$	$favorite(p3, b)$.
$r11$	$[1, 1]$	$favorite(p3, f)$.
$r12$	$[1, 1]$	$unhealthy(a)$.
$r13$	$[1, 1]$	$unhealthy(d)$.

Figure 4.4 shows an SLD-tree for this program and the query $?-\ disease(X)$. The pruned proof DAGs are shown in Figure 4.4. Suppose $\Phi_{disease} = \Phi^{DS}$. Then the result is $\{\langle disease(p1), \texttt{belief} : 0.28\rangle, \langle disease(p2), \texttt{belief} : 0.19\rangle\}$.

□

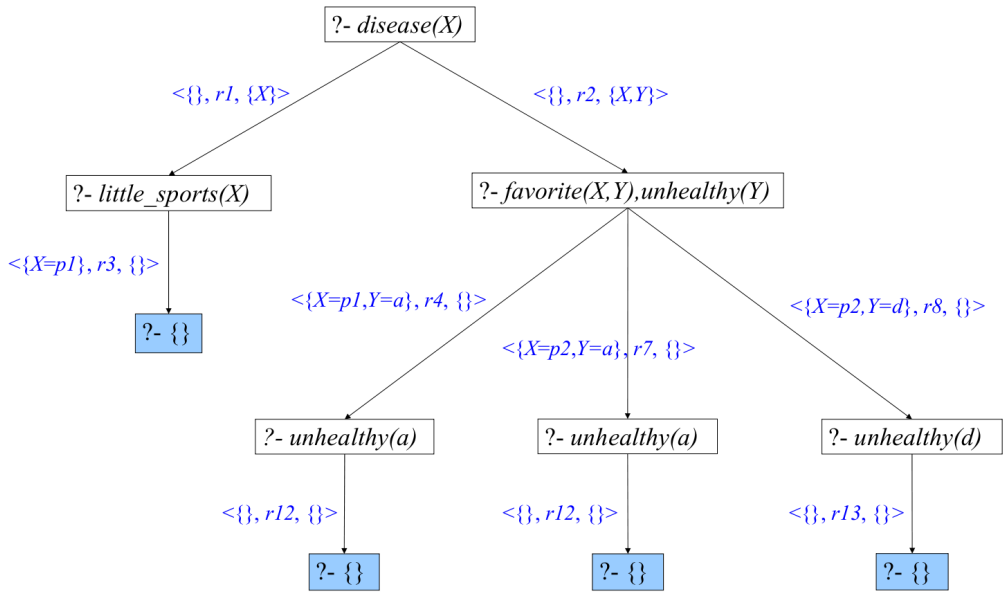


Figure 4.3: SLD-tree for Example 4.5

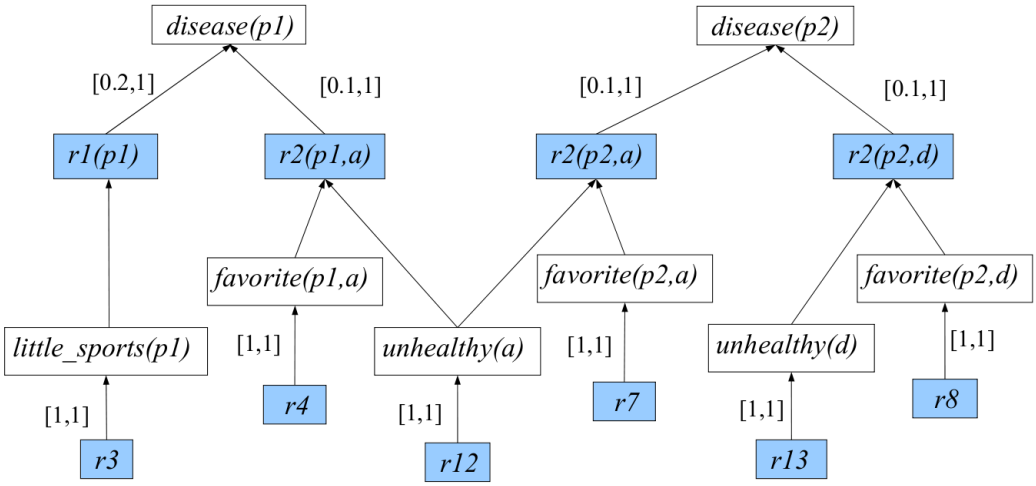


Figure 4.4: Pruned proof DAGs for Example 4.5

Chapter 5

Belief Logic Programs with Cycles

Like other quantitative logic programming frameworks, BLP faces the same challenge from recursive loops, and the semantics defined in Chapter 3 was restricted to belief logic programs without cyclic dependency among atoms. In this chapter we extend the previous work to belief logic programs with cycles by adopting a novel approach: instead of introducing time parameters to eliminate loops, we analyze the program structure and discard the support from unwanted loop influence. We define a transformational semantics and a fixpoint semantics in which self-supported beliefs are discarded. We also show that the proposed semantics are reasonable and are backward compatible with the semantics defined in Chapter 3.

5.1 Background

Let us review how prior quantitative logic programming frameworks deal with recursive loops.

Based on the approaches of uncertainty deduction, Lakshmanan and Shiri [35] classified the proposed quantitative logic programming frameworks into *annotation-based* and *implication-based* as follows:

- In the annotation-based frameworks such as [9, 30, 31, 48, 45, 49, 78], a rule is of the form $A : f(\beta_1, ..., \beta_n)$:- $B_1 : \beta_1 \wedge ... \wedge B_n : \beta_n$ which asserts "the certainty of A is at least (or is in) $f(\beta_1, ..., \beta_n)$, whenever the certainty of B_i is at least (or is in) β_i, $1 \leq i \leq n$."

- In the implication-based frameworks such as [2, 35, 29, 38, 39, 65, 73, 17], a rule is of the form $\beta :$ A :- $B_1 \wedge ... \wedge B_n$ which asserts " the certainty that $B_1 \wedge ... \wedge B_n$ implies A is at least (or is in) β."

In the annotation-based frameworks, when function f in every rule is a constant function, the certainty of the rule head does not depend on the certainty of atoms in the rule body. Recursive loops are not a problem in such cases. When function f is not a constant function, such as in [45], the certainty of the rule head depends on the certainty of atoms in the rule body. Recursive loops are looked on as feedback connections and may cause infinite feedback.

Things are different when we look at implication-based frameworks. In some frameworks, such as [39, 35], certainty values are assigned to atoms, rules are treated as constraints, and models of a program satisfy all the constraints in the program. Recursive loops do not present a problem in such cases. In other frameworks, such as [29, 63, 73], certainty values are assigned to possible worlds and certainty of the head of a rule cannot be computed until certainty of atoms in the rule body is established. Consequently recursive loops cause a problem: it becomes impossible to compute certainty of the atoms involved in a recursive loop. Some frameworks in this category [29, 63] simply do not allow programs with recursive loops. Some others, such as [22, 73], eliminate

recursive loops by introducing time parameters into atoms involved in loops, either explicitly or implicitly.

5.2 Motivating Example: Diagnosis

Suppose that the rate of false positive test results for a certain disease is 20%. Furthermore, suppose that the certainty that someone who had a contact with a contagious person will also contract that same disease is 60%.

Now, let us assume that the tests for two persons, p_1 and p_2, came back positive, but there is no evidence that p_1 and p_2 had a contact. An expert system might then diagnose both p_1 and p_2 as having contracted the disease with certainty 80%. Now, suppose that the test for two other persons, p_3 and p_4, came back positive and p_3 and p_4 are *known* to have had a contact. Common sense then suggests that p_3 and p_4 are more likely to have the disease compared with p_1 and p_2.

The support (or dependence) relation with regard to p_3 and p_4 is shown in Figure 5.1. We can see that there is a loop (a cyclic dependency) between "p_3 has disease" and "p_4 has disease".

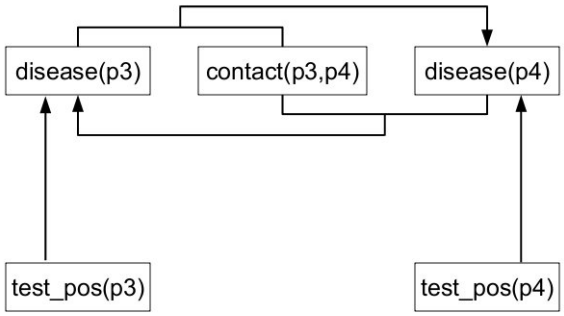


Figure 5.1: Support relation w.r.t. p_3 and p_4 in disease example

Due to the cyclic dependency, the belief in "p_3 has disease" pumps up the certainty of "p_4 has disease" and vice versa. In fact, this dependency is a self-supporting feedback loop, and if we keep combining evidence produced by this feedback loop, the belief in p_3 and p_4's diagnoses will end up close to 1. Clearly such inference is undesired. The problem, therefore, is: how can we discard loop influence when combining all the supporting evidence?

Our method, described in Section 5.4, identifies and discards such self-supporting feedback. After presenting the method, we will revisit the above example in Section 5.7 and show that the new method produces inference that is in accord with intuition.

5.3 Syntax of General BLP

A **belief logic program** (or a *blp*, for short) is a set of annotated rules. Each **annotated rule** has the following format:

$$[v, w]\ X\ \texttt{:-}\ Body$$

where X is a positive atom and $Body$ is a conjunction of literals, i.e., a conjunction of atoms and negation of atoms. [1]

A blp $\boldsymbol{P}$ is said to be **cyclic** if there is an atom that depends on itself. Otherwise $\boldsymbol{P}$ is said to be **acyclic**. In Section 3.2, we required that in a blp there can be no circular dependency among atoms, i.e., till now we only considered acyclic blps. This is a serious limitation of express power. In this chapter we remove this restriction and allow circular dependency.

We introduce some necessary notions first.

[1] In the BLP syntax and semantics in Section 3.2 and 3.4, rule bodies are Boolean combinations of literals. Since every blp can be transformed into an equivalent blp without body-disjunctions, as shown in Section 4.1, in this chapter we assume there is no disjunction in rule bodies.

Definition 5.1 *Let* $\boldsymbol{P}$ *be a blp.* $B_{\boldsymbol{P}}$ *can always be partitioned into disjoint atom sets* $\mathcal{C}_1, \ldots, \mathcal{C}_k$*, such that two atoms are in the same set if and only if they depend on each other. We call* $\mathcal{C}_1, \ldots, \mathcal{C}_k$ *the* ***atom cliques*** *of* $\boldsymbol{P}$ *and use* $clique(A)$ *to denote the atom clique that contains atom* A*.*

A ***clique ordering*** *for* $\boldsymbol{P}$ *is a bijective function,* $Order : \{\mathcal{C}_1, \ldots, \mathcal{C}_k\} \rightarrow \{1, \ldots, k\}$*, such that, for any pair of atoms* X *and* Y*,* X *does not depend on* Y *if* $order(clique(X)) < order(clique(Y))$*.* □

We can safely infer that every atom clique in an acyclic blp has size 1, but not vice versa. Actually a blp can be cyclic even if there is only one atom in it – that atom, A, can depend on itself via the rule $[v, w]$ A `:-` A.

5.4 Transformational Semantics for Cyclic BLP

In this section, we define a semantics for general (i.e., possibly cyclic) blps. We only consider *ground* blps in the semantics as usual. First, we transform a cyclic blp $\boldsymbol{P}$ into an acyclic blp $\boldsymbol{P'}$, which captures all the non-trivial and non-redundant belief derivations. Then the model of $\boldsymbol{P}$ is defined to be the model of the acyclic blp $\boldsymbol{P'}$.

Like what we did in Chapter 4, to simplify the description of the transformation, we assume that each rule, R, has a unique identifier, denoted ID_R — a new propositional constant.

We start with the definition of dependency graph. Note that it is identical to the definition of dependency DAG except that dependency graphs can be defined on cyclic blps and thus are not DAGs.

Definition 5.2 *The **dependency graph**, $\mathcal{H}$, of a ground blp **P** is a directed bipartite graph whose nodes are partitioned into a set of **atom nodes** (*a-nodes*, for short) and **rule nodes** (*r-nodes*, for short). The nodes and edges are defined as follows:*

- *For each atom A in **P**, $\mathcal{H}$ has an a-node labeled A.*
- *For each rule R in **P**, $\mathcal{H}$ has an r-node labeled with* proposition ID_R.
- *For each rule R in **P**, an edge goes from the r-node labeled ID_R to the a-node labeled with R's head.*
- *For each rule R in **P** and each atom A that appears in R's body, an edge goes from the a-node labeled A to the r-node labeled ID_R.*
- *Each edge that goes from an r-node, labeled R, to an a-node is labeled with the belief factor (interval) associated with the rule R.* □

The dependency graph $\mathcal{H}$ describes the dependency relation over $B_{\boldsymbol{P}}$. Not only does $\mathcal{H}$ stores the information whether an atom A depends on another atom B, but also the structural information such as through which rules (or through which path) A depends on B. The structural information will be useful for us to split the undesired loop influence from the other supports.

Definition 5.3 *Let **P** be a blp and $\mathcal{H}$ be **P**'s dependency graph. A directed graph $\mathcal{G}$ is called a **partial-proof DAG of P for the atom** A **(pp-DAG for** A**, for short)** if it has the following properties:*

1. *$\mathcal{G}$ is a maximal acyclic subgraph of $\mathcal{H}$ satisfying conditions 2-4, below.*
2. *Node A is the root of $\mathcal{G}$, i.e., every node in $\mathcal{G}$ is on a path leading to A.*
3. *Every a-node in $\mathcal{G}$ belongs to clique(A) and has exactly one child.*

4. *If an a-node D belongs to clique(A), and D's parent is in $\mathcal{G}$, then D itself is also in $\mathcal{G}$.* □

It is clear that an atom A can have more than one pp-DAGs. Each of them corresponds to a successful SLD-style derivation path for $? - A$, starting from outside of *clique*(A). The following example helps illustrate the observation.

Example 5.1 Returning to the example in Section 5.2. Suppose that the rate of false positive test results for a certain disease is 20%. The certainty that someone who had a contact with a contagious person will also contract the same disease is 60%. An expert system uses the following BLP rules to generate possible diagnosis.

$$[0.8, 1] \quad disease(?X) \ \texttt{:-} \ test\_pos(?X).$$
$$[0.6, 1] \quad disease(?X) \ \texttt{:-} \ contact(?X, ?Y) \wedge disease(?Y).$$

Suppose that the test for two persons, p_3 and p_4, came back positive and p_3 and p_4 are known to have had a contact. We get the following blp P_8 after grounding.

P_8:

$$r_1: \ [0.8, 1] \quad disease(p_3) \ \texttt{:-} \ test\_pos(p_3).$$
$$r_2: \ [0.8, 1] \quad disease(p_4) \ \texttt{:-} \ test\_pos(p_4).$$
$$r_3: \ [0.6, 1] \quad disease(p_3) \ \texttt{:-} \ contact(p_3, p_4) \wedge disease(p_4).$$
$$r_4: \ [0.6, 1] \quad disease(p_4) \ \texttt{:-} \ contact(p_4, p_3) \wedge disease(p_3).$$
$$r_5: \ [1, 1] \quad test\_pos(p_3).$$
$$r_6: \ [1, 1] \quad test\_pos(p_4).$$
$$r_7: \ [1, 1] \quad contact(p_3, p_4).$$
$$r_8: \ [1, 1] \quad contact(p_4, p_3).$$

There are five atom cliques in P_8: $\{test\_pos(p_3)\}$, $\{test\_pos(p_4)\}$,

$\{contact(p_3, p_4)\}$, $\{contact(p_4, p_3)\}$ and $\{disease(p_3), disease(p_4)\}$. The dependency graph and the pp-DAGs are shown in Figure 5.2 and Figure 5.3, respectively. □

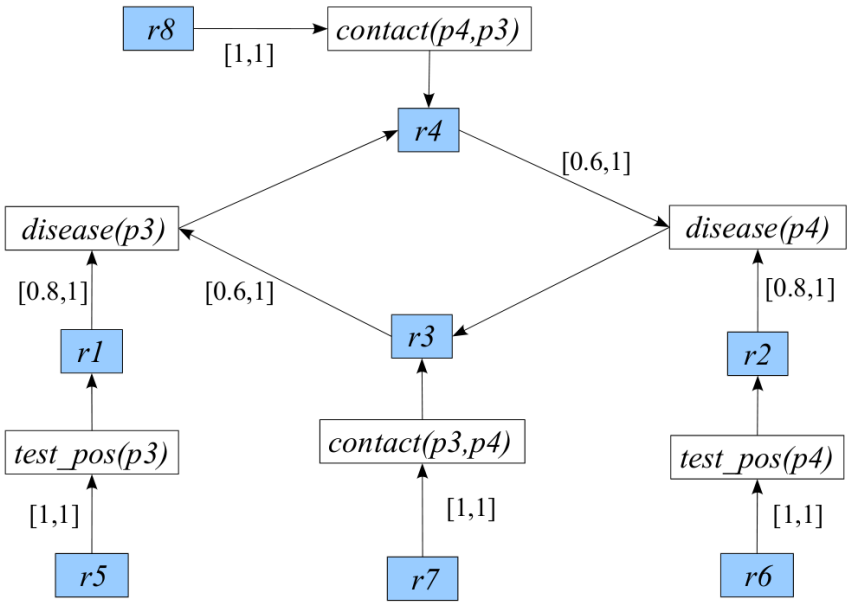


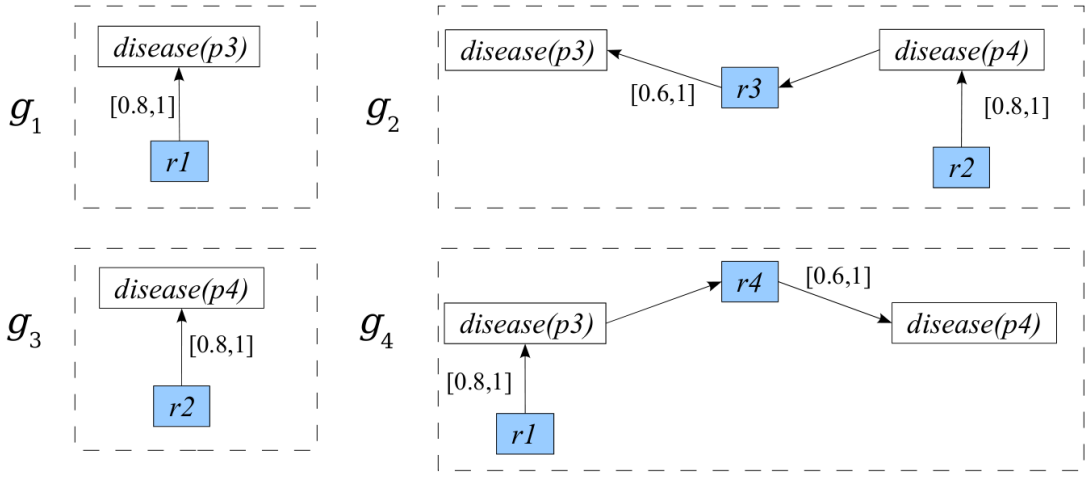
Figure 5.2: The dependency graph for Example 5.1

Definition 5.4 *Let $\mathcal{G}$ be a pp-DAG of $\boldsymbol{P}$ for the atom A. Another pp-DAG $\mathcal{G}'$ of $\boldsymbol{P}$ is said to be a* ***child pp-DAG of*** *$\mathcal{G}$ if:*

1. *$\mathcal{G}'$ is a subgraph of $\mathcal{G}$; and*
2. *$\mathcal{G}'$'s root, B, is a child of A's child in $\mathcal{G}$.* □

In Example 5.1, g_1 is a child pp-DAG of g_4, while g_3 is a child pp-DAG of g_2.

Now we are ready to define the transformation that converts cyclic blps to acyclic ones.



g_1, g_2 are pp-DAGs for $disease(p3)$, g_3, g_4 are pp-DAGs for $disease(p4)$.

Figure 5.3: The pp-DAGs in Example 5.1

Definition 5.5 ***Decyclification*** *of* $\boldsymbol{P}$*, denoted* $acyclic(\boldsymbol{P})$*, is obtained from* $\boldsymbol{P}$ *as follows. Let* $\mathcal{S}$ *be the set of new atoms labeled with pp-DAGs of* $\boldsymbol{P}$*:*

$$\mathcal{S} = \{A^{\mathcal{G}} | A \in B_{\boldsymbol{P}} \text{ and } \mathcal{G} \text{ is a pp-DAG with root } A\}$$

For each rule $R \in \boldsymbol{P}$ *of the form*

$$[v, w]\ A_0 \quad :\!-\quad A_1, \ldots, A_k, \overline{A_{k+1}}, \ldots, \overline{A_n}, D_1, \ldots, D_l, \overline{D_{l+1}}, \ldots, \overline{D_m}.$$

where $A_i \in clique(A_0), 1 \leq i \leq n$*,* $D_j \notin clique(A_0), 1 \leq j \leq m$*,*

1. *Replace* R *with the rule*

$$[v, w]\ A_0 \ :\!-\ ID_R$$

where ID_R *is the proposition that identifies* R*.*

2. *For every list* $\mathcal{G}_0, \ldots, \mathcal{G}_n$ *of pp-DAGs such that*

   - $\mathcal{G}_i$ *is a pp-DAG with the root* A_i*,* $0 \leq i \leq n$*, and*

- ID_R *is* A_0*'s child in* $\mathcal{G}_0$*, and*
- $\mathcal{G}_j$ *is a child pp-DAG of* $\mathcal{G}_0$*,* $1 \leq j \leq n$*,*

add the rules of the form

$$[v, w] \;\; A_0^{\mathcal{G}_0} \quad :- \quad A_1^{\mathcal{G}_1}, \ldots, A_k^{\mathcal{G}_k}, \overline{A_{k+1}^{\mathcal{G}_{k+1}}}, \ldots, \overline{A_n^{\mathcal{G}_n}}, D_1, \ldots, D_l, \overline{D_{l+1}}, \ldots, \overline{D_m}.$$
$$[1, 1] \;\; ID_R \quad :- \quad A_1^{\mathcal{G}_1}, \ldots, A_k^{\mathcal{G}_k}, \overline{A_{k+1}^{\mathcal{G}_{k+1}}}, \ldots, \overline{A_n^{\mathcal{G}_n}}, D_1, \ldots, D_l, \overline{D_{l+1}}, \ldots, \overline{D_m}.$$

where $A_i^{\mathcal{G}_i} \in \mathcal{S}, 0 \leq i \leq n$*, i.e.,* $A_i^{\mathcal{G}_i}$ *are auxiliary atoms for the transformation.* □

Intuitively, for each A, $A^{\mathcal{G}}$ is defined in such a way that its degree of belief is precisely that part of the belief in A, which is justified by the derivations that correspond to the pp-DAG $\mathcal{G}$. ID_R is defined in such a way that its degree of belief is the belief in R's body being derived without any loop influence. And the degree of belief in A_0 is obtained by combining the support to A_0 from all the ID_R's such that R has A_0 as head.

Note that in the resulting program, given any pair of atoms $A_i^{\mathcal{G}_i}$ and $A_j^{\mathcal{G}_j}$, $A_i^{\mathcal{G}_i}$ depends on $A_j^{\mathcal{G}_j}$ if and only if $\mathcal{G}_j$ is a child pp-DAG of $\mathcal{G}_i$. Thus, it is clear that the decyclification transformation eliminates all cycles.

Now we define the model of a general blp as follows.

Definition 5.6 *For a (possibly cyclic) blp* $\boldsymbol{P}$*,* $\widetilde{m}_{\boldsymbol{P}}$ *is a support function for* $\boldsymbol{P}$ *such that for any* $I \in \mathcal{TV}al(\boldsymbol{P})$*,*

$$\widetilde{m}_{\boldsymbol{P}}(I) = \sum_{I' \in \mathcal{TV}al(acyclic(\boldsymbol{P})), I'|_{B_{\boldsymbol{P}}} = I} \hat{m}_{acyclic(\boldsymbol{P})}(I')$$

The ***model*** *of* $\boldsymbol{P}$ *is a belief function for* $\boldsymbol{P}$*,* $\texttt{cmodel}_{\boldsymbol{P}}$*, such that for any formula* F *in* $\mathcal{B}ool(B_{\boldsymbol{P}})$*,* $\texttt{cmodel}_{\boldsymbol{P}}(F) = \texttt{model}_{acyclic(\boldsymbol{P})}(F)$*. (For acyclic blps,* $\hat{m}$ *and* `model` *are defined in Definitions 3.11 and 3.12.)* □

In other words, the semantics for acyclic BLP can be applied on $acyclic(\boldsymbol{P})$ to compute the model of $\boldsymbol{P}$. In practice, the query evaluation algorithm in Chapter 4 can be used on $acyclic(\boldsymbol{P})$ to compute $\texttt{cmodel}_{\boldsymbol{P}}(F)$ for any F in $\mathcal{B}ool(B_{\boldsymbol{P}})$. We will discuss this in more details in Section 5.6.

Example 5.2 *(Example 5.1 continued.)* Applying the decyclification transformation on P_8, we get the following acyclic blp P_8'.

P_8':

$$
\begin{array}{rrcl}
[0.8,1] & disease(p_3) & \texttt{:-} & r_1. \\
[1,1] & r_1 & \texttt{:-} & test\_pos(p_3). \\
[0.8,1] & disease(p_4) & \texttt{:-} & r_2. \\
[1,1] & r_2 & \texttt{:-} & test\_pos(p_4). \\
[0.6,1] & disease(p_3) & \texttt{:-} & r_3. \\
[1,1] & r_3 & \texttt{:-} & contact(p_3,p_4) \wedge disease^{g_2}(p_4). \\
[0.6,1] & disease(p_4) & \texttt{:-} & r_4. \\
[1,1] & r_4 & \texttt{:-} & contact(p_4,p_3) \wedge disease^{g_1}(p_3). \\
[0.8,1] & disease^{g_1}(p_3) & \texttt{:-} & test\_pos(p_3). \\
[0.8,1] & disease^{g_2}(p_4) & \texttt{:-} & test\_pos(p_4). \\
[0.6,1] & disease^{g_3}(p_3) & \texttt{:-} & contact(p_3,p_4) \wedge disease^{g_2}(p_4). \\
[0.6,1] & disease^{g_4}(p_4) & \texttt{:-} & contact(p_4,p_3) \wedge disease^{g_1}(p_3). \\
[1,1] & test\_pos(p_3). & & \\
[1,1] & test\_pos(p_4). & & \\
[1,1] & contact(p_3,p_4). & & \\
[1,1] & contact(p_4,p_3). & &
\end{array}
$$

If the combination function associated with *disease* is Φ^{DS}, we get the following conclusions: $\texttt{cmodel}_{P_8'}(disease(p_3)) = \texttt{cmodel}_{P_8'}(disease(p_4)) = 0.896$. Note that the support for p_3 having the disease is greater than 0.8 because p_3

has positive test results *and* the prior contact with p_4 pumps up the confidence in the diagnosis. Note that the decyclification transformation eliminates the self-supporting feedback loop of $disease(p_3)$ and $disease(p_4)$. Otherwise, the belief in $disease(p_3)$ and the belief in $disease(p_4)$ would have ended up close to 1 via these self-supporting feedback loops. □

Lemma 5.1 *Let* $\boldsymbol{P}$ *be an acyclic blp, then the following holds:*

1. *For every rule* R, $[v, w]\ A\ \texttt{:-}\ Body$, *in* $\boldsymbol{P}$,

$$\Big\{[v, w]\ A\ \texttt{:-}\ ID_R,\ [1, 1]\ ID_R\ \texttt{:-}\ Body\Big\} \subseteq acyclic(\boldsymbol{P})$$

2. *Let* L *be a rule in* $acyclic(\boldsymbol{P})$ *such that for any rule* R, $[v, w]\ A\ \texttt{:-}\ Body$, *in* $\boldsymbol{P}$, $L \notin \Big\{[v, w]\ A\ \texttt{:-}\ ID_R,\ [1, 1]\ ID_R\ \texttt{:-}\ Body\Big\}$. *Let* H *be* L*'s head, then* $H \notin B_{\boldsymbol{P}}$ *and* H *is not in the body of any rule in* $acyclic(\boldsymbol{P})$.

where ID_R *is the proposition that identifies* R.

Proof: Since $\boldsymbol{P}$ is acyclic, for every atom X in $\boldsymbol{P}$, we have $|clique(X)| = 1$. Hence there is a one-to-one mapping from the set of rules with X as head to the set of pp-DAGs for X: every rule r is mapped to a pp-DAG which has only two nodes: node X and node r. So there is not any pp-DAG for X which has a child pp-DAG.

The decyclification transformation of $\boldsymbol{P}$ is actually as follows:
For every rule R, $[v, w]\ A\ \texttt{:-}\ Body$, in $\boldsymbol{P}$, (there will not be any atom in $clique(A)$ appearing in $Body$), replace R with the rule

$$[v, w]\ A\ \texttt{:-}\ ID_R$$

plus, for every list $\mathcal{G}_0, \ldots, \mathcal{G}_n$ of pp-DAGs such that $\mathcal{G}_i$ is a pp-DAG with the

root A_i, $0 \leq i \leq n$, and ID_R is A_0's child in $\mathcal{G}_0$, we add the rules of the form

$$\begin{array}{lll} [v,w]\ A_0^{\mathcal{G}_0} & \texttt{:-} & Body. \\ [1,1]\ ID_R & \texttt{:-} & Body. \end{array} \tag{5.1}$$

Till now the first part of the lemma is proved.

As for the second part of the lemma, it is not hard to see that if L is a rule in $acyclic(\boldsymbol{P})$ such that $L \notin \{[v,w]\ A\ \texttt{:-}\ ID_R,\ [1,1]\ ID_R\ \texttt{:-}\ Body\}$ for any rule R, $[v,w]\ A\ \texttt{:-}\ Body$, in $\boldsymbol{P}$, L must be of the form (5.1). The head of L, $A_0^{\mathcal{G}_0}$, is not in $B_{\boldsymbol{P}}$ and is not in the body of any rule in $acyclic(\boldsymbol{P})$. □

The following theorem shows that the semantics of general blps is an extension of the semantics for acyclic blps.

Theorem 5.2 (Backward Compatibility) *Let $\boldsymbol{P}$ be an acyclic blp, and $F \in \mathcal{B}ool(B_{\boldsymbol{P}})$. Then $\widetilde{m}_{\boldsymbol{P}} = \hat{m}_{\boldsymbol{P}}$ and $\texttt{cmodel}_{\boldsymbol{P}}(F) = \texttt{model}_{\boldsymbol{P}}(F)$.* □

The proof of Theorem 5.2 can be found in Appendix A.8.

The following theorem shows that defining the semantics of BLP through the decyclification is "reasonable" because it discards self-supported beliefs, i.e., belief in A produced by the rules that contain A in their bodies.

Theorem 5.3 (Self-support) *Let $\boldsymbol{P}$ be a (possibly cyclic) blp, and A an atom in $B_{\boldsymbol{P}}$. Let $\boldsymbol{P}'_A$ be the blp obtained from $\boldsymbol{P}$ by deleting all the rules that contain A in their bodies. Then $\texttt{cmodel}_{\boldsymbol{P}}(A) = \texttt{cmodel}_{\boldsymbol{P}'_A}(A)$.* □

The proof of Theorem 5.3 can be found in Appendix A.9.

Example 5.3 (Example 5.2 continued.) Let P_9 be $P_8 - \{R_4\}$ where P_8 is the program in Example 5.1 and R_4 is the fourth rule of P_8. In the BLP semantics,

P_9, P_8 and P'_8 (the decyclification of P_8, as shown in Example 5.2) yield the same amount of support in $disease(p_3)$. □

5.5 Fixpoint Semantics and Modular Acyclicity

We now provide an alternative, fixpoint semantics for general blps, and show that the fixpoint semantics can be simplified for a special class of cyclic blps, called *modularly acyclic* blps.

First, for atom cliques we define some terms similar to those in Definition 3.9.

Definition 5.7 *Let $\boldsymbol{P}$ be a blp, I a truth valuation, and $\mathcal{C}$ an atom clique in the dependency graph of $\boldsymbol{P}$. $\boldsymbol{P}(\mathcal{C})$ is defined as the set of rules in $\boldsymbol{P}$ that has an atom from $\mathcal{C}$ in the head.* ***$\boldsymbol{P}$'s reduct under I with respect to $\mathcal{C}$****, denoted $\boldsymbol{P}_I(\mathcal{C})$,* [2] *is obtained from $\boldsymbol{P}(\mathcal{C})$ by*

1. *Replace a rule body with $false$ if it contains an atom $X \notin \mathcal{C}$ and $I(X) \neq \mathbf{t}$.*
2. *Deleting every atom $X \notin \mathcal{C}$ such that $I(X) = \mathbf{t}$.*
3. *If the combination Φ_A is such that $\forall v, w\ \Phi_A([v, w], [a, b]) = [a, b]$, and $\boldsymbol{P}$ has a fact of the form $[a, b]\ A$, then delete all the other rules with A in head.*

If $\boldsymbol{P}_I(\mathcal{C})$ is acyclic, we say $\boldsymbol{P}$ is ***weakly cyclic with respect to I and $\mathcal{C}$****.* □

Next we define a $\hat{T}_{\boldsymbol{P},Ord}$ operator.

[2] *Note that the definitions of $\boldsymbol{P}(\mathcal{C})$ and $\boldsymbol{P}_I(\mathcal{C})$ here is different from the definitions of $\boldsymbol{P}(X)$ and $\boldsymbol{P}_I(X)$ (in Definition 3.9): $\mathcal{C}$ is an atom clique, while X is an atom.*

Definition 5.8 *Let $\boldsymbol{P}$ be a blp with n atom cliques and a clique ordering Ord. Let $\mathcal{C}_1, \ldots, \mathcal{C}_n$ be the atom cliques of $\boldsymbol{P}$, such that $Ord(\mathcal{C}_i) = i, 1 \leq i \leq n$, and let $\alpha_0 = \varnothing$, $\alpha_i = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_i, 1 \leq i \leq n$.*

Given a support function m for $\alpha_k, 0 \leq k < n$, $\hat{T}_{\boldsymbol{P},Ord}(m)$ is a support function for α_{k+1} such that for every truth valuation $I \in \mathcal{TV}al(\alpha_{k+1})$,

$$\hat{T}_{\boldsymbol{P},Ord}(m)(I) = m(I \mid_{\alpha_k}) \cdot \widetilde{m}_Q(I \mid_{\mathcal{C}_{k+1}}) \tag{5.2}$$

where $Q = \boldsymbol{P}_{I|_{\alpha_k}}(\mathcal{C}_{k+1})$. □

Theorem 5.4 (Equivalence of fixpoint and transformational semantics) *Let $\boldsymbol{P}$ be a blp with n atom cliques and Ord a clique ordering of $\boldsymbol{P}$. Beginning with $m_0 = m_\varnothing$, let m_k be $\hat{T}^{\uparrow k}_{\boldsymbol{P},Ord}(m_0)$, $k = 0, 1, \ldots, n$. The support function m_n coincides with $\widetilde{m}_{\boldsymbol{P}}$.* □

The proof of Theorem 5.4 can be found in Appendix A.10.

The above theorem shows that the fixpoint semantics does not depend on the choice of the clique ordering in $\boldsymbol{P}$ and that this semantics coincides with the transformational semantics of Section 5.4.

Next, we will show that the computation in (5.2) can be simplified for a special class of cyclic blps.

Definition 5.9 *Let $\boldsymbol{P}$ be a blp with n atom cliques and a clique ordering Ord. Let $\mathcal{C}_1, \ldots, \mathcal{C}_n$ be the atom cliques of $\boldsymbol{P}$, such that $Ord(\mathcal{C}_i) = i, 1 \leq i \leq n$, and let $\alpha_0 = \varnothing$, $\alpha_i = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_i, 1 \leq i \leq n$. Also let $m_k = \hat{T}^{\uparrow k}_{\boldsymbol{P},Ord}(m_\varnothing)$, $k = 0, 1, \ldots, n$.*

*$\boldsymbol{P}$ is **modularly acyclic** if for every $0 \leq k \leq n-1$ and for every $I \in \mathcal{TV}al(\alpha_k)$ such that $m_k(I) \neq 0$, $\boldsymbol{P}$ is weakly cyclic with respect to I and $\mathcal{C}_{k+1}$.* □

If $\boldsymbol{P}$ is modularly acyclic, it follows from Theorem 5.2 that $\widetilde{m}_Q(I\ |_{\mathcal{C}_{k+1}})$ in (5.2) is equivalent to $\hat{m}_Q(I\ |_{\mathcal{C}_{k+1}})$, where Q is $\boldsymbol{P}_{I|_{\alpha_k}}(\mathcal{C}_{k+1})$, as defined in Definition 5.8. (Indeed, it follows from Definition 5.9 that Q is acyclic if $\boldsymbol{P}$ is modularly acyclic.) Since the computation of $\hat{m}$ does not involve decyclification, the computation of the model of a modularly acyclic blp can be greatly simplified.

Proposition 5.5 *Let $\boldsymbol{P}$ be a blp. If in every cycle in $\boldsymbol{P}$ (i.e., in every cycle in the dependency graph of $\boldsymbol{P}$), there is a rule R such that some atom in R's body is not in the head of any rule, then $\boldsymbol{P}$ is modularly acyclic.*

Proof: Let $\boldsymbol{P}$ be a blp with n atom cliques and a clique ordering Ord. Let $\mathcal{C}_1, \ldots, \mathcal{C}_n$ be the atom cliques of $\boldsymbol{P}$, such that $Ord(\mathcal{C}_i) = i, 1 \leq i \leq n$, and let $\alpha_0 = \varnothing$, $\alpha_i = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_i, 1 \leq i \leq n$. Also let $m_k = \hat{T}^{\uparrow k}_{\boldsymbol{P},Ord}(m_\varnothing)$, $k = 0, 1, \ldots, n$.

For every $0 \leq k \leq n-1$, consider the cycles in the dependency graph of $\boldsymbol{P}(\mathcal{C}_{k+1})$. In every such cycle, there is a rule R such that some atom X in R's body is not in the head of any rule. (According to the definition of cliques, we know $X \in \alpha_k$.) According to Definition 3.11 and Definition 3.10, for every $I \in \mathcal{TV}al(\alpha_k)$ such that $m_k(I) \neq 0$, $I(X) = \mathbf{u}$. Hence as to Definition 5.7, in $\boldsymbol{P}_I(\mathcal{C}_{k+1})$, R's body is replaced with $false$, and the cycle is broken. So, $\boldsymbol{P}_I(\mathcal{C}_{k+1})$ is acyclic and $\boldsymbol{P}$ is weakly cyclic with respect to I and $\mathcal{C}_{k+1}$. According to Definition 5.9 $\boldsymbol{P}$ is modularly acyclic. □

Example 5.4 Let us return to the diagnosis case for p_1 and p_2 in Section 5.2. Suppose that the test for two persons, p_1 and p_2, came back positive, but there

is no evidence that p_1 and p_2 had contact. The resulting blp P_{10} is

$$
\begin{aligned}
[0.8, 1] \quad & disease(p_1) \ \texttt{:-} \ test\_pos(p_1). \\
[0.8, 1] \quad & disease(p_2) \ \texttt{:-} \ test\_pos(p_2). \\
[0.6, 1] \quad & disease(p_1) \ \texttt{:-} \ contact(p_1, p_2) \wedge disease(p_2). \\
[0.6, 1] \quad & disease(p_2) \ \texttt{:-} \ contact(p_2, p_1) \wedge disease(p_1). \\
[1, 1] \quad & test\_pos(p_1). \\
[1, 1] \quad & test\_pos(p_2).
\end{aligned}
$$

with atom cliques $\mathcal{C}_1 = \{test\_pos(p_1)\}$, $\mathcal{C}_2 = \{test\_pos(p_2)\}$, $\mathcal{C}_3 = \{contact(p_1, p_2)\}$, $\mathcal{C}_4 = \{contact(p_2, p_1)\}$, $\mathcal{C}_5 = \{disease(p_1), disease(p_2)\}$.

Since $contact(p_1, p_2)$ and $contact(p_2, p_1)$ are not supported by any rule, it follows from Proposition 5.5 that P_{10} is modularly acyclic. The belief in $disease(p_1)$ is 0.8 and so is the belief in $disease(p_2)$. □

More interestingly, a blp can be modularly acyclic even when the condition in Proposition 5.5 is not satisfied, as shown in the following example.

Example 5.5 Consider a blp P_{11}

$$
\begin{aligned}
[0.8, 1] \quad & a \ \texttt{:-} \ d. \\
[0.8, 1] \quad & b \ \texttt{:-} \ e. \\
[0.6, 1] \quad & a \ \texttt{:-} \ b \wedge c_1. \\
[0.6, 1] \quad & b \ \texttt{:-} \ a \wedge c_2. \\
[1, 1] \quad & d. \\
[1, 1] \quad & e. \\
[0.5, 0.5] \quad & c_1. \\
[1, 1] \quad & c_2 \ \texttt{:-} \ \overline{c_1}.
\end{aligned}
$$

According to Definition 5.9, P_{11} is modularly acyclic. The underlying intuition is as follows. The last rule is the only rule that supports c_2, so we know that

c_2 and c_1 can not both be true in a truth valuation. Consequently, the rule $[0.6, 1]$ a :- $b \wedge c_1$ and the rule $[0.6, 1]$ b :- $a \wedge c_2$ do not both fire in a truth valuation. So, the cycle is "weak" and this program is modularly acyclic. □

5.6 Query Evaluation

In this section, we adapt the bottom-up inferencing algorithm and query evaluation algorithm in Chapter 4 to cyclic blps.

Ground Case. Let $\boldsymbol{P}$ be a ground cyclic blp, Algorithm 1 will output $\widetilde{m}_{\boldsymbol{P}}$ as defined in Definition 5.6 if the input to the algorithm is the decyclification of $\boldsymbol{P}$, $acyclic(\boldsymbol{P})$. Algorithm 2 will output $\texttt{cmodel}_{\boldsymbol{P}}(g)$ as defined in Definition 5.6 if the input to the algorithm is $acyclic(\boldsymbol{P})$, a query $? - \ g$ and the proof DAG of $acyclic(\boldsymbol{P})$ for query $? - \ g$.

Non-ground Case. Cyclic dependencies may cause infinite loops in SLD-resolutions during query evaluation of non-ground case. To address this problem, we use SLG-resolution [79] instead to construct *SLG-trees*.

The main idea of SLG-resolution is to keep a table of answers to the subgoals in cycles, and to resolve repeated subgoals against answers from the table instead of against program clauses.

Figure 5.4 illustrates a simple example of using SLG-resolution to construct an SLG-tree. When trying to resolve $? - p(Y), t(X, Y)$, we can find that the subgoal $? - p(Y)$ is actually a repetition of $? - p(X)$ which we already considered, so $?-p(Y)$ should be resolved against the answer table for $?-p(X)$. Another resolution path for $? - p(X)$ yields answers $X = a$ and $X = b$, which are then stored in the answer table for $? - p(X)$ (as shown by dashed lines in

Figure 5.4) and are used to resolve $? - p(Y), t(X, Y)$ (as shown by dotted lines in Figure 5.4).

r1 [0.5,0.5] *p*(*X*) :- *p*(*Y*), *t*(*X*,*Y*)
r2 [0.5,0.5] *p*(*X*) :- *s*(*X*)
r3 [0.5,0.5] *s*(*a*)
r4 [0.5,0.5] *s*(*b*)
r5 [1,1] *t*(*a*,*b*)
r6 [1,1] *t*(*b*,*a*)

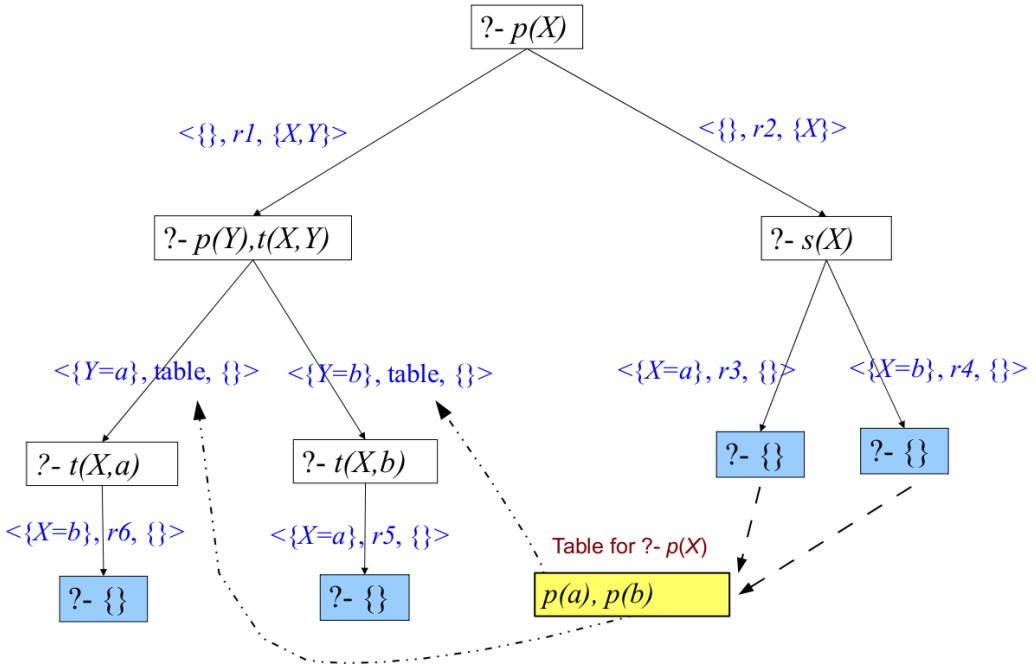


Figure 5.4: An example for SLG-resolution

To evaluate query $? - Goal$ on non-ground cyclic blp $\boldsymbol{P}$, we execute the following steps:

1. Execute SLG-resolution as described in [79] to obtain the SLG-tree T and a set S of positive instances of $Goal$.

2. Build the ground blp $\boldsymbol{P}'$ consisting of the rules appearing in T, and build its decyclification $acyclic(\boldsymbol{P}')$.

3. For each $g \in S$, execute Algorithm 2 with input $acyclic(\boldsymbol{P}')$, $?- \ g$ and the proof DAG of $acyclic(\boldsymbol{P}')$ for $?- \ g$, the output would be $\texttt{cmodel}_{\boldsymbol{P}'}(g)$ as defined in Definition 5.6.

Since $\boldsymbol{P}'$ contains all the rules that are relevant to the query in the grounding of $\boldsymbol{P}$, $\texttt{cmodel}_{\boldsymbol{P}'}(g)$ is equivalent to $\texttt{cmodel}_{\boldsymbol{P}}(g)$. Thus the query evaluation algorithm has been adapted to non-ground cyclic blps.

5.7 Diagnosis Example Revisited

In this section, we will contrast our method with an alternative approach of eliminating cycles by adding time parameters, which is utilized in [22] and implicitly in [73]. Adopting a similar methodology, the program of Example 5.1 can be transformed to the following blp:

$$
\begin{array}{rl}
[0.8,1] & disease(p_3,T) \ \texttt{:-} \ test\_pos(p_3). \\
[0.8,1] & disease(p_4,T) \ \texttt{:-} \ test\_pos(p_4). \\
[0.6,1] & disease(p_3,T) \ \texttt{:-} \ contact(p_3,p_4) \wedge disease(p_4,T-1). \\
[0.6,1] & disease(p_4,T) \ \texttt{:-} \ contact(p_4,p_3) \wedge disease(p_3,T-1). \\
[1,1] & disease(p_3,T) \ \texttt{:-} \ disease(p_3,T-1). \\
[1,1] & disease(p_4,T) \ \texttt{:-} \ disease(p_4,T-1). \\
[1,1] & test\_pos(p_3). \\
[1,1] & test\_pos(p_4). \\
[1,1] & contact(p_3,p_4). \\
[1,1] & contact(p_4,p_3).
\end{array}
$$

As a consequence of adding time parameters, the fifth and sixth rules must be added to ensure consistency. It is also worth noting that the first two rules assert that the test results provide support for diagnoses at any time point.

It is not difficult to observe the differences between the above transformed program and P_8' in Example 2 by our approach. One critical question in the time parameter methodology in [22] is, for a query $q(.)$, at which time point t does $q(., t)$ yield the correct answer. In [73], this problem is avoided by choosing the stationary state. However, this is based on a restriction that only stationary dynamic Bayesian networks can be modeled. Another assumption in the time parameter methodology is that an atom without time parameter takes the same value all the time. In this particular example, such an assumption translates to that p_3 and p_4 are having contact at all the time points. Obviously, this assumption may not hold in all applications. Our approach avoids the above problems by providing an alternative method to eliminate cycles.

It is worth noting that, the fact that we do not use the time parameter methodology to eliminate cycles *does not* mean that we do not allow time parameters. In the applications where time parameters are appropriate and feedbacks over time are desirable, time parameters may also be encoded into blps, e.g., $[0.9, 1]\ p(X, T)\ \texttt{:-}\ p(X, T-1)$.

It is also worth noting that the decyclification approach we introduced is not limited to BLP, and that it is generally applicable to other implication-based quantitative reasoning frameworks.

Chapter 6

Belief Logic Programs with Correlated Facts

So far we have been dealing with *structural correlation* among rules. This correlation arises due to the fact that rules might be sharing parts of their bodies and thus the corresponding inferred degrees of belief are not completely independent. However, there is also a different kind of correlation, which is not less important. We call this *ad hoc* or *quantitative* correlation. This type of correlation might exist among the facts used by the knowledge base and is determined by statistical analysis of the data, via learning algorithms, or simply by acclamation. For instance, it is generally accepted that advertising and sales are correlated, and statistical analysis might determine the exact degree of correlation in each particular case.

In this chapter we extend the BLP framework and capture ad hoc correlation through the notion of correlation formulas. We then extend the results of the previous chapters to this new framework.

Definition 6.1 *Given a support function m for a set of atoms $\{A_1, ..., A_n\}$, let*

*m_k be the projection of m on $\{A_k\}$, where $1 \leq k \leq n$. We say that the atoms $A_1, ..., A_n$ are **uncorrelated** if for any truth valuation $I \in \mathcal{TV}al(\{A_1, ..., A_n\})$,*

$$m(I) = \prod_{1 \leq k \leq n} m_k(I \mid_{\{A_k\}})$$

*Otherwise we say that the atoms $A_1, ..., A_n$ are **correlated**.* □

Example 6.1 A Web service collects opinions from two movie websites and provides recommendations to the users. One rule used by that site is:

$$[1, 1] \quad recom(?x) \ \texttt{:-} \ popular(?x) \wedge good\_review(?x)$$

Suppose Website1 provides information about popularity by measuring box office sales, which includes the following information about *movie_xyz*:

$$[0.6, 0.7] \quad popular(movie\_xyz)$$

In addition, Website2 is consulted for movie reviews; it has the following information:

$$[0.6, 0.7] \quad good\_review(movie\_xyz)$$

Assuming movie popularity and reviews are uncorrelated, the Web service would recommend *movie_xyz* with the confidence of 0.36.

On a second thought, it is clear that movie popularity in terms of the box office success and reviews are positively correlated: better reviews cause more sales and smash sales tend to lead to better reviews, since people tend to flock to see the same movie not without a reason. Therefore, evidential support for the statement "movie_xyz *is popular and receives good reviews*" should be higher than 0.36, and so our Web service should be expected to recommend *movie_xyz* with certainty higher than 0.36. □

We now extend BLP to enable it to handle correlated facts. First we assume that the available information about correlation is complete and then generalize the treatment to the case when this information is only partial.

6.1 Correlation Formula

In this section, we extend BLP with *correlation formulas*, which are intended to provide complete information about correlation among multiple facts.

Definition 6.2 *A **correlation formula** is a statement of the form*

$$A_1 \otimes ... \otimes A_n = \{C_1, ...C_k\} \tag{6.1}$$

where $n > 1$ and $A_1, ..., A_n$ are positive atoms and C_i, $1 \leq i \leq k$, are expressions of the form $str_i \rhd \mu_i$. Here str_i is an n-tuple whose elements are the truth values $\{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ such that $str_i \neq str_j$ if $i \neq j$. In addition, $0 \leq \mu_i \leq 1$ and $\sum_{1 \leq i \leq k} \mu_i = 1$.

*Any correlation formula, F, of the form (6.1) uniquely determines a support function over $\{A_1, ..., A_n\}$, which we call an associated **correlation function** and denote it as $corr(F)$. It is defined as follows. Let $I \in \mathcal{TV}al(\{A_1, ..., A_n\})$ be a truth valuation such that $I(A_j) = \tau_j$, where $\tau_j \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ and $1 \leq j \leq n$. Then*

- *If there is a C_i in (6.1) of the form $\tau_1 ... \tau_n \rhd \mu_i$, then $corr(F)(I) = \mu_i$.*[1]
- *Otherwise, $corr(F)(I) = 0$.* □

[1] *For brevity, we will write the n-tuples of truth values without punctuation. For instance, we will write* **tfu** *instead of* $\langle \mathbf{t}, \mathbf{f}, \mathbf{u} \rangle$.

Example 6.2 Given a correlation formula F:

$$a \otimes b = \{ \mathbf{tt} \rhd 0.5;\ \mathbf{ff} \rhd 0.5; \}$$

$corr(F)$ is a support function over $\{a, b\}$.

$$\begin{aligned} corr(F)(I) &= 0.5, \quad \text{if } I(a) = I(b) = \mathbf{t}; \\ corr(F)(I) &= 0.5, \quad \text{if } I(a) = I(b) = \mathbf{f}; \\ corr(F)(I) &= 0, \quad \text{otherwise.} \end{aligned}$$

□

A **belief logic program with correlation formulas** (or a **blp-cf**, for short) is composed of annotated rules and correlation formulas, such that the atoms appearing in correlation formulas do not appear in other correlation formulas or in the heads of the annotated rules. As in a regular blp, there can be no circular dependency in a blp-cf.

We represent blp-cf's as $\langle \boldsymbol{P}, \mathbf{CF} \rangle$, where $\boldsymbol{P}$ is a set of annotated rules and **CF** is a set of correlation formulas.

We now extend the BLP semantics to blp's augmented with correlation information. First, we define the Herbrand universes and bases in the obvious way: If $\langle \boldsymbol{P}, \mathbf{CF} \rangle$ is a blp-cf, then its **Herbrand Universe** $U_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}$ is the set of all constants, *excluding* **t**, **f**, and **u**, that occur in the program. The **Herbrand Base** $B_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}$ is the set of all atoms that can be constructed out of the predicates in the program and the elements of the Herbrand universe.

The definitions of **truth valuation**, **support function**, **belief function** are the same as for the regular blps.

We now define the amount of support for truth valuations provided by blp-cf programs.

Definition 6.3 *Given a blp-cf* $\langle \boldsymbol{P}, \boldsymbol{CF} \rangle$ *and a truth valuation* I *over* $B_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle}$*, we define*

$$\hat{m}_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle} = \prod_{F \in \boldsymbol{CF}} corr(F)(I \mid_{atm(F)}) \cdot \prod_{X \in B_{\boldsymbol{P}} - atm(\boldsymbol{CF})} s_{\boldsymbol{P}}(I, X)$$

where $atm(F)$ *and* $atm(\boldsymbol{CF})$ *are the sets of atoms that appear in* F *and* $\boldsymbol{CF}$*, respectively. Recall that* $s_{\boldsymbol{P}}(I, X)$ *is defined in Definition 3.10.* □

Theorem 6.1 *For any blp-cf* $\langle \boldsymbol{P}, \boldsymbol{CF} \rangle$*,* $\hat{m}_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle}$ *is a support function.* □

The proof of Theorem 6.1 can be found in Appendix A.11.

Definition 6.4 *Given a blp-cf* $\langle \boldsymbol{P}, \boldsymbol{CF} \rangle$*, its* ***cf-model*** *is a belief function defined as*

$$\texttt{CFmodel}_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle}(F) = \sum_{\substack{I \in \mathcal{TV}al(\langle \boldsymbol{P}, \boldsymbol{CF} \rangle) \\ such\ that\ I \models F}} \hat{m}_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle}(I)$$

for any $F \in \mathcal{B}ool(B_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle})$.[2] □

Example 6.3 Continuing with Example 6.1, suppose that the Web service also has the following formula that correlates popularity and reviews:

$$\begin{aligned} & popular(movie\_xyz) \ \otimes \ good\_review(movie\_xyz) \\ = \ & \{ \ \mathbf{tt} \rhd 0.55; \ \mathbf{tf} \rhd 0.03; \ \mathbf{tu} \rhd 0.02; \\ & \ \ \mathbf{ft} \rhd 0.05; \ \mathbf{ff} \rhd 0.23; \ \mathbf{fu} \rhd 0.02; \\ & \ \ \mathbf{ut} \rhd 0; \ \mathbf{uf} \rhd 0.04; \ \mathbf{uu} \rhd 0.06; \} \end{aligned}$$

This formula together with the rule

$$[1, 1] \quad recom(?x) \ \texttt{:-} \ popular(?x) \wedge good\_review(?x)$$

form a blp-cf P_{12}, which captures correlation between movie popularity and reviews.

[2] *Recall that* $\mathcal{B}ool(B_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle})$ *is the set of all Boolean formulas constructed out of the atoms in* $B_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle}$.

Let bel be the cf-model of P_{12} as defined in Definition 6.4. bel($popular(movie\_xyz)$) is 0.6 and bel($good\_review(movie\_xyz)$) is 0.6 — same as the belief value in Example 6.1.

On the other hand, bel($recom(movie\_xyz)$) = 0.55 as opposed to 0.36 in Example 6.1, where popularity and reviews were considered independent. The difference in the strength of recommendation is due to the fact that now these facts are treated as correlated. □

6.2 Query Evaluation

Dependency DAGs and proof DAGs can be modified to accommodate correlation formulas by treating them as facts with multiple atoms in head. For each correlation formula F, there is a leaf r-node labeled F, with edges going from this node to nodes labeled with atoms in $atm(F)$.

The bottom-up inferencing algorithm (Algorithm 1) introduced in Section 4.2 can be modified to accommodate correlated facts, with a slight change to the traversal step in Algorithm 1.

The modified algorithm is shown below as Algorithm 4. The difference from Algorithm 1 is that lines 8–22 in Algorithm 1 is replaced with lines 9–34 in Algorithm 4. Let $Base$ be the base of the support function constructed so far, visit a-node A:

- If A is already in $Base$, then do nothing and visit the next node.
- If A appears in some correlation formula $F \in \mathbf{CF}$ and A is not already in $Base$, then execute lines 11–18 of Algorithm 4 — expand m_{Base} to $m_{Base \cup atm(F)}$ as follows: for every $I \in \mathcal{TV}al(Base \cup atm(F))$, $m_{Base \cup atm(F)}(I) = m_{Base}(I \mid_{Base}) \cdot corr(F)(I \mid_{atm(F)})$, where $atm(F)$ is

the set of atoms that appear in F.

- If A does not appear in any correlation formula then execute lines 20–34 of Algorithm 4 — expand m_{Base} to $m_{Base \cup \{A\}}$ in exactly the same way as described in lines 8–22 of Algorithm 1.

Algorithm 4

(∗ Bottom-up inferencing with correlated formulas ∗)

Input: A ground blp-cf $\langle \boldsymbol{P}, \mathbf{CF} \rangle$.

Output: $\hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}$ as defined in Definition 6.3.

1. Construct the dependency DAG, $\mathcal{H}$, of $\boldsymbol{P}$
2. Let $< X_1, ..., X_n >$ be a post-order traversal of $\mathcal{H}$.
3. $m \leftarrow m_{\varnothing}$
4. $Visited \leftarrow \varnothing$
5. $Base \leftarrow \varnothing$
6. **for** $i \leftarrow 1$ **to** n **do**
7. **if** X is an a-node labeled with an atom A
8. **then**
9. **if** $A \notin Base$
10. **then**
11. **if** A appears in some $F \in \mathbf{CF}$
12. Let $atm(F)$ be the set of atoms that appear in F.
13. Initialize a new support function m' for $Base \cup atm(F)$.
14. **forall** $I \in \mathcal{TV}al(Base \cup atm(F))$ **do**
15. $m'(I) \Leftarrow m(I \mid_{Base}) \times corr(F)(I \mid_{atm(F)})$
16. $m \Leftarrow m'$
17. $Visited \Leftarrow Visited \cup atm(F)$
18. $Base \Leftarrow Base \cup atm(F)$
19. **else** (∗ A does not appear in **CF** ∗)

20. **then** Initialize a new support function m' for $Base \cup \{A\}$.
21. Let $Id_1, \ldots, Id_k$ be the labels of the child r-nodes of X.
22. **forall** $I \in \mathcal{TV}al(Base)$ such that $m(I) \neq 0$ **do**
23. $\mathcal{S} \leftarrow \varnothing$
24. **for** $j \leftarrow 1$ **to** k **do**
25. **if** $I(Id_j) = \mathbf{t}$
26. Let $[V_j, W_j]$ be the belief factor of the rule identified by Id_j.
27. $\mathcal{S} \leftarrow \mathcal{S} \cup \{[V_j, W_j]\}$
28. $[V, W] \leftarrow \Phi_A(\mathcal{S})$
29. $m'(I \uplus \{A \rightarrow \mathbf{t}\}) \leftarrow m(I) \cdot V$
30. $m'(I \uplus \{A \rightarrow \mathbf{f}\}) \leftarrow m(I) \cdot (1 - W)$
31. $m'(I \uplus \{A \rightarrow \mathbf{u}\}) \leftarrow m(I) \cdot (W - V)$
32. $m \leftarrow m'$
33. $Visited \leftarrow Visited \cup \{A\}$
34. $Base \leftarrow Base \cup \{A\}$
35. **else** ($*$ X is an r-node labeled with Id which is the identifier of rule R. $*$) Initialize a new support function m' for $Base \cup \{Id\}$.
36. Let $A_1, \ldots, A_k$ be the labels of the child a-nodes of X.
37. Let F be the body of R.
38. **forall** $I \in \mathcal{TV}al(Base)$ such that $m(I) \neq 0$ **do**
39. **forall** $\tau \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ **do**
40. **if** $\tau = I(F)$
41. **then** $m'(I \uplus \{Id \rightarrow \tau\}) \leftarrow m(I)$
42. $m \leftarrow m'$
43. $Visited \leftarrow Visited \cup \{Id\}$
44. $Base \leftarrow Base \cup \{Id\}$
45. $m \leftarrow projection(m, B_{\boldsymbol{P}})$
46. **return** m

Theorem 6.2 *The above modified algorithm computes the* $\hat{m}_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle}$ *as defined in Definition 6.3.* □

The proof of Theorem 6.2 can be found in Appendix A.12.

The query evaluation algorithm (Algorithm 2) in Section 4.3 can be adjusted to deal with correlated formulas in the same way we adjusted Algorithm 1 to Algorithm 4 (as described in Page 98). We call the resulting adjusted algorithm **Algorithm 5**.

Theorem 6.3 *Given a blp-cf* $\langle \boldsymbol{P}, \boldsymbol{CF} \rangle$ *and a query* $? - g$*, let* m_1 *be the outcome of the modified query evaluation algorithm and* m_2 *be the outcome of the modified bottom-up inferencing algorithm. The belief function constructed out of* m_1 *(via Definition 3.8) yields the same degree of belief in* g *as does the belief function constructed using* m_2*.*

Proof: The proof is similar to that of Theorem 4.4. First, one has to show that m_1 is a projection of m_2 on $\{g\}$. Then the result follows from Lemma 4.2. □

For a non-ground blp and a non-ground query, Algorithm 3 in Section 4.4 can first be used to construct pruned proof DAGs. Then the query can be answered by applying the *modified* query evaluation algorithm, Algorithm 5, on those pruned proof DAGs.

6.3 Cyclic Belief Logic Programs with Correlated Facts

Since there is a restriction that the atoms appearing in correlation formulas

do not appear in other correlation formulas or in the heads of the annotated rules, these atoms are not involved in any cyclic dependencies. Therefore, the BLP extension to correlated facts and the extension to cyclic blps are orthogonal to each other.

Definition 6.5 *For a blp-cf $\langle \boldsymbol{P}, \boldsymbol{CF} \rangle$ (where $\boldsymbol{P}$ is possibly cyclic), $\widetilde{m}_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle}$ is a support function for $B_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle}$ such that for any truth valuation I over $B_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle}$,*

$$\widetilde{m}_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle}(I) = \sum_{I' \in \mathcal{TV}al(B_{\langle acyclic(\boldsymbol{P}), \boldsymbol{CF} \rangle})), I'|_{B_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle}} = I} \hat{m}_{\langle acyclic(\boldsymbol{P}), \boldsymbol{CF} \rangle}(I')$$

*The **model** of $\langle \boldsymbol{P}, \boldsymbol{CF} \rangle$ is a belief function $\texttt{cmodel}_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle}$, such that for any formula F in $\mathcal{B}ool(B_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle})$, $\texttt{cmodel}_{\langle \boldsymbol{P}, \boldsymbol{CF} \rangle}(F) = \texttt{model}_{\langle acyclic(\boldsymbol{P}), \boldsymbol{CF} \rangle}(F)$. (For acyclic blp-cfs, $\hat{m}$ and* `model` *are defined in Definitions 6.3 and 6.4.)* □

For a blp-cf $\langle \boldsymbol{P}, \mathbf{CF} \rangle$ such that $\boldsymbol{P}$ is cyclic, the algorithms for bottom-up inferencing and query evaluation follow those in Section 5.6, with a modification on the atom-node traversing, as described in Section 6.2. We give the details below.

Ground Case. Let $\langle \boldsymbol{P}, \mathbf{CF} \rangle$ be a ground blp-cf such that $\boldsymbol{P}$ is cyclic, Algorithm 4 will output $\widetilde{m}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}$ as defined in Definition 6.5 if the input to the algorithm is $\langle acyclic(\boldsymbol{P}), \mathbf{CF} \rangle$. Algorithm 5 [3] will output $\texttt{cmodel}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}(g)$ as defined in Definition 6.5 if the input to the algorithm is $\langle acyclic(\boldsymbol{P}), \mathbf{CF} \rangle$, a query $? - g$ and the proof DAG for query $? - g$.

Non-ground Case. To evaluate query $? - Goal$ on non-ground blp-cf $\langle \boldsymbol{P}, \mathbf{CF} \rangle$ such that $\boldsymbol{P}$ is cyclic, we execute the following steps:

[3] Described on Page 101. Algorithm 5 is modified from Algorithm 2 to accommodate correlated facts.

1. Execute SLG-resolution as described in [79] to obtain the SLG-tree T and a set S of positive instances of $Goal$. Correlation formulas are treated as facts in SLG-resolution.

2. Build the ground blp $\boldsymbol{P'}$ consisting of the rules appearing in T, and build its decyclification $acyclic(\boldsymbol{P'})$.

3. For each $g \in S$, execute Algorithm 5 with input $\langle acyclic(\boldsymbol{P'}), \mathbf{CF} \rangle$, query $?- g$ and the proof DAG for $?- g$, the output would be $\texttt{cmodel}_{\langle \boldsymbol{P'}, \mathbf{CF} \rangle}(g)$ as defined in Definition 6.5.

Since $\boldsymbol{P'}$ contains all the rules that are relevant to the query in the grounding of $\boldsymbol{P}$, $\texttt{cmodel}_{\langle \boldsymbol{P'}, \mathbf{CF} \rangle}(g)$ is equivalent to $\texttt{cmodel}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}(g)$. Thus the query evaluation algorithm has been adapted to non-ground cyclic blp-cfs.

Chapter 7

Approximation of Belief Logic Programs

In Chapter 6 we discussed how to represent and reason with quantitative correlations among base facts. In many applications, complete quantitative correlation information is not always available. In this chapter we show that partial correlation information might suffice in certain cases. The basic idea is as follows. If correlation function is not available, we cannot use the results of Chapter 6 directly. However, we might be able to construct a different, *approximate* support function, which, if used in lieu of the correlation function, could yield the same belief factors for the query. We will formulate a general condition for such approximate to give correct answers.

The construction of the approximate correlation function is inspired by the concept of correlation from the probability theory, so we begin by recalling the definition of correlation from that theory.

7.1 Correlation in Probability Theory

Given two real-valued random variables X and Y, their *correlation coefficient* is defined as

$$\rho \;=\; \frac{E[X \cdot Y] - E[X] \cdot E[Y]}{\sqrt{\big(E[X^2] - (E[X])^2\big) \cdot \big(E[Y^2] - (E[Y])^2\big)}}$$

where $E[X]$ and $E[Y]$ are the expectations for X and Y, respectively.

It is known that $-1 \leq \rho \leq 1$, and ρ equals 1 or -1 if and only if X and Y are linearly dependent, i.e., $X = vY + w$, where v and w are real-valued constants.

We can apply these ideas to random Boolean variables X and Y, which can be viewed as real-valued random variables that take only the values 0 (for $false$) and 1 (for $true$). Suppose we know the joint distribution of random logic variables X and Y, i.e., the following four values are available:

$$\begin{aligned} prob[XY] &= a \\ prob[X\overline{Y}] &= b \\ prob[\overline{X}Y] &= c \\ prob[\overline{XY}] &= d \end{aligned} \tag{7.1}$$

Since $E[X \cdot Y] = prob[XY] = a$, $E[X] = E[X^2] = prob(X) = a + b$ and $E[Y] = E[Y^2] = prob(Y) = a + c$, by (7.1) we get

$$\rho = \frac{a - (a+b)(a+c)}{\sqrt{\big((a+b) - (a+b)^2\big)\big((a+c) - (a+c)^2\big)}}$$

Clearly, $\rho = 1$ if and only if $X = Y$, and $\rho = -1$ if and only if $X = \overline{Y}$. In the special cases when $prob(X) = 1$ or $prob(Y) = 1$, ρ is undefined.

Now, suppose we do *not* know the joint distribution (7.1) of X and Y, but we do know the distribution of X, $prob[X]$, and the distribution of Y, $prob[Y]$,

separately. In addition, supposed that the correlation coefficient ρ of X and Y is also known. We can then compute the joint distribution of X and Y by solving the following four equations for a, b, c and d:

$$\begin{aligned} &a+b+c+d=1\\ &a+b=prob[X]\\ &a+c=prob[Y]\\ &\frac{a-(a+b)(a+c)}{\sqrt{\big((a+b)-(a+b)^2\big)\big((a+c)-(a+c)^2\big)}}=\rho \end{aligned}$$

7.2 Belief Correlation Coefficient

We now perform a mental leap and replace expectations of the probability theory with support functions. This leap can be seen as a heuristic that blindly adapts the usual probabilistic correlation to BLP under the name of *belief correlation coefficients*. Such a heuristic might seem risky and unfounded, but luckily it does the job. Namely, we shall see that for a broad class of blp's, it is enough to know just these coefficients and complete correlation functions can be dispensed with.

Belief correlation coefficients can be computed only for pairs of facts, so one restriction that stems from this is that we will now limit our attention to the case of *binary* correlation formulas.

Definition 7.1 *Let A_1 and A_2 be a pair of atoms and suppose we know the support function for the set $\{A_1, A_2\}$, i.e., all the values $m_{\{A_1,A_2\}}(I_k)$ are known, where $I_1, ..., I_9 \in \mathcal{TV}al(\{A_1, A_2\})$ are defined as*

	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9
A_1	**t**	**t**	**t**	**f**	**f**	**f**	**u**	**u**	**u**
A_2	**t**	**f**	**u**	**t**	**f**	**u**	**t**	**f**	**u**

Let us define the values a, b, c, and d used in probabilistic correlation, as outlined in Section 7.1, as follows:

$$\begin{aligned} a &= m_{\{A_1,A_2\}}(I_1) \\ b &= m_{\{A_1,A_2\}}(I_2) + m_{\{A_1,A_2\}}(I_3) \\ c &= m_{\{A_1,A_2\}}(I_4) + m_{\{A_1,A_2\}}(I_7) \\ d &= m_{\{A_1,A_2\}}(I_5) + m_{\{A_1,A_2\}}(I_6) + m_{\{A_1,A_2\}}(I_8) + m_{\{A_1,A_2\}}(I_9) \end{aligned}$$

The ***belief correlation coefficient*** *of A_1 and A_2, denoted $A_1 \odot A_2$, is defined by the following formula:*

$$\frac{a - (a+b)(a+c)}{\sqrt{\big((a+b) - (a+b)^2\big)\big((a+c) - (a+c)^2\big)}}$$

□

We thus see that belief correlation coefficients have exactly the same form as probabilistic correlation, but the meaning of the parameters, a, b, c, and d is different.

Suppose the actual support function $m_{\{A_1,A_2\}}$ over the set $\{A_1, A_2\}$ is not known, but we do know $bel(A_1)$, $bel(A_2)$, and the belief correlation coefficient $A_1 \odot A_2$. By solving

$$\begin{aligned} &a + b + c + d = 1 \\ &a + b = bel(A_1) \\ &a + c = bel(A_2) \\ &\frac{a - (a+b)(a+c)}{\sqrt{\big((a+b) - (a+b)^2\big)\big((a+c) - (a+c)^2\big)}} = A_1 \odot A_2 \end{aligned}$$

we get

$$
\begin{aligned}
a &= bel(A_1) \cdot bel(A_2) \\
&\quad + \ A_1 \odot A_2 \cdot \sqrt{bel(A_1)} \cdot \sqrt{bel(A_2)} \cdot \sqrt{1 - bel(A_1)} \cdot \sqrt{1 - bel(A_2)} \\
b &= bel(A_1) - a \\
c &= bel(A_2) - a \\
d &= 1 + a - bel(A_1) - bel(A_2)
\end{aligned}
\tag{7.2}
$$

Note that knowing the values of a, b, c, and d does not give enough information to calculate the actual support function $m_{\{A_1,A_2\}}$. However, we can approximate $m_{\{A_1,A_2\}}$ by defining $\tilde{m}_{\{A_1,A_2\}}$ over the truth valuations $I_1, ..., I_9$ of Definition 7.1 as follows:

$$
\begin{aligned}
\tilde{m}_{\{A_1,A_2\}}(I_1) &= a \\
\tilde{m}_{\{A_1,A_2\}}(I_2) &= b \\
\tilde{m}_{\{A_1,A_2\}}(I_4) &= c \\
\tilde{m}_{\{A_1,A_2\}}(I_5) &= d
\end{aligned}
\tag{7.3}
$$

For $i = 3, 6, 7, 8, 9$, i.e., for all those valuations that take the value $\mathbf{u}$ on either A_1 or A_2, we set $\tilde{m}_{\{A_1,A_2\}}(I_i) = 0$.

The support function $\tilde{m}_{\{A_1,A_2\}}$ is a special case of what we call *u-to-f retraction* of $m_{\{A_1,A_2\}}$.

Definition 7.2 *Let m and m' be a pair of support functions for an atom set $\mathcal{S}$, and let $\mathcal{C}$ be a subset of $\mathcal{S}$. We say that m' is the* ***u-to-f retraction of*** *m* ***with respect to*** *$\mathcal{C}$ if for any truth valuation $I \in \mathcal{TV}al(\mathcal{S})$, the following holds:*

- *If there is an atom $X \in \mathcal{C}$ such that $I(X) = \mathbf{u}$, then $m'(I) = 0$.*
- *Otherwise, let $\mathcal{C}_I^{\mathbf{f}}$ be the subset of $\mathcal{C}$ such that $I \mid_{\mathcal{C}_I^{\mathbf{f}}} = \mathbf{f}$ and $I \mid_{\mathcal{C}-\mathcal{C}_I^{\mathbf{f}}} = \mathbf{t}$,*

then

$$m'(I) = \sum_{\substack{I' \in \mathcal{TV}al(\mathcal{S}) \text{ such that} \\ I'|_{\mathcal{S}-\mathcal{C}_I^{\mathbf{f}}} = I|_{\mathcal{S}-\mathcal{C}_I^{\mathbf{f}}} \text{ and} I'(\mathcal{C}_I^{\mathbf{f}}) \subseteq \{\mathbf{u},\mathbf{f}\}}} m(I')$$

□

By the above definition we can see that $\tilde{m}_{\{A_1,A_2\}}$ in (7.3) is the u-to-f retraction of $m_{\{A_1,A_2\}}$ with respect to $\{A_1, A_2\}$.

Lemma 7.1 *Let m and m' be support functions for an atom set $\mathcal{S}$ and $\mathcal{C}$ be a subset of $\mathcal{S}$. If m' is the u-to-f retraction of m with respect to $\mathcal{C}$ then for any $a \in \mathcal{S}$*

$$\sum_{\substack{I \in \mathcal{TV}al(\mathcal{S}) \\ \text{such that } I(a)=\mathbf{t}}} m'(I) = \sum_{\substack{I \in \mathcal{TV}al(\mathcal{S}) \\ \text{such that } I(a)=\mathbf{t}}} m(I)$$

□

The proof of Lemma 7.1 can be found in Appendix A.13.

Theorem 7.2 *Let $\langle \boldsymbol{P}, \boldsymbol{CF} \rangle$ be a blp-cf and $? - g$ be a query. Consider some correlation formula $F \in \boldsymbol{CF}$ and let $atm(F)$ be the set of atoms that appear in F. Also let F' be another correlation formula for $atm(F)$ such that $corr(F')$ is the u-to-f retraction of $corr(F)$ with respect to $atm(F)$. Finally, let $\texttt{bel}_1$ be the cf-model of $\langle \boldsymbol{P}, \boldsymbol{CF} \rangle$ (see Definition 6.4) and $\texttt{bel}_2$ be the cf-model of $\langle \boldsymbol{P}, \boldsymbol{CF} \cup \{F'\} - \{F\} \rangle$. If there is no atom $X \in atm(F)$ such that $\overline{X}$ appears in $\boldsymbol{P}$, then $\texttt{bel}_1(g) = \texttt{bel}_2(g)$.* □

The proof of Theorem 7.2 can be found in Appendix A.14.

We can now apply Theorem 7.2 to $\tilde{m}_{\{A_1,A_2\}}$ and $m_{\{A_1,A_2\}}$ discussed earlier. Here the m function is the actual, but unknown correlation function, while $\tilde{m}$ is the known, but approximate correlation function. The advantage of using $\tilde{m}$

is that much less information is needed to construct it—only the belief factors for A_1, A_2, and their belief correlation coefficient are needed.

More precisely, Theorem 7.2 lets us do the following trick. Consider a blp-cf $\langle \boldsymbol{P}, \mathbf{CF} \rangle$, and let F be a correlation formula in $\mathbf{CF}$ with a correlation function $corr(F) = m_{\{A_1,A_2\}}$. We can take the approximate correlation function $\tilde{m}_{\{A_1,A_2\}}$ and construct a correlation formula F' such that $corr(F') = \tilde{m}_{\{A_1,A_2\}}$. By Theorem 7.2, if $\overline{A_1}$ and $\overline{A_2}$ do not appear in $\boldsymbol{P}$ then the blp-cf $\langle \boldsymbol{P}, \mathbf{CF} \cup \{F'\} - \{F\} \rangle$ yields the same answer to queries as does $\langle \boldsymbol{P}, \mathbf{CF} \rangle$.

Note that if it so happens that the u-to-f retraction of $m_{\{A_1,A_2\}}$ equals $m_{\{A_1,A_2\}}$, then $\tilde{m}_{\{A_1,A_2\}} = m_{\{A_1,A_2\}}$ and the restriction that $\overline{A_1}$ and $\overline{A_2}$ do not appear in $\boldsymbol{P}$ is not needed: simply knowing $\texttt{bel}(A_1)$, $\texttt{bel}(A_2)$, and the belief correlation coefficient of A_1 and A_2 suffices for reconstructing the complete correlation picture for these atoms.

Example 7.1 Continuing with Example 6.1, suppose that (unlike in Example 6.3) the Web service does not supply complete correlation information for movie popularity and reviews. Instead, it has only partial information, which consists of the belief factors for $popular(movie\_xyz)$ and $good\_review(movie\_xyz)$, and of the belief correlation coefficient for $popular(movie\_xyz)$ and $good\_review(movie\_xyz)$. Suppose also that the belief factors are the same as in Example 6.3:

- $\texttt{bel}(popular(movie\_xyz)) = 0.6$
- $\texttt{bel}(good\_review(movie\_xyz)) = 0.6$
- $popular(movie\_xyz) \odot good\_review(movie\_xyz) = 0.79166$

By (7.2) we get $a = 0.55,\ b = 0.05,\ c = 0.05,\ d = 0.35$. So, although we do not have the exact correlation formula, we can approximate it with:

$$\begin{aligned} & \mathit{popular}(\mathit{movie\_xyz}) \ \otimes \ \mathit{good\_review}(\mathit{movie\_xyz}) \\ = \ & \{\ \mathbf{tt} \rhd 0.55;\ \mathbf{tf} \rhd 0.05;\ \mathbf{tu} \rhd 0; \\ & \quad \mathbf{ft} \rhd 0.05;\ \mathbf{ff} \rhd 0.35;\ \mathbf{fu} \rhd 0; \\ & \quad \mathbf{ut} \rhd 0;\ \mathbf{uf} \rhd 0;\ \mathbf{uu} \rhd 0; \} \end{aligned}$$

This formula together with the rule

$$[1,1] \quad \mathit{recom}(?x) \ \texttt{:-}\ \mathit{popular}(?x) \wedge \mathit{good\_review}(?x)$$

form a blp-cf, which is an approximation of the blp-cf in Example 6.3.

This program and the query $? - \ \mathit{recom}(\mathit{movie\_xyz})$ do not use negated literals, so Theorem 7.2 assures that the approximation program yields the same results as the program in Example 6.3. Indeed, the answer to the query is $\texttt{bel}(\mathit{recom}(\mathit{movie\_xyz})) \ = \ 0.55$, as in Example 6.3. □

Chapter 8

Conclusions and Future Directions

This dissertation has developed a novel logic theory, Belief Logic Programming (BLP), for reasoning with uncertainty. The main advantages and contributions include:

- BLP is based on the formalism of belief function in Dempster-Shafer theory, so it is inherently capable of modeling lack of information and combining evidence from different sources, an area where other uncertainty formalisms fall short.

- BLP is not a naive integration of Dempster-Shafer theory and logic programming. First, unlike the previous efforts in applying Dempster-Shafer theory in logic programming, such as [1, 2, 45], BLP can correlate structural information contained in derivation paths for beliefs and provides more accurate certainty estimates, as illustrated in the motivating example in Section 3.1 and Section 3.5. Secondly, BLP does not depend on

a particular method of combining evidence. In fact, different combination methods can be used simultaneously for different types of uncertain information.

- This dissertation defined both a model-theoretic semantics and a fixpoint semantics of BLP. The semantics are non-monotonic, and BLP is closely related to some other theories of non-monotonic reasoning, including defeasible reasoning and paraconsistent reasoning. Also developed are efficient algorithms that can answer queries for non-ground programs with the help of an SLD-like procedure.

- To address the common challenge of recursive loops in quantitative logic programming frameworks, BLP has been extended to cyclic programs by adopting a novel approach: instead of introducing time parameters to eliminate loops, we analyze the program structure and discard the support from unwanted loop influence. The proposed semantics are proved to be well-justified and are backward compatible with the semantics for acyclic BLP.

- BLP is also extended to capture quantitative correlation among basic facts through the notion of correlation formulas. This dissertation also describes a method to approximate correlation information when only partial correlation information is available. Under certain circumstances, this approximation method is proved to yield the same results as the precise method.

BLP has potential applications in the areas of business, science, health care, etc. Apart from traditional uses in expert systems, such a language can be used to integrate semantic Web services and information sources, such as sensor networks and forecasts, which deal with uncertain data.

The problems raised in this dissertation and our results point to a number of interesting directions worthy of further investigation. They include:

- Combining BLP with Annotated Logic Programming (ALP) [31]. ALP has great expressive power and efficient query processing, and, if extended with the ability to combining correlated pieces of uncertain information as was envisioned in [30], it would be able to model a wider spectrum of applications.

- Machine learning on belief logic programs. It would be both theoretically interesting and practically important to be able to learn belief logic programs from large data sets in data-rich application domains. Inductive logic programming [41, 43] can serve as a method for learning the rule structure. There are already works on extending probabilistic learning methods to belief functions, such as [7, 24], and it would be interesting to apply these methods to learning belief factors on blp rules.

- Reasoning with uncertainty in the Semantic Web. Many approaches have been proposed to extend description logic with probabilistic reasoning (e.g., [40]), fuzzy logic (e.g., [77]) and possibilistic logic (e.g., [61]), but there is little work on combining description logic with belief functions. This dissertation can form the basis for research in this direction.

Bibliography

[1] J. F. Baldwin. Support logic programming. *International Journal of Intelligent Systems*, 1:73–104, 1986.

[2] J. F. Baldwin. Evidential support logic programming. *Fuzzy Sets and Systems*, 24(1):1–26, 1987.

[3] U. Bergsten and J. Schubert. Dempster's rule for evidence ordered in a complete directed acyclic graph. *International Journal of Approximate Reasoning*, 9:37–73, 1993.

[4] H. A. Blair and V. S. Subrahmanian. Paraconsistent logic programming. In *7th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 340–360, London, UK, 1987. Springer-Verlag.

[5] H. A. Blair and V. S. Subrahmanian. Paraconsistent foundations for logic programming. *Journal of Non-Classical Logic*, 1988.

[6] H. A. Blair and V. S. Subrahmanian. Paraconsistent logic programming. *Theoretical Computer Science*, 68(2):135–154, 1989.

[7] E. Côme, L. Oukhellou, T. Denux, and P. Aknin. Learning from partially supervised data using mixture models and belief functions. *Pattern Recognition*, 42(3):334–348, 2009.

[8] G. A. Davis and J. Pei. Bayesian networks and traffic accident reconstruction. In *International Conference on Artificial Intelligence and Law*, pages 171–176, New York, NY, USA, 2003. ACM.

[9] A. Dekhtyar and V. S. Subrahmanian. Hybrid probabilistic programs. *J. of Logic Programming*, 43:391–405, 1997.

[10] A. P. Dempster. Upper and lower probabilities induced by a multi-valued mapping. *Ann. Mathematical Statistics*, 38, 1967.

[11] M. J. Druzdzel. SMILE: Structural modeling, inference, and learning engine and GeNIe: A development environment for graphical decision-theoretic models. In *National Conference on Artificial Intelligence (AAAI)*, pages 902–903, Orlando, FL, July 1999.

[12] D. Dubois, J. Lang, and H. Prade. Possibilistic logic. pages 439–513, 1994.

[13] D. Dubois and H. Prade. On the combination of evidence in various mathematical frameworks. In *Reliability Data Collection and Analysis*, pages 213–241. Kluwer Academic Publishers, Dordrecht, 1992.

[14] D. Dubois and H. Prade. Possibility theory, probability theory and multiple-valued logics: A clarification. *Annals of Mathematics and Artificial Intelligence*, 32(1-4):35–66, 2001.

[15] D. Dubois and H. Prade. A set-theoretic view of belief functions. In *Classic Works of the Dempster-Shafer Theory of Belief Functions*, pages 375–410. 2008.

[16] C. Elkan. The paradoxical success of fuzzy logic. In *IEEE Expert: Intelligent Systems and Their Applications*, pages 698–703, 1993.

[17] M. H. V. Emden. Quantitative deduction and its fixpoint theory. *Journal of Logic Programming*, 3(1):37–53, 1986.

[18] M. Fitting. Bilattices and the semantics of logic programming. *Journal of Logic Programming*, 11(1,2):91–116, 1991.

[19] A. V. Gelder, K. Ross, and J. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38:620–650, 1991.

[20] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *International Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[21] W. Gilks, S. Richardson, and D. Spiegelhalter. *Markov chain Monte Carlo Methods in Practice*. CRC Press, 1996.

[22] S. Glesner and D. Koller. Constructing flexible dynamic belief networks from first-order probabilistic knowledge bases. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 217–226, 1995.

[23] B. Grosof. A courteous compiler from generalized courteous logic programs to ordinary logic programs. Technical Report Supplementary Update Follow-On to RC 21472, IBM, July 1999.

[24] S. Haider. Belief functions based parameter and structure learning of bayesian networks in the presence of missing data. *International Journal of Hybrid Intelligent Systems*, 1(3-4):164–175, 2004.

[25] J. Y. Halpern. *Reasoning About Uncertainty*. MIT Press, 2003.

[26] Q. Han and Z. Q. Lin. Paraconsistent default reasoning. In J. P. Delgrande and T. Schaub, editors, *International Workshop on Non-Monotonic Reasoning*, pages 197–203, Whistler, Canada, June 2004.

[27] T. Inagaki. Interdependence between safety-control policy and multiple-sensor schemes via dempster-shafer theory. *IEEE Transactions on Reliability*, 40(2):182–188, 1991.

[28] M. Ishizuka and N. Kanai. Prolog-elf incorporating fuzzy logic. *New Generation Computing*, 3(4):479–486, 1985.

[29] K. Kersting and L. D. Raedt. Bayesian logic programs. Technical report, Albert-Ludwigs University at Freiburg, 2001.

[30] M. Kifer and A. Li. On the semantics of rule-based expert systems with uncertainty. In *International Conference on Database Theory (ICDT)*, pages 102–117, London, UK, 1988. Springer-Verlag.

[31] M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *Journal of Logic Programming*, 12(3,4):335–367, 1992.

[32] R. Kindermann and J. L. Snell. *Markov Random Fields and Their Applications*. American Mathematical Society, 1980.

[33] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall PTR, May 1995.

[34] R. A. Kowalski. Predicate logic as programming language. In *IFIP Congress*, pages 569–574, 1974.

[35] L. V. S. Lakshmanan and N. Shiri. A parametric approach to deductive databases with uncertainty. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):554–570, 2001.

[36] S. K. Lee. An extended relational database model for uncertain and imprecise information. In *International Conference on Very Large Data Bases (VLDB)*, pages 211–220, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.

[37] Z. Liu and H. Li. A probabilistic fuzzy logic system for modeling and control. *IEEE Transactions on Fuzzy Systems*, 13(6):848–859, 2005.

[38] T. Lukasiewicz. Probabilistic logic programming under inheritance with overriding. In *the Conference on Uncertainty in Artificial Intelligence*, pages 329–336, San Francisco, CA, 2001. Morgan Kaufmann Publishers.

[39] T. Lukasiewicz. Probabilistic logic programming with conditional constraints. *ACM Transactions on Computational Logic*, 2(3):289–339, 2001.

[40] T. Lukasiewicz. Probabilistic description logic programs. *International Journal of Approximate Reasoning*, 45(2):288–307, 2007.

[41] S. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

[42] S. Muggleton. Stochastic logic programs. In L. de Raedt, editor, *Advances in Inductive Logic Programming*, pages 254–264. IOS Press, 1996.

[43] S. Muggleton and L. de Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19:629–679, 1994.

[44] C. K. Murphy. Combining belief functions when evidence conflicts. *Decision Support Systems*, 29(1):1–9, 2000.

[45] R. T. Ng. Reasoning with uncertainty in deductive databases and logic programs. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 5(3):261–316, 1997.

[46] R. T. Ng and V. S. Subrahmanian. Relating dempster-shafer theory to stable semantics. In *International Symposium on Logic Programming*, pages 551–565, 1991.

[47] R. T. Ng and V. S. Subrahmanian. A semantical framework for supporting subjective and conditional probabilities in deductive databases. In K. Furukawa, editor, *International Conference on Logic Programming*, pages 565–580. The MIT Press, 1991.

[48] R. T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1992.

[49] R. T. Ng and V. S. Subrahmanian. A semantical framework for supporting subjective probabilities in deductive databases. *Journal of Automated Reasoning*, 10(2):191–235, 1993.

[50] D. Nute. Defeasible logic. In *Handbook of logic in artificial intelligence and logic programming*, pages 353–395. Oxford University Press, 1994.

[51] D. Pearce and G. Wagner. Logic programming with strong negation. In P. Schroeder-Heister, editor, *International Workshop on Extensions of Logic Programming*, pages 311–326. Springer, Berlin, Heidelberg, 1991.

[52] D. Pearce and G. Wagner. Logic programming with strong negation. In *International Workshop on Extensions of logic programming*, pages 311–326, New York, NY, USA, 1991. Springer-Verlag New York, Inc.

[53] J. Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *7th Conference of the Cognitive Science Society, University of California, Irvine*, pages 329–334, August 1985.

[54] J. Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288, 1986.

[55] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[56] D. Poole. Probabilistic horn abduction and bayesian networks. *Artificial Intelligence*, 64:81–129, 1993.

[57] D. Poole. The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94:7–56, 1997.

[58] D. Poole. The independent choice logic and beyond. In *Probabilistic Inductive Logic Programming*, pages 222–243, 2008.

[59] T. Przymusinski. Well-founded and stationary models of logic programs. *Annals of Mathematics and Artificial Intelligence*, 12:141–187, 1994.

[60] T. C. Przymusinski. Three-valued non-monotonic formalisms and semantics of logic programs. In *Logic Programming and Non-monotonic Reasoning*, pages 103–106, 1990.

[61] G. Qi, J. Z. Pan, and Q. Ji. Extending description logics with uncertainty reasoning in possibilistic logic. In *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 828–839, Berlin, Heidelberg, 2007. Springer-Verlag.

[62] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[63] L. D. Raedt and K. Kersting. Probabilistic inductive logic programming. In *Probabilistic Inductive Logic Programming*, pages 1–27, 2008.

[64] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[65] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.

[66] I. Ruthven and M. Lalmas. Using dempster-shafers theory of evidence to combine aspects of information use. *Journal of Intelligent Systems*, 19:267–301, 2002.

[67] T. Sato and Y. Kameya. Prism: A language for symbolic-statistical modeling. In *International Joint Conference on Artificial Intelligence*, pages 1330–1339, 1997.

[68] T. Sato and Y. Kameya. Parameter learning of logic programs for symbolic-statistical modeling. *Journal of Artificial Intelligence Research*, 15:391–454, 2001.

[69] T. Sato and N. Zhou. A new perspective of statistical modeling with PRISM. In L. Getoor and D. Jensen, editors, *Working Notes of the*

IJCAI-2003 Workshop on Learning Statistical Models from Relational Data, pages 133–140, Acapulco, Mexico, August 11, 2003.

[70] K. Sentz and S. Ferson. Combination of evidence in dempster-shafer theory. Technical report, Report SAND2002-0835, Sandia National Laboratories, 2002.

[71] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[72] E. Y. Shapiro. Logic programs with uncertainties: A tool for implementing rule-based systems. In *International Joint Conference on Artificial Intelligence*, pages 529–532, Karlsruhe, Germany, 1983.

[73] Y. Shen. Reasoning with recursive loops under the plp framework. *ACM Transactions on Computational Logic*, 9(4):1–31, 2008.

[74] P. P. Shenoy. Using dempster-shafer's belief-function theory in expert systems. In *Advances in the Dempster-Shafer theory of evidence*, pages 395–414. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[75] P. P. Shenoy and G. Shafer. Axioms for probability and belief-function propagation. In *Uncertainty in Artificial Intelligence*, pages 169–198. North-Holland, 1990.

[76] R. Stelzer, T. Proell, and R. I. John. Fuzzy logic control system for autonomous sailboats. In *IEEE International Conference on Fuzzy Systems*, pages 97–102, London, UK., July 2007.

[77] G. Stoilos, G. Stamou, J. Z. Pan, V. Tzouvaras, and I. Horrocks. Reasoning with very expressive fuzzy description logics. *Journal of Artificial Intelligence Research*, 30(1):273–320, 2007.

[78] V. S. Subrahmanian. On the semantics of quantitative logic programs. In *IEEE Symposium on Logic Programming*, pages 173–182, 1987.

[79] T. Swift and D. S. Warren. An abstract machine for slg resolution: Definite programs. In *International Symposium on Logic Programming*, pages 633–654, 1994.

[80] J. M. V. Novak, I. Perfilieva. *Mathematical principles of fuzzy logic*. Kluwer, Dordrecht, 1999.

[81] M. H. Van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of ACM*, 23(4):733–742, 1976.

[82] P. Vojtás. Fuzzy logic programming. *Fuzzy Sets and Systems*, 124(3):361–370, 2001.

[83] H. Wan. Belief logic programming with cyclic dependencies. In *Proceedings of 3rd International Conference on Web Reasoning and Rule Systems (RR)*, pages 150–165, 2009.

[84] H. Wan, B. Grosof, M. Kifer, P. Fodor, and S. Liang. Logic programming with defaults and argumentation theories. In *International Conference on Logic Programming*, pages 432–448, 2009.

[85] H. Wan and M. Kifer. Belief logic programming: Uncertainty reasoning with correlation of evidence. In *Proceedings of 10th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 316–328, 2009.

[86] H. Wan and M. Kifer. Query answering in belief logic programming. In *Proceedings of 3rd International Conference on Scalable Uncertainty Management (SUM)*, pages 268–281, 2009.

[87] H. Xu and P. Smets. Evidential reasoning with conditional belief functions. In *the Conference on Uncertainty in Artificial Intelligence*, pages 598–605, 1994.

[88] R. R. Yager. On the dempster-shafer framework and new combination rules. *Information Sciences*, 41(2):93–137, 1987.

[89] L. A. Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.

[90] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

[91] L. A. Zadeh. A review of (a mathematical theory of evidence. g. shafer, princeton university press, princeton, nj, 1976). *The AI Magazine*, 1984.

[92] L. A. Zadeh. Probability theory and fuzzy logic are complementary rather than competitive. *Technometrics*, 37(3):271–276, 1995.

[93] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 100(supp.):9–34, 1999.

[94] L. A. Zadeh. Toward a perception-based theory of probabilistic reasoning with imprecise probabilities. *Journal of Statistical Planning and Inference*, 105(1):233–26, 2002.

[95] L. Zhang. Representation, independence, and combination of evidence in the dempster-shafer theory. In *Advances in the Dempster-Shafer theory of evidence*, pages 51–69. John Wiley & Sons, Inc., New York, NY, USA, 1994.

Appendix A

Proofs

A.1 Proof of Theorem 3.1

Proof: Since there is no circular dependency among the atoms in blp, we can order all the atoms in $B_{\boldsymbol{P}}$: $b_1, ..., b_m, a_1, ..., a_n$, where $\boldsymbol{P}(b_l) = \varnothing, 1 \leq l \leq m$, and a_k depends only on atoms in $\{a_{k+1}, ..., a_n\} \cup \{b_1, ..., b_m\}$, $1 \leq k \leq n$.

Let $\mathbb{Y}$ be $\{b_1, ..., b_m\}$, $\mathbb{X}_k$ be $\{a_{k+1}, ..., a_n\}$, $0 \leq k < n$. From Definition 3.11, we have

$$\sum_{I \in \mathcal{TV}al(\boldsymbol{P})} \hat{m}_{\boldsymbol{P}}(I) = \sum_{I \in \mathcal{TV}al(\mathbb{Y} \cup \mathbb{X}_0)} \prod_{A \in \mathbb{Y}} s_{\boldsymbol{P}}(I, A) \cdot \prod_{A \in \mathbb{X}_0} s_{\boldsymbol{P}}(I, A) \tag{A.1}$$

By Definition 3.10, it follows that for any $I \in \mathcal{TV}al(\mathbb{Y} \cup \mathbb{X}_0)$, such that $\exists C \in \mathbb{Y}, I(C) \neq \mathbf{u}$, we have $\prod_{A \in \mathbb{Y}} s_{\boldsymbol{P}}(I, A) = 0$. Continuing with (A.1), we obtain

$$\begin{aligned} \sum_{I \in \mathcal{TV}al(\boldsymbol{P})} \hat{m}_{\boldsymbol{P}}(I) &= \sum_{I \in \mathcal{TV}al(\mathbb{Y} \cup \mathbb{X}_0) \text{ such that } I|_{\mathbb{Y}} = \mathbf{u}} \prod_{A \in \mathbb{X}_0} s_{\boldsymbol{P}}(I, A) \\ &= \sum_{I_0 \in \mathcal{TV}al(\mathbb{X}_0)} \prod_{A \in \mathbb{X}_0} s_{\boldsymbol{P}}(I'_0, A) \end{aligned}$$

where $I'_0 = I_0 \uplus \{\mathbb{Y} \to \mathbf{u}\}$. Continuing, we get

$$\begin{aligned}\sum_{I \in \mathcal{TV}al(\boldsymbol{P})} \hat{m}_{\boldsymbol{P}}(I) &= \sum_{I_0 \in \mathcal{TV}al(\mathbb{X}_0)} \Big(s_{\boldsymbol{P}}(I'_0, a_1) \cdot \prod_{A \in \mathbb{X}_1} s_{\boldsymbol{P}}(I'_0, A) \Big) \\ &= \sum_{I_1 \in \mathcal{TV}al(\mathbb{X}_1)} \Big(\; s_{\boldsymbol{P}}(I_1^{\mathbf{t}}, a_1) \cdot \prod_{A \in \mathbb{X}_1} s_{\boldsymbol{P}}(I_1^{\mathbf{t}}, A) \\ &\qquad + s_{\boldsymbol{P}}(I_1^{\mathbf{f}}, a_1) \cdot \prod_{A \in \mathbb{X}_1} s_{\boldsymbol{P}}(I_1^{\mathbf{f}}, A) \\ &\qquad + s_{\boldsymbol{P}}(I_1^{\mathbf{u}}, a_1) \cdot \prod_{A \in \mathbb{X}_1} s_{\boldsymbol{P}}(I_1^{\mathbf{u}}, A) \Big)\end{aligned} \tag{A.2}$$

where

$$\begin{aligned} I_0^{\mathbf{u}} &= I_0 \uplus \{\mathbb{Y} \to \mathbf{u}\} \\ I_1^{\mathbf{t}} &= I_1 \uplus \{\mathbb{Y} \to \mathbf{u}\} \uplus \{a_1 \to \mathbf{t}\} \\ I_1^{\mathbf{f}} &= I_1 \uplus \{\mathbb{Y} \to \mathbf{u}\} \uplus \{a_1 \to \mathbf{f}\} \\ I_1^{\mathbf{u}} &= I_1 \uplus \{\mathbb{Y} \to \mathbf{u}\} \uplus \{a_1 \to \mathbf{u}\} \end{aligned}$$

In (A.2), for any pair of truth values $\tau, \nu \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, $I_1^{\tau} \mid_{\mathbb{X}_1 \cup \mathbb{Y}}$ equals $I^{\nu} \mid_{\mathbb{X}_1 \cup \mathbb{Y}}$ (since they differ only on a_1), and the atoms in $\mathbb{X}_1 \cup \{a_1\}$ depend only on the atoms in $\mathbb{X}_1 \cup \mathbb{Y}$. Therefore, we obtain that $\forall A \in \mathbb{X}_1 \cup \{a_1\}, \boldsymbol{P}_{I_1^{\tau}}(A) = \boldsymbol{P}_{I_1^{\nu}}(A)$. Taking into account Definition 3.10, we can conclude that $\sum_{\nu \in \{\mathbf{t},\mathbf{f},\mathbf{u}\}} s_{\boldsymbol{P}}(I_1^{\nu}, a_1) = 1$ and $\forall \tau, \nu \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}, \forall A \in \mathbb{X}_1$, the following holds: $s_{\boldsymbol{P}}(I_1^{\tau}, A) = s_{\boldsymbol{P}}(I_1^{\nu}, A)$. Hence $\prod_{A \in \mathbb{X}_1} s_{\boldsymbol{P}}(I_1^{\tau}, A) = \prod_{A \in \mathbb{X}_1} s_{\boldsymbol{P}}(I_1^{\nu}, A)$. Continuing with (A.2) further, we get:

$$\sum_{I \in \mathcal{TV}al(\boldsymbol{P})} \hat{m}_{\boldsymbol{P}}(I) = \sum_{I_1 \in \mathcal{TV}al(\mathbb{X}_1)} \prod_{A \in \mathbb{X}_1} s_{\boldsymbol{P}}(I_1^{\mathbf{t}}, A) \tag{A.3}$$

Considering $a_2, ..., a_n$ one by one in a similar way, we conclude that for any $2 \leq k \leq n-1$

$$\sum_{I \in \mathcal{TV}al(\boldsymbol{P})} \hat{m}_{\boldsymbol{P}}(I) = \sum_{I_{k-1} \in \mathcal{TV}al(\mathbb{X}_{k-1})} \prod_{A \in \mathbb{X}_{k-1}} s_{\boldsymbol{P}}(I_{k-1}^{\mathbf{t}}, A)$$

where

$$I^{\mathbf{t}}_{k-1} = I_{k-1} \uplus \{\mathbb{Y} \to \mathbf{u}\} \uplus \{(\mathbb{X}_0 - \mathbb{X}_{k-1}) \to \mathbf{t}\}$$

Continuing, we have

$$\begin{aligned}\sum_{I \in \mathcal{TV}al(\boldsymbol{P})} \hat{m}_{\boldsymbol{P}}(I) \;\; = \;\; \sum_{I_k \in \mathcal{TV}al(\mathbb{X}_k)} \Big(& \prod_{A \in \mathbb{X}_k} s_{\boldsymbol{P}}(I^{\mathbf{t}}_k, A) \cdot s_{\boldsymbol{P}}(I^{\mathbf{t}}_k, a_k) \\ & + \prod_{A \in \mathbb{X}_k} s_{\boldsymbol{P}}(I^{\mathbf{f}}_k, A) \cdot s_{\boldsymbol{P}}(I^{\mathbf{f}}_k, a_k) \\ & + \prod_{A \in \mathbb{X}_k} s_{\boldsymbol{P}}(I^{\mathbf{u}}_k, A) \cdot s_{\boldsymbol{P}}(I^{\mathbf{u}}_k, a_k) \Big)\end{aligned} \tag{A.4}$$

where

$$\begin{aligned} I^{\mathbf{t}}_k &= I_k \uplus \{\mathbb{Y} \to \mathbf{u}\} \uplus \{\mathbb{X}_0 - \mathbb{X}_k \to \mathbf{t}\} \\ I^{\mathbf{f}}_k &= I_k \uplus \{\mathbb{Y} \to \mathbf{u}\} \uplus \{\mathbb{X}_0 - \mathbb{X}_k \to \mathbf{f}\} \\ I^{\mathbf{u}}_k &= I_k \uplus \{\mathbb{Y} \to \mathbf{u}\} \uplus \{\mathbb{X}_0 - \mathbb{X}_k \to \mathbf{u}\} \end{aligned}$$

Continuing further, we get:

$$\sum_{I \in \mathcal{TV}al(\boldsymbol{P})} \hat{m}_{\boldsymbol{P}}(I) = \sum_{I_k \in \mathcal{TV}al(\mathbb{X}_k)} \prod_{A \in \mathbb{X}_k} s_{\boldsymbol{P}}(I^{\mathbf{t}}_k, A) \tag{A.5}$$

Here (A.5) follows from (A.4) the same way as (A.3) follows from (A.2): $\sum_{\nu \in \{\mathbf{t},\mathbf{f},\mathbf{u}\}} s_{\boldsymbol{P}}(I^{\nu}_k, a_k) = 1$ and $\prod_{A \in \mathbb{X}_k} s_{\boldsymbol{P}}(I^{\tau}_k, A) = \prod_{A \in \mathbb{X}_k} s_{\boldsymbol{P}}(I^{\nu}_k, A)$ for any pair of truth values $\tau, \nu \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$.

Finally, we get

$$\sum_{I \in \mathcal{TV}al(\boldsymbol{P})} \hat{m}_{\boldsymbol{P}}(I) \;\; = \sum_{I_{n-1} \in \mathcal{TV}al(\{a_n\})} s_{\boldsymbol{P}}(I^{\mathbf{t}}_{n-1}, a_n) = 1$$

which completes the proof. □

A.2 Proof of Lemma 3.3

Proof: Since there is no circular dependency among the atoms in blp, we can order all the atoms in $B_{\boldsymbol{P}}$: $a_1, ..., a_n$, where a_k depends only on atoms in $\{a_1, ..., a_{k-1}\}$, $1 \leq k \leq n$. Without loss of generality, we assume $A = a_m$.

Let $\mathbb{X}_k$ be $\{a_1, ..., a_{k-1}\}$, $1 \leq k \leq n$. From Definition 3.11, we have

$$\sum_{I \in \mathcal{T}_{\mathcal{S}}} \hat{m}_{\boldsymbol{P}}(I) = \sum_{I \in \mathcal{T}_{\mathcal{S}}} \prod_{x \in B_{\boldsymbol{P}}} s_{\boldsymbol{P}}(I, x)$$

Note that for any rule R in $\boldsymbol{P}(A)$, $Body(R)$ only depends on atoms in $\mathbb{X}_{m-1}$. Let $\mathcal{T}_{1\mathcal{S}}$ be a set of truth valuations on $\mathbb{X}_{m-1}$:

$$\{I_1 \in \mathcal{TV}al(\mathbb{X}_{m-1}) \mid \forall R \in \mathcal{S}, I_1 \models Body(R), \forall R' \in \boldsymbol{P}(A) - \mathcal{S}, I_1 \not\models Body(R')\}$$

$$\begin{aligned}
&\sum_{I \in \mathcal{T}_{\mathcal{S}}} \hat{m}_{\boldsymbol{P}}(I) \\
&= \sum_{I' \in \mathcal{T}_{1\mathcal{S}}} \sum_{I \in \mathcal{TV}al(\{a_m, ..., a_n\})} \prod_{x \in B_{\boldsymbol{P}}} s_{\boldsymbol{P}}(I' \uplus I, x) \\
&= \sum_{I' \in \mathcal{T}_{1\mathcal{S}}} \sum_{I_{n-1} \in \mathcal{TV}al(\{a_m, ..., a_{n-1}\})} \Big(\prod_{i=1}^{n-1} s_{\boldsymbol{P}}(I_n^{\mathbf{t}}, a_i) \cdot s_{\boldsymbol{P}}(I_n^{\mathbf{t}}, a_n) \\
&\qquad + \prod_{i=1}^{n-1} s_{\boldsymbol{P}}(I_n^{\mathbf{f}}, a_i) \cdot s_{\boldsymbol{P}}(I_n^{\mathbf{f}}, a_n) \qquad \text{(A.6)} \\
&\qquad + \prod_{i=1}^{n-1} s_{\boldsymbol{P}}(I_n^{\mathbf{u}}, a_i) \cdot s_{\boldsymbol{P}}(I_n^{\mathbf{u}}, a_n) \Big)
\end{aligned}$$

where

$$I_n^{\mathbf{t}} = I' \uplus I_{n-1} \uplus \{a_n \to \mathbf{t}\}$$
$$I_n^{\mathbf{f}} = I' \uplus I_{n-1} \uplus \{a_n \to \mathbf{f}\}$$
$$I_n^{\mathbf{u}} = I' \uplus I_{n-1} \uplus \{a_n \to \mathbf{u}\}$$

Since for any $1 \leq i \leq n$, for any pair of truth value $\tau, \nu \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, $\boldsymbol{P}_{I_i^\tau}(a_i) = \boldsymbol{P}_{I_i^\nu}(a_i)$. Also for any $1 \leq i \leq n-1$, $I_i^\tau(a_i) = I_i^\nu(a_i)$. According to Definition 3.10 we have $\sum_{\nu \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}} s_{\boldsymbol{P}}(I_n^\nu, a_n) = 1$, and $\forall \tau, \nu \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}, \forall 1 \leq i \leq n-1$, the following holds: $s_{\boldsymbol{P}}(I_n^\tau, a_i) = s_{\boldsymbol{P}}(I_n^\nu, a_i)$. Continuing with (A.6), we get

$$\sum_{I \in \mathcal{T}_{\mathcal{S}}} \hat{m}_{\boldsymbol{P}}(I) = \sum_{I' \in \mathcal{T}_{1\mathcal{S}}} \sum_{I_{n-1} \in \mathcal{TV}al(\{a_m, ..., a_{n-1}\})} \prod_{i=1}^{n-1} s_{\boldsymbol{P}}(I_n^{\mathbf{t}}, a_i)$$

Considering $a_{n-1}, ..., a_{m+1}$ one by one in a similar way, we conclude that

$$\sum_{I \in \mathcal{T}_{\mathcal{S}}} \hat{m}_{\boldsymbol{P}}(I) = \sum_{I' \in \mathcal{T}_{1\mathcal{S}}} \sum_{I_m \in \mathcal{TV}al(\{a_m\})} \prod_{i=1}^{m} s_{\boldsymbol{P}}(I^{\mathbf{t}}_{m+1}, a_i) \tag{A.7}$$

where

$$I^{\mathbf{t}}_{m+1} = I' \uplus I_m \uplus \{\{a_{m+1}, ..., a_n\} \to \mathbf{t}\}$$

Continuing, we get

$$\begin{aligned}
\sum_{I \in \mathcal{T}_{\mathcal{S}}} \hat{m}_{\boldsymbol{P}}(I) &= \sum_{I' \in \mathcal{T}_{1\mathcal{S}}} \Big(\prod_{i=1}^{m-1} s_{\boldsymbol{P}}(I^{\mathbf{t}}_m, a_i) \cdot s_{\boldsymbol{P}}(I^{\mathbf{t}}_m, A) \\
&\qquad + \prod_{i=1}^{m-1} s_{\boldsymbol{P}}(I^{\mathbf{f}}_m, a_i) \cdot s_{\boldsymbol{P}}(I^{\mathbf{f}}_m, A) \\
&\qquad + \prod_{i=1}^{m-1} s_{\boldsymbol{P}}(I^{\mathbf{u}}_m, a_i) \cdot s_{\boldsymbol{P}}(I^{\mathbf{u}}_m, A) \Big) \\
&= \sum_{I' \in \mathcal{T}_{1\mathcal{S}}} \prod_{i=1}^{m-1} s_{\boldsymbol{P}}(I^{\mathbf{t}}_m, a_i)
\end{aligned} \tag{A.8}$$

where

$$\begin{aligned}
I^{\mathbf{t}}_m &= I' \uplus \{A \to \mathbf{t}\} \uplus \{\{a_{m+1}, ..., a_n\} \to \mathbf{t}\} \\
I^{\mathbf{f}}_m &= I' \uplus \{A \to \mathbf{f}\} \uplus \{\{a_{m+1}, ..., a_n\} \to \mathbf{t}\} \\
I^{\mathbf{u}}_m &= I' \uplus \{A \to \mathbf{u}\} \uplus \{\{a_{m+1}, ..., a_n\} \to \mathbf{t}\}
\end{aligned}$$

Similar to (A.7), we get

$$\begin{aligned}
\sum_{I \in \mathcal{T}_{\mathcal{S}}, I \models A} \hat{m}_{\boldsymbol{P}}(I) &= \sum_{I' \in \mathcal{T}_{1\mathcal{S}}} \sum_{J_m = \{A \to \mathbf{t}\}} \prod_{i=1}^{m} s_{\boldsymbol{P}}(J^{\mathbf{t}}_{m+1}, a_i) \\
&= \sum_{I' \in \mathcal{T}_{1\mathcal{S}}} \Big(s_{\boldsymbol{P}}(J^{\mathbf{t}}_{m+1}, A) \cdot \prod_{i=1}^{m-1} s_{\boldsymbol{P}}(J^{\mathbf{t}}_{m+1}, a_i) \Big)
\end{aligned}$$

where

$$J^{\mathbf{t}}_{m+1} = I' \uplus J_m \uplus \{\{a_{m+1}, ..., a_n\} \to \mathbf{t}\}$$

Note that $s_{\boldsymbol{P}}(J^{\mathbf{t}}_{m+1}, A) = v$, so

$$\sum_{I \in \mathcal{T}_{\mathcal{S}}, I \models A} \hat{m}_{\boldsymbol{P}}(I) = \sum_{I' \in \mathcal{T}_{1\mathcal{S}}} v \cdot \prod_{i=1}^{m-1} s_{\boldsymbol{P}}(J^{\mathbf{t}}_{m+1}, a_i) \tag{A.9}$$

Considering (A.8) and (A.9), we can get (3.1).

Similarly we can prove (3.2). □

A.3 Proof of Theorem 3.2

Proof: (We write $\texttt{model}_{\boldsymbol{P}}$ as model for simplicity.)

For any truth valuation I such that $\hat{m}_{\boldsymbol{P}}(I) > 0$ and $I \models \bigwedge_{R\in\mathcal{S}} Body(R)$, it is clear that $\forall R \in \mathcal{S}, I \models Body(R)$. Now we will prove that $\forall R' \in \boldsymbol{P}(A) - \mathcal{S}, I \not\models Body(R')$. Indeed, if there is a rule R' in $\boldsymbol{P}(A) - \mathcal{S}$ such that $I \models Body(R')$, then for $\mathcal{S}' = \mathcal{S} \cup \{R'\} \supset \mathcal{S}$, $I \models \bigwedge_{R\in\mathcal{S}'} Body(R)$. Since $\hat{m}_{\boldsymbol{P}}(I) > 0$, we get $\texttt{model}(\bigwedge_{R\in\mathcal{S}'} Body(R)) > 0$, this violates condition (ii) for $\mathcal{S}$. Thus, $\{I \mid \hat{m}_{\boldsymbol{P}}(I) > 0 \text{ and } I \models \bigwedge_{R\in\mathcal{S}} Body(R)\} \subseteq \mathcal{T}_{\mathcal{S}} \cap \{I \mid \hat{m}_{\boldsymbol{P}}(I) > 0\}$, where $\mathcal{T}$ is as defined in Lemma 3.3. It is easy to see that $\{I \mid \hat{m}_{\boldsymbol{P}}(I) > 0 \text{ and } I \models \bigwedge_{R\in\mathcal{S}} Body(R)\} \supseteq \mathcal{T}_{\mathcal{S}} \cap \{I \mid \hat{m}_{\boldsymbol{P}}(I) > 0\}$.

According to Definition 3.12, we have

$$\begin{aligned}
\texttt{model}\big(\bigwedge_{R\in\mathcal{S}} Body(R)\big) &= \sum_{I\mid \hat{m}_{\boldsymbol{P}}(I)>0 \text{ and } I\models\bigwedge_{R\in\mathcal{S}} Body(R)} \hat{m}_{\boldsymbol{P}}(I) \\
&= \sum_{I\mid \hat{m}_{\boldsymbol{P}}(I)>0 \text{ and } I\in\mathcal{T}_{\mathcal{S}}} \hat{m}_{\boldsymbol{P}}(I) \\
\texttt{model}\big(A \wedge \bigwedge_{R\in\mathcal{S}} Body(R)\big) &= \sum_{I\mid \hat{m}_{\boldsymbol{P}}(I)>0 \text{ and } I\models A, I\models\bigwedge_{R\in\mathcal{S}} Body(R)} \hat{m}_{\boldsymbol{P}}(I) \\
&= \sum_{I\mid \hat{m}_{\boldsymbol{P}}(I)>0 \text{ and } I\models A, I\in\mathcal{T}_{\mathcal{S}}} \hat{m}_{\boldsymbol{P}}(I)
\end{aligned}$$

Now according to Lemma 3.3,

$$\sum_{I\mid \hat{m}_{\boldsymbol{P}}(I)>0 \text{ and } I\models A, I\in\mathcal{T}_{\mathcal{S}}} \hat{m}_{\boldsymbol{P}}(I) = v \cdot \sum_{I\mid \hat{m}_{\boldsymbol{P}}(I)>0 \text{ and } I\in\mathcal{T}_{\mathcal{S}}} \hat{m}_{\boldsymbol{P}}(I)$$

We proved

$$\frac{\texttt{model}\big(A \wedge \bigwedge_{R\in\mathcal{S}} Body(R)\big)}{\texttt{model}\big(\bigwedge_{R\in\mathcal{S}} Body(R)\big)} = v$$

The other part of the theorem can be proved similarly. □

A.4 Proof of Theorem 3.6

Proof: First, since every combination function is associative and commutative, we have that: for any multiset $\mathcal{S}$ that is composed of $[0,0]$, $\Phi_1(\mathcal{S}) = [0,0]$; for any multiset $\mathcal{S}$ that is composed of $[1,1]$, $\Phi_1(\mathcal{S}) = [1,1]$; for any multiset $\mathcal{S}$ that is composed of $[0,0]$ and $[1,1]$, $\Phi_1(\mathcal{S}) = [0,1]$.

Since Π is acyclic, there is an ordering of atoms in B_Π, $A_1, ..., A_n$, such that A_k only depends on atoms in $\{A_1, ..., A_{k-1}\}$.

We extend Definition 3.9 to LPDA program Π. Also let $J : B_{\boldsymbol{P}} \longrightarrow \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$ be Π's model under the LPDA semantics and denote $\lambda(\Pi)$ with $\boldsymbol{P}$.

For any $1 \leq k \leq n$, we have

- $J(A_k) = t$
  $\Rightarrow$ $\Pi_J(A_k) \supset \varnothing$, and $\Pi_J(\overline{A_k}) = \varnothing$
  $\Rightarrow$ $\boldsymbol{P}_J(A_k) \supset \varnothing$, and the belief factor of every rule in $\boldsymbol{P}_J(A_k)$ is $[1,1]$
  $\Rightarrow$ $s_{\boldsymbol{P}}(J, A_k) = 1$

- $J(A_k) = f$
  $\Rightarrow$ $\Pi_J(A_k) = \varnothing$, and $\Pi_J(\overline{A_k}) \supset \varnothing$
  $\Rightarrow$ $\boldsymbol{P}_J(A_k) \supset \varnothing$, and the belief factor of every rule in $\boldsymbol{P}_J(A_k)$ is $[0,0]$
  $\Rightarrow$ $s_{\boldsymbol{P}}(J, A_k) = 1$

- $J(A_k) = u$
  $\Rightarrow$ $\Pi_J(A_k) \supset \varnothing$, and $\Pi_J(\overline{A_k}) \supset \varnothing$
  $\Rightarrow$ $\boldsymbol{P}_J(A_k) \supset \varnothing$, there is a rule in $\boldsymbol{P}_J(A_k)$ with belief factor $[0,0]$, there is also a rule in $\boldsymbol{P}_J(A_k)$ with belief factor $[1,1]$
  $\Rightarrow$ $s_{\boldsymbol{P}}(J, A_k) = 1$

So $\hat{m}_{\boldsymbol{P}}(J) = \prod_{k=1}^{n} s_{\boldsymbol{P}}(J, A_k) = 1$

Thus, we also know that for any $I \neq J$, $\hat{m}_{\boldsymbol{P}}(I) = 0$.

Now we can infer:

$$\begin{aligned} \Pi \models_{LPDA} F &\iff J \models F \\ &\iff \sum_{I \in \mathcal{TV}al(B_{\boldsymbol{P}}), I \models F} \hat{m}_{\boldsymbol{P}}(I) = \hat{m}_{\boldsymbol{P}}(J) = 1 \\ &\iff bel_{\boldsymbol{P}}(F) = 1 \end{aligned}$$

The theorem is proved. □

A.5 Proof of Theorem 3.7

Proof: Let α_0 be $\varnothing$ and $\alpha_{k+1} = \alpha_k \cup dep(\alpha_k)$, $0 \leq k$. Since $m_0 = m_\varnothing$ and $m_k = \hat{T}_{\boldsymbol{P}}^{\uparrow k}(m_0)$, by the definition of $\hat{T}_{\boldsymbol{P}}$, α_k is the base of m_k, $0 \leq k$. Let ω be $min\{k \mid \alpha_{k+1} = \alpha_k\}$, it is not hard to see that $\alpha_\omega = B_{\boldsymbol{P}}$, and we will prove $m_\omega = \hat{m}_{\boldsymbol{P}}$.

Let $\beta_k = \alpha_k - \alpha_{k-1} = dep(\alpha_{k-1})$, $1 \leq k$, we can see that $B_{\boldsymbol{P}}$ is partitioned into $\beta_1, ..., \beta_\omega$. Let I_0 be $I_\varnothing$, which is the only truth valuation over $\varnothing$. By Definition 3.16, for every $I_1 \in \mathcal{TV}al(\beta_1)$,..., $I_\omega \in \mathcal{TV}al(\beta_\omega)$,

$$\hat{T}_{\boldsymbol{P}}^{\uparrow k}(m_0)(\biguplus_{1 \leq k \leq \omega} I_k) = \prod_{1 \leq k \leq \omega} \prod_{X \in \beta_k} Val\Big(\Phi_X\big(BF(X, \boldsymbol{P}, I_{k-1})\big), I_k(X)\Big)$$

That is to say, for every $I \in \mathcal{TV}al(\boldsymbol{P}) = \mathcal{TV}al\big(\bigcup_{1 \leq k \leq \omega} \beta_k\big)$,

$$\hat{T}_{\boldsymbol{P}}^{\uparrow k}(m_0)(I) = \prod_{1 \leq k \leq \omega} \prod_{X \in \beta_k} Val\Big(\Phi_X\big(BF(X, \boldsymbol{P}, I \mid_{\beta_{k-1}})\big), I \mid_{\beta_k} (X)\Big) \qquad \text{(A.10)}$$

By Definition 3.10 and Definition 3.16, we have

$$Val\Big(\Phi_X\big(BF(X, \boldsymbol{P}, I \mid_{\beta_{k-1}})\big), I \mid_{\beta_k} (X)\Big) = s_{\boldsymbol{P}}(I, X)$$

Continuing with (A.10) we can conclude that

$$\begin{aligned} m_\omega(I) = &\ \hat{T}_{\boldsymbol{P}}^{\uparrow k}(m_0)(I) \\ = &\ \prod_{1\leq k\leq\omega}\prod_{X\in\beta_k} s_{\boldsymbol{P}}(I,X) \\ = &\ \prod_{X\in B_{\boldsymbol{P}}} s_{\boldsymbol{P}}(I,X) \\ = &\ \hat{m}_{\boldsymbol{P}}(I) \end{aligned}$$

□

A.6 Proof of Theorem 4.3

Proof: Suppose $a_1, ..., a_n$ are the labels for the nodes in the dependency DAG of $\boldsymbol{P}$. Note that a_i can be associated with either an a-node or an r-node. Without loss of generality, suppose the traversal order in the bottom-up algorithm is $\langle a_1, ..., a_n\rangle$. Since it is a post-order traversal, we know that for all $1 \leq k \leq n$, a_k depends only on the atoms in $\{a_1, ..., a_{k-1}\}$.

Let $\mathbb{X}_0$ be $\varnothing$ and $\mathbb{X}_k$ be $\{a_1, ..., a_k\}$, $1 \leq k \leq n$. Since we are considering only ground rules, $\mathbb{X}_n = B_{\boldsymbol{P}} \cup R$, where R is the set of propositions that label rules. Also let $m_{\mathbb{X}_k}$, $1 \leq k \leq n$, be the support function created during the traversal.

Let $I \in \mathcal{TV}al(\mathbb{X}_n)$ be a truth valuation. During the process of expansion of the support function from $m_{\mathbb{X}_0}$ to $m_{\mathbb{X}_1}$ to ... to $m_{\mathbb{X}_n}$ in the algorithm, we see that, for any $1 \leq k \leq n$:

- If $a_k \in B_{\boldsymbol{P}}$, then

$$m_{\mathbb{X}_k}(I\mid_{\mathbb{X}_k}) \;=\; m_{\mathbb{X}_{k-1}}(I\mid_{\mathbb{X}_{k-1}}) \;\cdot\; s_{\boldsymbol{P}}(I, a_k)$$

- If $a_k \in R$, then

$$m_{\mathbb{X}_k}(I\mid_{\mathbb{X}_k}) \;=\; m_{\mathbb{X}_{k-1}}(I\mid_{\mathbb{X}_{k-1}}) \;\cdot\; 1, \text{ if } I(a_k) = t$$

$$m_{\mathbb{X}_k}(I\mid_{\mathbb{X}_k}) \;=\; m_{\mathbb{X}_{k-1}}(I\mid_{\mathbb{X}_{k-1}}) \;\cdot\; 0, \text{ if } I(a_k) \neq t$$

From the above, we can conclude the following for any $I' \in \mathcal{TV}al(\boldsymbol{P})$ and $I \in \mathcal{TV}al(\mathbb{X}_n)$ such that $I \mid_{B_{\boldsymbol{P}}} = I'$:

1. If $\exists c \in R,\ I(c) \neq t$, then
$$m_{\mathbb{X}_n}(I) = 0$$

2. If $I = I' \uplus \{R \rightarrow t\}$, then
$$m_{\mathbb{X}_n}(I) = \prod_{1 \leq k \leq n} \delta_k$$
where $\delta_k = s_{\boldsymbol{P}}(I, a_k)$ if $a_k \in B_{\boldsymbol{P}}$ and $\delta_k = 1$ if $a_k \in R$. Therefore,
$$m_{\mathbb{X}_n}(I) = \textstyle\prod_{a \in B_{\boldsymbol{P}}} s_{\boldsymbol{P}}(I', a) = \hat{m}_{\boldsymbol{P}}(I')$$

Thus, for any given $I' \in \mathcal{TV}al(\boldsymbol{P})$, $m_{\mathbb{X}_n}(I)$ is non-zero for only one $I \in \mathcal{TV}al(\mathbb{X}_n)$ such that $I \mid_{B_{\boldsymbol{P}}} = I'$. Summing up $m_{\mathbb{X}_n}(I)$ for all such I's, we obtain
$$\sum_{I \in \mathcal{TV}al(\mathbb{X}_n) \text{ where } I\mid_{B_{\boldsymbol{P}}} = I'} m_{\mathbb{X}_n}(I) = \hat{m}_{\boldsymbol{P}}(I')$$
By Definition 4.3, $m_{\mathbb{X}_n}$ is an expansion of $\hat{m}_{\boldsymbol{P}}$. Theorem 4.3 then follows from Lemma 4.1 and Lemma 4.2. □

A.7 Proof of Theorem 4.4

Proof: let m be the outcome of Algorithm 2 and let m' be the outcome of Algorithm 1. We will prove that m is a projection of m' on $\{g\}$. The result then follows by Lemma 4.2, Theorem 4.3 and Definition 3.12.

Suppose $\langle a_1, ..., a_d \rangle$ is a traversal order used by Algorithm 2. Then there must exist a traversal order $\langle b_1, ..., b_n \rangle$ used by Algorithm 1, where there must be a sequence $1 = i_1 < ... < i_d \leq n$ such that $b_{i_k} = a_k$, for all $1 \leq k \leq d$.

Let m_0 be $m_\varnothing$, m_k, $1 \leq k \leq d$, be the support function constructed by Algorithm 2 after visiting a_k, and $\mathcal{S}_k$ be the base of m_k. Similarly, let m'_0 be $m_\varnothing$, m'_j, $1 \leq j \leq n$, be the support function constructed by Algorithm 1 after visiting b_j, and $\mathcal{C}_j$ be the base of m'_j.

We will prove that m_k is a projection of m'_{i_k} for all $1 \leq k \leq d$, by induction on k.

The base case of the induction, $k = 0$, is obvious: both m_0 and m'_0 are $m_\varnothing$.

For the inductive step, we will show that *if* m_{k-1} *is a projection of* $m'_{i_{(k-1)}}$ *then* m_k *is a projection of* m'_{i_k}.

In Algorithm 2, when visiting a_k, we expand m_{k-1} with base $\mathcal{S}_{k-1}$ to a support function, mx_k, with base $\mathcal{S}_{k-1} \cup \{a_k\}$ and then project it to obtain m_k. Since $a_k = b_{i_k}$, m_{k-1} is expanded to mx_k in exactly the same way as m'_{i_k-1} is expanded to m'_{i_k} in Algorithm 1.

For any $I \in \mathcal{TV}al(\mathcal{S}_{k-1})$, let $I'_1 = I \uplus \{a_k \to t\}$, $I'_2 = I \uplus \{a_k \to f\}$, $I'_3 = I \uplus \{a_k \to u\}$, where the constant truth valuations of the form $\{a_k \to ...\}$ were introduced right below Definition 3.6. The construction used in Algorithm 2 establishes that

$$mx_k(I'_j) = m_{k-1}(I) \cdot V_j \tag{A.11}$$

for some value V_j. Similarly, for any $J \in \mathcal{TV}al(\mathcal{C}_{i_k-1})$, let $J'_1 = J \uplus \{a_k \to t\}$, $J'_2 = J \uplus \{a_k \to f\}$, $J'_3 = J \uplus \{a_k \to u\}$.[1] The construction in Algorithm 1 similarly establishes that

$$m'_{i_k}(J'_j) = m'_{i_k-1}(J) \cdot V_j \tag{A.12}$$

for some V_j and, moreover, it is the same V_j as in (A.11).

Since $i_{k-1} \leq i_k - 1$, it follows by construction that $m'_{i_{(k-1)}}$ is a projection of m'_{i_k-1}. Since also m_{k-1} is a projection of $m'_{i_{(k-1)}}$ by the inductive assumption,

[1] Recall that $a_k = b_{i_k}$ and $J'_j \in \mathcal{TV}al(\mathcal{C}_{i_k})$.

Lemma 4.1 assures that m_{k-1} is a projection of m'_{i_k-1}. By Definition 4.3 and (A.11) we have:

$$\begin{aligned} mx_k(I'_j) &= m_{k-1}(I) \cdot V_j \\ &= V_j \cdot \sum_{\substack{J \in \mathcal{TV}al(\mathcal{C}_{i_k-1}) \\ J|_{S_{k-1}} = I}} m'_{i_k-1}(J) \end{aligned}$$

Combining with (A.12), we get

$$\begin{aligned} mx_k(I'_j) &= \sum_{\substack{J'_j \in \mathcal{TV}al(\mathcal{C}_{i_k}) \\ J'_j|_{S_{k-1}} = I \\ J'_j(a_k) = I'_j(a_k)}} m'_{i_k}(J'_j) \\ &= \sum_{\substack{J'_j \in \mathcal{TV}al(\mathcal{C}_{i_k}) \\ J'_j|_{S_k} = I'_j}} m'_{i_k}(J'_j) \end{aligned}$$

The above equation is an instance of the condition in Definition 4.3, which establishes mx_k as a projection of m'_{i_k}. Since m_k is constructed out of mx_k by projecting mx_k to a smaller base, m_k is a projection of mx_k. By transitivity (Lemma 4.1), m_k is a projection of m'_{i_k}, which completes the induction.

For $k = d$, we get that m_d is a projection of m'_{i_d} on S_d which is $\{g\}$. Since m'_{i_d} is a projection of m'_n, we know that m_d is a projection of m'_n on $\{g\}$. It remains to recall that m_d is m and m'_n is m'. So, m is a projection of m' on $\{g\}$. □

A.8 Proof of Theorem 5.2

Proof: According to Definition 5.6, for any $I \in \mathcal{TV}al(\boldsymbol{P})$,

$$\widetilde{m}_{\boldsymbol{P}}(I) = \sum_{I' \in \mathcal{TV}al(acyclic(\boldsymbol{P})), I'|_{B_{\boldsymbol{P}}} = I} \hat{m}_{acyclic(\boldsymbol{P})}(I') \tag{A.13}$$

and $\texttt{cmodel}_{\boldsymbol{P}}(F) = \texttt{model}_{acyclic(\boldsymbol{P})}(F)$.

Consider any pair of $I \in \mathcal{TV}al(\boldsymbol{P})$ and $I' \in \mathcal{TV}al(acyclic(\boldsymbol{P}))$ so that $I' \mid_{B_{\boldsymbol{P}}} = I$, by definition, $\hat{m}_{\boldsymbol{P}}(I) = \prod_{X \in B_{\boldsymbol{P}}} s_{\boldsymbol{P}}(I, X)$, and

$$\begin{aligned} & \hat{m}_{acyclic(\boldsymbol{P})}(I') \\ = & \prod_{X \in B_{acyclic(\boldsymbol{P})}} s_{acyclic(\boldsymbol{P})}(I', X) \\ = & \prod_{X \in B_{\boldsymbol{P}}} s_{acyclic(\boldsymbol{P})}(I', X) \cdot \prod_{X \in B_{acyclic(\boldsymbol{P})} - B_{\boldsymbol{P}}} s_{acyclic(\boldsymbol{P})}(I', X) \end{aligned}$$

According to Lemma 5.1, for every $X \in B_{\boldsymbol{P}}$, $s_{acyclic(\boldsymbol{P})}(I', X) = s_{\boldsymbol{P}}(I, X)$, so

$$\begin{aligned} & \hat{m}_{acyclic(\boldsymbol{P})}(I') \\ = & \prod_{X \in B_{\boldsymbol{P}}} s_{\boldsymbol{P}}(I, X) \cdot \prod_{X \in B_{acyclic(\boldsymbol{P})} - B_{\boldsymbol{P}}} s_{acyclic(\boldsymbol{P})}(I', X) \\ = & \hat{m}_{\boldsymbol{P}}(I) \cdot \prod_{X \in B_{acyclic(\boldsymbol{P})} - B_{\boldsymbol{P}}} s_{acyclic(\boldsymbol{P})}(I', X) \end{aligned}$$

Let $B_{acyclic(\boldsymbol{P})} - B_{\boldsymbol{P}}$ be $\{X_1, ..., X_n\}$. For any $I \in \mathcal{TV}al(\boldsymbol{P})$,

$$\begin{aligned} & \sum_{I' \in \mathcal{TV}al(acyclic(\boldsymbol{P})), I'|_{B_{\boldsymbol{P}}} = I} \hat{m}_{acyclic(\boldsymbol{P})}(I') \\ = & \sum_{I' \in \mathcal{TV}al(acyclic(\boldsymbol{P})), I'|_{B_{\boldsymbol{P}}} = I} \hat{m}_{\boldsymbol{P}}(I) \cdot \prod_{X \in B_{acyclic(\boldsymbol{P})} - B_{\boldsymbol{P}}} s_{acyclic(\boldsymbol{P})}(I', X) \\ = & \hat{m}_{\boldsymbol{P}}(I) \cdot \sum_{I' \in \mathcal{TV}al(acyclic(\boldsymbol{P})), I'|_{B_{\boldsymbol{P}}} = I} \Big(\prod_{1 \le i \le n} s_{acyclic(\boldsymbol{P})}(I', X_i) \Big) \end{aligned} \tag{A.14}$$

We can prove $\sum_{I' \in \mathcal{TV}al(acyclic(\boldsymbol{P})), I'|_{B_{\boldsymbol{P}}} = I} \prod_{1 \le i \le n} s_{acyclic(\boldsymbol{P})}(I', X_i) = 1$ similarly to the proof to Theorem 3.1. Continue with (A.14), for any $I \in \mathcal{TV}al(\boldsymbol{P})$, we have

$$\sum_{I' \in \mathcal{TV}al(acyclic(\boldsymbol{P})), I'|_{B_{\boldsymbol{P}}} = I} \hat{m}_{acyclic(\boldsymbol{P})}(I') = \hat{m}_{\boldsymbol{P}}(I) \tag{A.15}$$

Together with (A.13), we get $\widetilde{m}_{\boldsymbol{P}}(I) = \hat{m}_{\boldsymbol{P}}(I)$.

For any $F \in \mathcal{B}ool(B_{\boldsymbol{P}})$,

$$\begin{aligned}
& \texttt{cmodel}_{\boldsymbol{P}}(F) \\
= \; & \texttt{model}_{acyclic(\boldsymbol{P})}(F) \\
= \; & \sum_{I' \in \mathcal{TV}al(acyclic(\boldsymbol{P})), I' \models F} \hat{m}_{acyclic(\boldsymbol{P})}(I') \\
= \; & \sum_{I' \in \mathcal{TV}al(acyclic(\boldsymbol{P})), I'|B_{\boldsymbol{P}}=I, I \models F} \hat{m}_{acyclic(\boldsymbol{P})}(I') \\
= \; & \sum_{I \in \mathcal{TV}al(\boldsymbol{P}), I \models F} \Big(\sum_{I' \in \mathcal{TV}al(acyclic(\boldsymbol{P})), I'|B_{\boldsymbol{P}}=I} \hat{m}_{acyclic(\boldsymbol{P})}(I') \Big) \qquad \text{(A.16)}
\end{aligned}$$

Recall that we already proved (A.15), continue with (A.16),

$$\begin{aligned}
\texttt{cmodel}_{\boldsymbol{P}}(F) &= \sum_{I \in \mathcal{TV}al(\boldsymbol{P}), I \models F} \hat{m}_{\boldsymbol{P}}(I) \\
&= \texttt{model}_{\boldsymbol{P}}(F)
\end{aligned}$$

□

A.9 Proof of Theorem 5.3

Proof: We only need to prove

$$\texttt{model}_{acyclic(\boldsymbol{P})}(A) = \texttt{model}_{acyclic(\boldsymbol{P}'_A)}(A)$$

Recall that, according to Theorem 4.4, if we apply Algorithm 2 on the proof DAG of $acyclic(\boldsymbol{P})$ for A, the result is $\texttt{model}_{acyclic(\boldsymbol{P})}(A)$, if we apply Algorithm 2 on the proof DAG of $acyclic(\boldsymbol{P}'_A)$ for A, the result is $\texttt{model}_{acyclic(\boldsymbol{P}'_A)}(A)$. That is to say, this theorem is proved if we can prove that the proof DAG of $acyclic(\boldsymbol{P})$ for A is the same as the proof DAG of $acyclic(\boldsymbol{P}'_A)$ for A.

Indeed, according to the definition of decyclification, $acyclic(\boldsymbol{P}) \supset acyclic(\boldsymbol{P}'_A)$, and every rule r_0 in $acyclic(\boldsymbol{P}) - acyclic(\boldsymbol{P}'_A)$ satisfies at least one of the following conditions:

1. There is a rule $R \in \boldsymbol{P}$ with A in body, of the form

$$[v, w]\ A_0\ \texttt{:-}\ A_1, \ldots, A_k, \overline{A_{k+1}}, \ldots, \overline{A_n}, D_1, \ldots, D_l, \overline{D_{l+1}}, \ldots, \overline{D_m}.$$

where $A_i \in clique(A_0), 1 \leq i \leq n$, $D_j \notin clique(A_0), 1 \leq j \leq m$, $A \in \{A_1, ...A_n, D_1, ..., D_m\}$, such that r_0 is the rule

$$[v, w]\ A_0\ \texttt{:-}\ ID_R$$

or a rule of the form

$$[v, w]\ A_0^{\mathcal{G}_0}\ \texttt{:-}\ A_1^{\mathcal{G}_1}, \ldots, A_k^{\mathcal{G}_k}, \overline{A_{k+1}^{\mathcal{G}_{k+1}}}, \ldots, \overline{A_n^{\mathcal{G}_n}}, D_1, \ldots, D_l, \overline{D_{l+1}}, \ldots, \overline{D_m}$$

or

$$[1, 1]\ ID_R\ \texttt{:-}\ A_1^{\mathcal{G}_1}, \ldots, A_k^{\mathcal{G}_k}, \overline{A_{k+1}^{\mathcal{G}_{k+1}}}, \ldots, \overline{A_n^{\mathcal{G}_n}}, D_1, \ldots, D_l, \overline{D_{l+1}}, \ldots, \overline{D_m}$$

2. There is a rule $R' \in \boldsymbol{P}$ with A in body, such that r_0 is a rule of the form

$$[v, w]\ A_0^{\mathcal{G}_0}\ \texttt{:-}\ A_1^{\mathcal{G}_1}, \ldots, A_k^{\mathcal{G}_k}, \overline{A_{k+1}^{\mathcal{G}_{k+1}}}, \ldots, \overline{A_n^{\mathcal{G}_n}}, D_1, \ldots, D_l, \overline{D_{l+1}}, \ldots, \overline{D_m}$$

or

$$[1, 1]\ ID_R\ \texttt{:-}\ A_1^{\mathcal{G}_1}, \ldots, A_k^{\mathcal{G}_k}, \overline{A_{k+1}^{\mathcal{G}_{k+1}}}, \ldots, \overline{A_n^{\mathcal{G}_n}}, D_1, \ldots, D_l, \overline{D_{l+1}}, \ldots, \overline{D_m}$$

where R' (and hence A) is in some $\mathcal{G}_i, 1 \leq i \leq n$.

That is to say, every rule r_0 satisfies at least one of the following conditions:

1. r_0 has atom A or an atom $A^{\mathcal{G}}$ in body; or
2. there is an atom $D^{\mathcal{C}}$ in r_0's body such that A appears in the pp-DAG $\mathcal{C}$; or
3. there is some atom B in r_0's body such that every rule with B in the head has atom A or an atom $A^{\mathcal{G}}$ in the body.

Recall that decyclification eliminates cycles and loop effects, we know that in $acyclic(\boldsymbol{P})$, A does not depend on any rule in $acyclic(\boldsymbol{P}) - acyclic(\boldsymbol{P}'_A)$. So, the proof DAG of $acyclic(\boldsymbol{P})$ for A is the same as the proof DAG of $acyclic(\boldsymbol{P}'_A)$ for A. The theorem is proved. □

A.10 Proof of Theorem 5.4

Proof: First, for every $I \in \mathcal{TV}al(\boldsymbol{P})$, $m_n(I) = \prod_{1 \le k \le n} \widetilde{m}_{Q_k}(I \mid_{\mathcal{C}_k})$, where $Q_k = \boldsymbol{P}_{I|_{\alpha_{k-1}}}(\mathcal{C}_k)$.

Let Q'_k be $acyclic(Q_k)$ and $\boldsymbol{P}'$ be $acyclic(\boldsymbol{P})$, $\forall I \in \mathcal{TV}al(\boldsymbol{P})$,

$$
\begin{aligned}
\widetilde{m}_{\boldsymbol{P}}(I) &= \sum_{I' \in \mathcal{TV}al(\boldsymbol{P}'), I'|_{B_{\boldsymbol{P}}} = I} \hat{m}_{\boldsymbol{P}'}(I') \\
&= \sum_{I'' \in \mathcal{TV}al(B'_{\boldsymbol{P}} - B_{\boldsymbol{P}})} \hat{m}_{\boldsymbol{P}'}(I'' \uplus I) \\
&= \sum_{I'' \in \mathcal{TV}al(B'_{\boldsymbol{P}} - B_{\boldsymbol{P}})} \Big(\prod_{Y \in B'_{\boldsymbol{P}} - B_{\boldsymbol{P}}} s_{\boldsymbol{P}'}(I'' \uplus I, Y) \cdot \prod_{X \in B_{\boldsymbol{P}}} s_{\boldsymbol{P}'}(I, X) \Big) \\
&= \prod_{X \in B_{\boldsymbol{P}}} s_{\boldsymbol{P}'}(I, X) \cdot \sum_{I'' \in \mathcal{TV}al(B'_{\boldsymbol{P}} - B_{\boldsymbol{P}})} \prod_{Y \in B'_{\boldsymbol{P}} - B_{\boldsymbol{P}}} s_{\boldsymbol{P}'}(I'' \uplus I, Y) \\
&= \prod_{X \in B_{\boldsymbol{P}}} s_{\boldsymbol{P}'}(I, X)
\end{aligned}
$$

and

$$
\begin{aligned}
& m_n(I) \\
= & \prod_{1\leq k\leq n} \widetilde{m}_{Q_k}(I\mid_{\mathcal{C}_k}) \\
= & \prod_{1\leq k\leq n} \Big(\sum_{I'_k\in\mathcal{TV}al(Q'_k), I'_k\mid_{\mathcal{C}_k}=I\mid_{\mathcal{C}_k}} \hat{m}_{Q'_k}(I'_k)\Big) \\
= & \prod_{1\leq k\leq n} \Big(\sum_{I''_k\in\mathcal{TV}al(B_{Q'_k}-\mathcal{C}_k)} \hat{m}_{Q'_k}(I''_k \uplus I\mid_{\mathcal{C}_k})\Big) \\
= & \prod_{1\leq k\leq n} \Big(\sum_{I''_k\in\mathcal{TV}al(B_{Q'_k}-\mathcal{C}_k)} \Big(\prod_{Y\in B_{Q'_k}-\mathcal{C}_k} s_{Q'_k}(I''_k\uplus I\mid_{\mathcal{C}_k}, Y)\cdot \prod_{X\in\mathcal{C}_k} s_{Q'_k}(I\mid_{\mathcal{C}_k}, X)\Big)\Big) \\
= & \prod_{1\leq k\leq n} \Big(\prod_{X\in\mathcal{C}_k} s_{Q'_k}(I\mid_{\mathcal{C}_k}, X)\cdot \sum_{I''_k\in\mathcal{TV}al(B_{Q'_k}-\mathcal{C}_k)} \prod_{Y\in B_{Q'_k}-\mathcal{C}_k} s_{Q'_k}(I''_k\uplus I\mid_{\mathcal{C}_k}, Y)\Big) \\
= & \prod_{1\leq k\leq n}\prod_{X\in\mathcal{C}_k} s_{Q'_k}(I\mid_{\mathcal{C}_k}, X)
\end{aligned}
$$

Let $\boldsymbol{P'}(\mathcal{C}_k)$ be $acyclic(\boldsymbol{P}(\mathcal{C}_k))$, it follows from Definition 5.7 and Definition 5.5 that $\forall X\in\mathcal{C}_k$, $s_{Q'_k}(I\mid_{\mathcal{C}_k}, X) = s_{\boldsymbol{P'}(\mathcal{C}_k)}(I, X)$. Also $\boldsymbol{P'} = \bigcup_{1\leq k\leq n}\boldsymbol{P'}(\mathcal{C}_k)$, and hence $\forall X\in\mathcal{C}_k$, $s_{\boldsymbol{P'}(\mathcal{C}_k)}(I, X) = s_{\boldsymbol{P'}}(I, X)$.

Combine the above together, $m_n = \widetilde{m}_{\boldsymbol{P}}$ is proved. □

A.11 Proof of Theorem 6.1

Proof: We need to prove that, for any blp-cf $\langle \boldsymbol{P}, \mathbf{CF}\rangle$,

$$\sum_{I\in\mathcal{TV}al(\langle \boldsymbol{P},\mathbf{CF}\rangle)} \hat{m}_{\langle \boldsymbol{P},\mathbf{CF}\rangle}(I) = 1$$

Let us split $B_{\langle \boldsymbol{P},\mathbf{CF}\rangle}$ into three subsets of atoms: $\mathbb{X}$ — the set of atoms appearing in the head of some rule, $\mathbb{Z}$ — the set of atoms appearing in **CF**, and $\mathbb{Y} = B_{\langle \boldsymbol{P},\mathbf{CF}\rangle} - \mathbb{X} - \mathbb{Z}$.

Since there is no circular dependency among the atoms in blp-cf, we can order all atoms in $\mathbb{X}$ as $a_1, ..., a_n$, so that a_k depends only on atoms in

$\{a_{k+1}, ..., a_n\} \cup \mathbb{Y} \cup \mathbb{Z}$, $1 \leq k \leq n$. Let $\mathbb{X}_k$ denote $\{a_{k+1}, ..., a_n\}$, $0 \leq k < n$. Note that $\mathbb{X}_0 = \mathbb{X}$.

For any $F \in \mathbf{CF}$, let $atm(F)$ be the set of atoms appearing in F. From Definition 6.3, we have

$$\begin{aligned}
&\sum_{I \in \mathcal{TV}al(\langle \boldsymbol{P}, \mathbf{CF} \rangle)} \hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}(I) \\
= &\sum_{I \in \mathcal{TV}al(\mathbb{Z} \cup \mathbb{Y} \cup \mathbb{X}_0)} \Big(\prod_{F \in \mathbf{CF}} corr(F)(I \mid_{atm(F)}) \cdot \prod_{A \in \mathbb{Y}} s_{\boldsymbol{P}}(I, A) \cdot \prod_{A \in \mathbb{X}_0} s_{\boldsymbol{P}}(I, A) \Big) \\
= &\sum_{I' \in \mathcal{TV}al(\mathbb{Z})} \sum_{I'' \in \mathcal{TV}al(\mathbb{Y} \cup \mathbb{X}_0)} \Big(\prod_{F \in \mathbf{CF}} corr(F)\big((I' \uplus I'') \mid_{atm(F)}\big) \cdot \\
&\qquad \prod_{A \in \mathbb{Y}} s_{\boldsymbol{P}}(I' \uplus I'', A) \cdot \prod_{A \in \mathbb{X}_0} s_{\boldsymbol{P}}(I' \uplus I'', A) \Big)
\end{aligned} \tag{A.17}$$

Since $corr(F)(I' \uplus I'' \mid_{atm(F)}) = corr(F)(I' \mid_{atm(F)})$, we get the following from (A.17):

$$\begin{aligned}
&\sum_{I \in \mathcal{TV}al(\langle \boldsymbol{P}, \mathbf{CF} \rangle)} \hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}(I) \\
= &\sum_{I' \in \mathcal{TV}al(\mathbb{Z})} \Bigg(\prod_{F \in \mathbf{CF}} corr(F)(I' \mid_{atm(F)}) \cdot \\
&\qquad \sum_{I'' \in \mathcal{TV}al(\mathbb{Y} \cup \mathbb{X}_0)} \Big(\prod_{A \in \mathbb{Y}} s_{\boldsymbol{P}}(I' \uplus I'', A) \cdot \prod_{A \in \mathbb{X}_0} s_{\boldsymbol{P}}(I' \uplus I'', A) \Big) \Bigg)
\end{aligned} \tag{A.18}$$

Next, we can prove that for any $I' \in \mathcal{TV}al(\mathbb{Z})$

$$\sum_{I'' \in \mathcal{TV}al(\mathbb{Y} \cup \mathbb{X}_0)} \Big(\prod_{A \in \mathbb{Y}} s_{\boldsymbol{P}}(I' \uplus I'', A) \cdot \prod_{A \in \mathbb{X}_0} s_{\boldsymbol{P}}(I' \uplus I'', A) \Big) \; = \; 1$$

by following the same sequence of steps as the one used to prove Theorem 3.1 from (A.1).

Let $\mathbf{CF} = \{F_1, ..., F_m\}$ and $\bigcup_{1 \leq k \leq m} atm(F_k) = \mathbb{Z}$. Since in a blp-cf no atom appears in two different correlation formulas at once, $atm(F_i) \cap atm(F_j) = \varnothing$, if $i \neq j$. Continuing with (A.18), we obtain

$$
\begin{aligned}
&\sum_{I \in \mathcal{TV}al(\langle \boldsymbol{P}, \mathbf{CF} \rangle)} \hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}(I) \\
&= \sum_{I \in \mathcal{TV}al(\mathbb{Z})} \prod_{k=1}^{m} corr(F_k)(I \mid_{atm(F_k)}) \\
&= \sum_{I_1 \in \mathcal{TV}al(atm(F_1))} \cdots \sum_{I_m \in \mathcal{TV}al(atm(F_m))} \prod_{k=1}^{m} corr(F_k)\Big((\biguplus_{j=1}^{m} I_j) \mid_{atm(F_k)} \Big) \\
&= \sum_{I_1 \in \mathcal{TV}al(atm(F_1))} \cdots \sum_{I_m \in \mathcal{TV}al(atm(F_m))} \prod_{k=1}^{m} corr(F_k)(I_k) \\
&= \prod_{k=1}^{m} \sum_{I_k \in \mathcal{TV}al(atm(F_k))} corr(F_k)(I_k) \;=\; \prod_{k=1}^{m} 1 \\
&= 1
\end{aligned}
$$

□

A.12 Proof of Theorem 6.2

Proof: Suppose $a_1, ..., a_n$ are the labels for the nodes in the dependency DAG of $\langle \boldsymbol{P}, \mathbf{CF} \rangle$. Note that a_i can be a label for an a-node or an r-node. Without loss of generality, suppose the traversal order in the modified bottom-up algorithm is $\langle a_1, ..., a_n \rangle$.

Let $m_0 = m_\varnothing$, m_k, $1 \leq k \leq n$, be the support functions constructed by the modified bottom-up algorithm after visiting a_k, and $\mathcal{S}_k$ be the base of m_k. Also, let R be the set of propositions that label the rules in $\boldsymbol{P}$. We can see that $\mathcal{S}_n = \{a_1, ..., a_n\} = B_{\langle \boldsymbol{P}, \mathbf{CF} \rangle} \cup R$. As before, we use $atm(\mathbf{CF})$ to denote the set of atoms that appear in $\mathbf{CF}$ and $atm(F)$ will denote the set of atoms that appear in $F \in \mathbf{CF}$.

We need to prove that m_n is an expansion of $\hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}$ (see Definition 6.3). The result then follows from Lemma 4.2.

Let $I \in \mathcal{TV}al(\mathcal{S}_n)$ be a truth valuation. During the process of expansion

of the support function from m_0 to m_1 to ... to m_n in the algorithm, we see that, for any k ($1 \leq k \leq n$):

- If $a_k \in B_{\boldsymbol{P}} - atm(\mathbf{CF})$, then $\mathcal{S}_k = \mathcal{S}_{k-1} \cup \{a_k\}$, and

$$m_k(I \mid_{\mathcal{S}_k}) \;=\; m_{k-1}(I \mid_{\mathcal{S}_{k-1}}) \;\cdot\; s_{\boldsymbol{P}}(I, a_k)$$

- If $a_k \in R$, then $\mathcal{S}_k = \mathcal{S}_{k-1} \cup \{a_k\}$, and

$$m_k(I \mid_{\mathcal{S}_k}) \;=\; m_{k-1}(I \mid_{\mathcal{S}_{k-1}}) \;\cdot\; 1, \text{ if } I(a_k) = \mathbf{t}$$

$$m_k(I \mid_{\mathcal{S}_k}) \;=\; m_{k-1}(I \mid_{\mathcal{S}_{k-1}}) \;\cdot\; 0, \text{ if } I(a_k) \neq \mathbf{t}$$

- If $a_k \in atm(\mathbf{CF})$, let $F \in \mathbf{CF}$ be the correlation formula such that $a_k \in atm(F)$.
  - If $a_k \notin \mathcal{S}_{k-1}$, then $\mathcal{S}_k = \mathcal{S}_{k-1} \cup atm(F)$, and

$$m_k(I \mid_{\mathcal{S}_k}) \;=\; m_{k-1}(I \mid_{\mathcal{S}_{k-1}}) \;\cdot\; corr(F)(I \mid_{atm(F)})$$

  - If $a_k \in \mathcal{S}_{k-1}$, then $\mathcal{S}_k = \mathcal{S}_{k-1}$, and

$$m_k(I \mid_{\mathcal{S}_k}) \;=\; m_{k-1}(I \mid_{\mathcal{S}_{k-1}})$$

From the above, we can conclude the following for any $I' \in \mathcal{TV}al(\langle \boldsymbol{P}, \mathbf{CF} \rangle)$ and $I \in \mathcal{TV}al(\mathcal{S}_n)$ such that $I \mid_{B_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}} = I'$:

1. If $\exists c \in R,\ I(c) \neq \mathbf{t}$, then
$$m_n(I) = 0$$
2. If $I = I' \uplus \{R \rightarrow \mathbf{t}\}$, then
$$m_n(I) = \prod_{1 \leq k \leq n} \delta_k$$
where

- $\delta_k = s_{\boldsymbol{P}}(I, a_k)$ if $a_k \in B_{\boldsymbol{P}} - atm(\mathbf{CF})$;
- $\delta_k = 1$ if $a_k \in R$;
- $\delta_k = corr(F)(I\ |_{atm(F)})$ if $a_k \in atm(F)$, where $F \in \mathbf{CF}$, and $a_k \notin \mathcal{S}_{k-1}$;
- $\delta_k = 1$ if $a_k \in atm(\mathbf{CF})$ and $a_k \in \mathcal{S}_{k-1}$.

Note that for any given $F \in \mathbf{CF}$, there is exactly one atom $a_i \in atm(\mathbf{CF})$, $1 \leq i \leq n$, such that $\delta_i = corr(F)(I\ |_{atm(F)})$. Therefore,

$$\begin{aligned} m_n(I) &= \prod_{F \in \mathbf{CF}} corr(F)(I\ |_{atm(F)}) \cdot \prod_{a \in B_{\boldsymbol{P}} - atm(\mathbf{CF})} s_{\boldsymbol{P}}(I', a) \\ &= \hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}(I') \end{aligned}$$

Thus, given $I' \in \mathcal{TV}al(\langle \boldsymbol{P}, \mathbf{CF} \rangle)$, $m_n(I)$ is non-zero for only one $I \in \mathcal{TV}al(\mathcal{S}_n)$ such that $I\ |_{B_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}} = I'$. Summing up $m_n(I)$ for all such I's, we obtain

$$\sum_{I \in \mathcal{TV}al(\mathcal{S}_n) \text{ where } I|_{B_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}} = I'} m_n(I) \;=\; \hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}(I')$$

By Definition 4.3, m_n is an expansion of $\hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}$, so the result follows from Lemma 4.2. □

A.13 Proof of Lemma 7.1

Proof: For any $I \in \mathcal{TV}al(\mathcal{S})$ such that $I(\mathcal{C}) \subseteq \{\mathbf{t}, \mathbf{f}\}$, let $\mathcal{C}_I^{\mathbf{f}}$ be the subset of $\mathcal{C}$ such that $I\ |_{\mathcal{C}_I^{\mathbf{f}}} = \mathbf{f}$ and $I\ |_{\mathcal{C} - \mathcal{C}_I^{\mathbf{f}}} = \mathbf{t}$. Let

$$\Omega(I) \;=\; \Big\{ I' \mid I' \in \mathcal{TV}al\big(\mathcal{S}\big),\ I'\ |_{\mathcal{S} - \mathcal{C}_I^{\mathbf{f}}} = I\ |_{\mathcal{S} - \mathcal{C}_I^{\mathbf{f}}},\ \text{and } I'(\mathcal{C}_I^{\mathbf{f}}) \subseteq \{\mathbf{u}, \mathbf{f}\} \Big\}$$

Since m' is the u-to-f retraction of m with respect to $\mathcal{C}$, by Definition 7.2, we have

$$\begin{aligned}
&\sum_{I\in\mathcal{TV}al(\mathcal{S}) \text{ such that } I(a)=\mathbf{t}} m'(I) \\
&= \sum_{\substack{I\in\mathcal{TV}al(\mathcal{S}) \text{ such that } I(a)=\mathbf{t} \\ \text{and } I(\mathcal{C})\subseteq\{\mathbf{t},\mathbf{f}\}}} m'(I) \\
&= \sum_{\substack{I\in\mathcal{TV}al(\mathcal{S}) \text{ such that } I(a)=\mathbf{t} \\ \text{and } I(\mathcal{C})\subseteq\{\mathbf{t},\mathbf{f}\}}} \sum_{I'\in\Omega(I)} m(I') \qquad \text{(A.19)}
\end{aligned}$$

For any pair of valuations $I_1, I_2 \in \mathcal{TV}al(\mathcal{S})$ such that $I_1(\mathcal{C}) \subseteq \{\mathbf{t},\mathbf{f}\}$, $I_2(\mathcal{C}) \subseteq \{\mathbf{t},\mathbf{f}\}$, it is easy to see that $\Omega(I_1) \cap \Omega(I_2) = \varnothing$ if $I_1 \neq I_2$. So, continuing with (A.19), we have

$$\sum_{I\in\mathcal{TV}al(\mathcal{S}) \text{ such that } I(a)=\mathbf{t}} m'(I) \;=\; \sum_{\substack{I'\in\mathcal{TV}al(\mathcal{S}) \text{ such that } \exists I\in\mathcal{TV}al(\mathcal{S}), \\ I(\mathcal{C})\subseteq\{\mathbf{t},\mathbf{f}\},\ I'\in\Omega(I),\ I(a)=\mathbf{t}}} m(I')$$

Now we only need to prove that the set

$$\mathcal{T}_1 \;=\; \big\{I' \in \mathcal{TV}al(\mathcal{S}) \mid\ I'(a) = \mathbf{t}\big\}$$

equals the set

$$\mathcal{T}_2 \;=\; \Big\{I' \in \mathcal{TV}al(\mathcal{S}) \mid\ \exists I \in \mathcal{TV}al(\mathcal{S}),\ I(a) = \mathbf{t},\ I(\mathcal{C}) \subseteq \{\mathbf{t},\mathbf{f}\},\ I' \in \Omega(I)\Big\}$$

Indeed, consider some $I' \in \mathcal{TV}al(\mathcal{S})$.

- There is a truth valuation $I \in \mathcal{TV}al(\mathcal{S})$, such that for any $X \in \mathcal{S}$
  - $I(X) = I'(X)$ if $X \notin \mathcal{C}$;
  - $I(X) = \mathbf{t}$ if $X \in \mathcal{C}$ and $I'(X) = \mathbf{t}$;
  - $I(X) = \mathbf{f}$ if $X \in \mathcal{C}$ and $I'(X) \in \{\mathbf{f},\mathbf{u}\}$;

  Clearly, $I' \in \Omega(I)$. If $I' \in \mathcal{T}_1$, then $I'(a) = \mathbf{t}$ and so also $I(a) = \mathbf{t}$. Therefore, $I' \in \mathcal{T}_2$. This proves $\mathcal{T}_1 \subseteq \mathcal{T}_2$.

- If $I' \in \mathcal{T}_2$, then $I' \in \Omega(I)$ and there is a truth valuation $I \in \mathcal{TV}al(\mathcal{S})$, such that $I(\mathcal{C}) \subseteq \{\mathbf{t}, \mathbf{f}\}$ and $I(a) = \mathbf{t}$.

  - If $a \notin \mathcal{C}$, it follows $a \in \mathcal{S} - \mathcal{C}_I^{\mathbf{f}}$.
  - If $a \in \mathcal{C}$, since $I(a) = \mathbf{t}$, we also conclude that $a \in \mathcal{S} - \mathcal{C}_I^{\mathbf{f}}$.

  Since $I' \in \Omega(I)$, we have $I' \mid_{\mathcal{S}-\mathcal{C}_I^{\mathbf{f}}} = I \mid_{\mathcal{S}-\mathcal{C}_I^{\mathbf{f}}}$ and, therefore, $I'(a) = I(a) = \mathbf{t}$. This means that $I' \in \mathcal{T}_1$, which proves $\mathcal{T}_2 \subseteq \mathcal{T}_1$.

□

A.14 Proof of Theorem 7.2

Proof: Note that $atm(F) = atm(F')$, so

$$\mathcal{TV}al(\langle \boldsymbol{P}, \mathbf{CF} \rangle) = \mathcal{TV}al(\langle \boldsymbol{P}, \mathbf{CF} \cup \{F'\} - \{F\} \rangle)$$

By Definition 6.4,

$$\begin{aligned} \mathtt{bel}_1(g) &= \sum_{\substack{I \in \mathcal{TV}al\left(\langle \boldsymbol{P}, \mathbf{CF} \rangle\right) \\ \text{such that } I(g)=\mathbf{t}}} \hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}(I) \\ \mathtt{bel}_2(g) &= \sum_{\substack{I \in \mathcal{TV}al\left(\langle \boldsymbol{P}, \mathbf{CF} \rangle\right) \\ \text{such that } I(g)=\mathbf{t}}} \hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \cup \{F'\} - \{F\} \rangle}(I) \end{aligned}$$

According to Lemma 7.1, we only need to prove that $\hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \cup \{F'\} - \{F\} \rangle}$ is a u-to-f retraction of $\hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}$ with respect to $atm(F)$. Indeed, by Definition 6.3, for any $I \in \mathcal{TV}al\left(\langle \boldsymbol{P}, \mathbf{CF} \rangle\right)$ we have:

$$\begin{aligned} \hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \cup \{F'\} - \{F\} \rangle}(I) &= corr(F')(I \mid_{atm(F)}) \cdot \\ & \prod_{E \in \mathbf{CF} - \{F\}} corr(E)(I \mid_{atm(E)}) \cdot \prod_{X \in B_{\boldsymbol{P}} - atm(\mathbf{CF})} s_{\boldsymbol{P}}(I, X) \end{aligned}$$

- If there is some X in $atm(F)$ such that $I(X) = \mathbf{u}$, since $corr(F')$ is the u-to-f retraction of $corr(F)$ with respect to $atm(F)$, we conclude that $corr(F')(I\mid_{atm(F)}) = 0$. Therefore,

$$\hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \cup \{F'\} - \{F\}\rangle}(I) = 0$$

- If there is no X in $atm(F)$ such that $I(X) = \mathbf{u}$, let $\mathcal{C}_I^{\mathbf{f}}$ be the subset of $atm(F)$ such that $I\mid_{\mathcal{C}_I^{\mathbf{f}}} = \mathbf{f}$ and $I\mid_{atm(F)-\mathcal{C}_I^{\mathbf{f}}} = \mathbf{t}$. Also, let

$$\Omega(I) \;=\; \Big\{ I' \mid I' \in \mathcal{TV}al\big(\langle \boldsymbol{P}, \mathbf{CF}\rangle\big),\; I'\mid_{B_{\langle \boldsymbol{P},\mathbf{CF}\rangle} - \mathcal{C}_I^{\mathbf{f}}} = I\mid_{B_{\langle \boldsymbol{P},\mathbf{CF}\rangle} - \mathcal{C}_I^{\mathbf{f}}}, \text{ and } I'(\mathcal{C}_I^{\mathbf{f}}) \subseteq \{\mathbf{u}, \mathbf{f}\} \Big\}$$

Note that if $E \in \mathbf{CF} - \{F\}$ is a correlation formula other than F then $atm(E) \cap atm(F) = \varnothing$, by the assumption in the definition of blp-cfs (below Example 6.2). Therefore, $B_{\langle \boldsymbol{P},\mathbf{CF}\rangle} - atm(E) \subseteq B_{\langle \boldsymbol{P},\mathbf{CF}\rangle} - atm(F) \subseteq B_{\langle \boldsymbol{P},\mathbf{CF}\rangle} - \mathcal{C}_I^{\mathbf{f}}$. So, for any $I' \in \Omega(I)$, we must have $I'\mid_{atm(E)} \;=\; I\mid_{atm(E)}$ and thus

$$\prod_{E \in \mathbf{CF} - \{F\}} corr(E)(I'\mid_{atm(E)}) \;=\; \prod_{E \in \mathbf{CF} - \{F\}} corr(E)(I\mid_{atm(E)}) \quad \text{(A.20)}$$

Recall that, by assumption, $\overline{X}$ does not appear in $\boldsymbol{P}$ for all $X \in atm(F)$. Therefore, by construction of $C_I^{\mathbf{f}}$ and $\Omega(I)$, it follows that $\boldsymbol{P}_{I'}(X) = \boldsymbol{P}_I(X)$ and

$$s_{\boldsymbol{P}}(I', X) = s_{\boldsymbol{P}}(I, X) \quad \text{(A.21)}$$

for any $I' \in \Omega(I)$ and $X \in B_{\boldsymbol{P}}$.

By (A.20) and (A.21), we obtain

$$
\begin{aligned}
&\sum_{I' \in \Omega(I)} \hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}(I') \\
= &\sum_{I' \in \Omega(I)} \Big(corr(F)(I' \mid_{atm(F)}) \cdot \prod_{E \in \mathbf{CF} - \{F\}} corr(E)(I' \mid_{atm(E)}) \\
&\qquad \cdot \prod_{X \in B_{\boldsymbol{P}} - atm(\mathbf{CF})} s_{\boldsymbol{P}}(I', X) \Big) \\
= &\sum_{I' \in \Omega(I)} corr(F)(I' \mid_{atm(F)}) \cdot \prod_{E \in \mathbf{CF} - \{F\}} corr(E)(I \mid_{atm(E)}) \qquad \text{(A.22)} \\
&\cdot \prod_{X \in B_{\boldsymbol{P}} - atm(\mathbf{CF})} s_{\boldsymbol{P}}(I, X)
\end{aligned}
$$

Since $corr(F')$ is a u-to-f retraction of $corr(F)$ with respect to $atm(F)$, we have

$$
\begin{aligned}
&corr(F')(I \mid_{atm(F)}) \\
&\quad = \sum_{\substack{I' \in \mathcal{TV}al(atm(F)) \text{ such that} \\ I'|_{atm(F) - \mathcal{C}_I^{\mathbf{f}}} = I|_{atm(F) - \mathcal{C}_I^{\mathbf{f}}} \\ \text{and } I'(\mathcal{C}_I^{\mathbf{f}}) \subseteq \{\mathbf{u}, \mathbf{f}\}}} corr(F)(I') \\
&\quad = \sum_{\substack{I' \in \mathcal{TV}al(\langle \boldsymbol{P}, \mathbf{CF} \rangle) \text{ such that} \\ I'|_{B_{\langle \boldsymbol{P}, \mathbf{CF} \rangle} - \mathcal{C}_I^{\mathbf{f}}} = I|_{B_{\langle \boldsymbol{P}, \mathbf{CF} \rangle} - \mathcal{C}_I^{\mathbf{f}}} \\ \text{and } I'(\mathcal{C}_I^{\mathbf{f}}) \subseteq \{\mathbf{u}, \mathbf{f}\}}} corr(F)(I' \mid_{atm(F)}) \\
&\quad = \sum_{I' \in \Omega(I)} corr(F)(I' \mid_{atm(F)}) \qquad \text{(A.23)}
\end{aligned}
$$

Combining (A.22) and (A.23) we get

$$
\begin{aligned}
&\sum_{I' \in \Omega(I)} \hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}(I') \\
&\quad = corr(F')(I \mid_{atm(F)}) \cdot \prod_{E \in \mathbf{CF} - \{F\}} corr(E)(I \mid_{atm(E)}) \\
&\qquad \cdot \prod_{X \in B_{\boldsymbol{P}} - atm(\mathbf{CF})} s_{\boldsymbol{P}}(I, X) \\
&\quad = \hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \cup \{F'\} - \{F\} \rangle}(I) \qquad \text{(A.24)}
\end{aligned}
$$

By Definition 7.2, $\hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \cup \{F'\} - \{F\} \rangle}$ is a u-to-f retraction of $\hat{m}_{\langle \boldsymbol{P}, \mathbf{CF} \rangle}$ with respect to $atm(F)$. Now, by Lemma 7.1, we get $\mathtt{bel}_1(g) = \mathtt{bel}_2(g)$, which concludes the proof. □