

Belief Logic Programming with Cyclic Dependencies

Hui Wan

State University of New York at Stony Brook
Stony Brook, NY 11794, USA

Abstract. Our previous work [26] introduced Belief Logic Programming (BLP), a novel form of quantitative logic programming with correlation of evidence. Unlike other quantitative approaches to logic programming, this new theory is able to provide accurate conclusions in the presence of uncertainty when the sources of information are not independent. However, the semantics defined in [26] is not sufficiently general—it does not allow cyclic dependencies among beliefs, which is a serious limitation of expressive power. This paper extends the semantics of BLP to allow cyclic dependencies. We show that the new semantics is backward compatible with the semantics for acyclic BLP and has the expected properties. The results are illustrated with examples of inference in a simple diagnostic expert system.

1 Introduction

Quantitative reasoning has been widely used for dealing with uncertainty and inconsistency in knowledge representation and management, and, more recently, on the Semantic Web [14]. Among the various forms of quantitative reasoning, quantitative logic programming is a very important one.

Based on the approaches of uncertainty deduction, Lakshmanan and Shiri [11] classified the proposed quantitative logic programming frameworks into *annotation-based* and *implication-based* as follows:

- In the annotation-based frameworks such as [2, 9, 10, 16, 15, 17, 24], a rule is of the form $A : f(\beta_1, \dots, \beta_n) :- B_1 : \beta_1 \wedge \dots \wedge B_n : \beta_n$ which asserts “the certainty of A is at least (or is in) $f(\beta_1, \dots, \beta_n)$, 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 [1, 11, 8, 12, 13, 20, 22, 5], a rule is of the form $\beta : A :- B_1 \wedge \dots \wedge B_n$ which asserts “the certainty that $B_1 \wedge \dots \wedge B_n$ 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 [15], 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 cases, such as [13, 11], 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 [8, 19, 22], 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 [8, 19] simply do not allow programs with recursive loops. Some others, such as [6, 22], eliminate recursive loops by introducing time parameters into atoms involved in loops, either explicitly or implicitly.

In a vast open world like the Semantic Web, it is often impossible for an application to acquire complete information from one information source, thus combination and correlation of evidence from multiple sources are necessary. In our previous work [26], we introduced a novel form of implication-based quantitative reasoning, called *Belief Logic Programming* (BLP) [26]. BLP can take into account correlation of evidence obtained from different, but overlapping and, possibly, contradicting information sources. This makes BLP very suitable for Web reasoning. BLP's semantics is based on belief combination functions and is inspired by Dempster-Shafer theory of evidence [3, 21]. In [26], we related BLP semantics to Dempster-Shafer theory and also showed the connection with certain forms of defeasible reasoning, such as Courteous Logic Programming [7] and, more generally, Logic Programming with Courteous Argumentation Theories (LPDA) [25]. [26] also provides a detailed motivation of BLP and the arguments showing limitations of the earlier logic programming approaches—those based on probability theory, Fuzzy Logic, Dempster-Shafer theory, and other approaches [1, 2, 9–11, 13, 15, 17–19, 24]. The same limitations apply to other frameworks of uncertainty reasoning in the Semantic Web, e.g., [4, 23]. In this paper we will not rehash the detailed motivation of BLP but focus on how to deal with recursive loops in BLP.

Like other quantitative logic programming frameworks, BLP faces the same challenge from recursive loops, and the semantics defined in [26] was restricted to belief logic programs without cyclic dependency among atoms. In the present paper we extend the previous work to belief logic programs with cycles by adapting 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 [26]. The query answering algorithms proposed in [27] can also be adapted to belief logic programs with cycles, but for reason of focus and space we will not address query answering in this paper.

The paper is organized as follows. Section 2 explains the problem in detail by a motivating example. Section 3 is an overview of the syntax and the semantics for *acyclic* BLP programs. In Section 4, we introduce a transformational se-

mantics and a fixpoint semantics for general BLP programs, which may include cycles. Section 5 concludes the paper. Proofs can be found in [28].

2 Motivating Example

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 1. We can see that there is a loop (a cyclic dependency) between “ p_3 has disease” and “ p_4 has disease”.

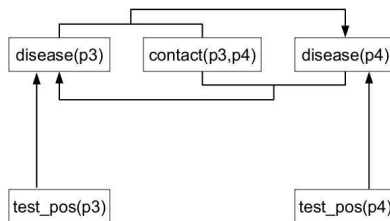


Fig. 1. Support Relation w.r.t. p_3 and p_4 in disease example

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

3 Preliminaries

3.1 Syntax of BLP

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

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

where X is a positive atom and $Body$ is a conjunction of literals, i.e., a conjunction of atoms and negation of atoms.¹ An atom in BLP has the form $p(t_1, \dots, t_n)$, where p is a predicate and t_i is a constant or a variable, $1 \leq i \leq n$. We will use capital letters to denote positive atoms, e.g., A , and a bar over such a letter will denote negation, e.g., \bar{A} . 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 \bar{X} to the degree $1 - w$. The difference, $w - v$, is the *information gap* (or the degree of ignorance) with regard to X .

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 paper we will deal only with annotated rules and facts and refer to them simply as rules and facts.

Definition 1. *Given a blp P , an atom X is said to **depend on** an atom Y*

- *directly, if X is the head of a rule R and Y occurs in the body of R ;*
- *indirectly, if X is dependent on Z , and Z depends on Y .* □

A blp P is said to be **cyclic** if there is an atom that depends on itself. Otherwise P is said to be **acyclic**. In [26], we required that in a blp there can be no circular dependency among atoms, i.e., [26] only considered acyclic blps. This is a serious limitation of express power. In this paper we will remove this restriction and allow circular dependency in Section 4.

3.2 Combination Functions

Definition 2. *Let D be the set of all belief factors, $\Phi : D \times D \rightarrow D$ is said to be a **belief combination function** if Φ is associative and commutative.* □

Due to the associativity of Φ , we can extend it from two arguments to nullary case, single argument, and three and more arguments: $\Phi() = [0, 1]$, $\Phi([v, w]) = [v, w]$, $\Phi([v_1, w_1], \dots, [v_k, w_k]) = \Phi(\Phi([v_1, w_1], \dots, [v_{k-1}, w_{k-1}]), [v_k, w_k])$. 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 belief factors, e.g., $\Phi(\{[v_1, w_1], \dots, [v_k, w_k]\})$.

Different types of beliefs might require different ways to combine them, so predicates in the same blp might be using different combination functions. Here are some popular combination functions:

- *Dempster's combination rule:*
 - $\Phi^{DS}([0, 0], [1, 1]) = [0, 1]$.
 - $\Phi^{DS}([v_1, w_1], [v_2, w_2]) = [v, w]$ if $\{[v_1, w_1], [v_2, w_2]\} \neq \{[0, 0], [1, 1]\}$, where $v = \frac{v_1 \cdot w_2 + v_2 \cdot w_1 - v_1 \cdot v_2}{K}$, $w = \frac{w_1 \cdot w_2}{K}$, and $K = 1 + v_1 \cdot w_2 + v_2 \cdot w_1 - v_1 - v_2$.
- *Maximum:* $\Phi^{max}([v_1, w_1], [v_2, w_2]) = [max(v_1, v_2), max(w_1, w_2)]$.
- *Minimum:* $\Phi^{min}([v_1, w_1], [v_2, w_2]) = [min(v_1, v_2), min(w_1, w_2)]$.

¹ In the BLP syntax and semantics in [26], rule bodies are Boolean combinations of literals. Since every blp can be transformed into an equivalent blp without body-disjunctions, as shown in [27], here we assume there is no disjunction in rule bodies.

3.3 Semantics of Acyclic BLP

Given a blp \mathbf{P} , the definitions of *Herbrand Universe* $U_{\mathbf{P}}$ and *Herbrand Base* $B_{\mathbf{P}}$ of \mathbf{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 $X \in B_{\mathbf{P}}$ has an associated belief combination function, denoted Φ_X .² Intuitively, Φ_X is used to help determine the *combined* belief in X accorded by the rules in \mathbf{P} that support X .

Definition 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 \mathbf{P} is a truth valuation over $B_{\mathbf{P}}$. Let $\mathcal{TV}al(\mathbf{P})$ denote the set of all the truth valuations for \mathbf{P} , so $\mathcal{TV}al(\mathbf{P}) = \mathcal{TV}al(B_{\mathbf{P}})$. \square

Definition 4. 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 \mathbf{P} is a mapping m from $\mathcal{TV}al(\mathbf{P})$ to $[0, 1]$ such that $\sum_{I \in \mathcal{TV}al(\mathbf{P})} m(I) = 1$. \square

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

Definition 5. Given a truth valuation I over a set of atoms α and a formula $F \in \mathcal{Bool}(\alpha)$, $I(F)$ is defined as in Lukasiewicz's three-valued logic: $I(A \vee B) = \max(I(A), I(B))$, $I(A \wedge B) = \min(I(A), I(B))$, and $I(\bar{A}) = \neg I(A)$, 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 $I(F) = \mathbf{t}$. \square

Definition 6. A mapping $\mathbf{bel} : \mathcal{Bool}(B_{\mathbf{P}}) \rightarrow [0, 1]$ is said to be a *belief function* for \mathbf{P} if there exists a support function m for \mathbf{P} , so that for all $F \in \mathcal{Bool}(B_{\mathbf{P}})$, $\mathbf{bel}(F) = \sum_{I \in \mathcal{TV}al(\mathbf{P})} \text{such that } I \models F m(I)$. \square

Belief functions can be thought of as *interpretations* of belief logic programs. However, as usual in logic programming, we are interested not just in interpretations, but in models. We define the model of an acyclic blp next.

Definition 7. Given an acyclic blp \mathbf{P} and a truth valuation I , we define \mathbf{P} 's *reduct under* I to be $\mathbf{P}_I = \{R \mid R \in \mathbf{P}, I \models \text{Body}(R)\}$, where $\text{Body}(R)$ denotes the body of the rule R .

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

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

Definition 8. Given an acyclic blp \mathbf{P} and a truth valuation I for \mathbf{P} , for any $X \in B_{\mathbf{P}}$, we define $s_{\mathbf{P}}(I, X)$, called the *\mathbf{P} -support for X in I* , as follows.

² Separate belief combination functions can be associated to different predicates or even ground atoms.

1. If $\mathbf{P}_I(X) = \phi$, then
 - If $I(X) = \mathbf{t}$ or $I(X) = \mathbf{f}$, then $s_{\mathbf{P}}(I, X) = 0$;
 - If $I(X) = \mathbf{u}$, then $s_{\mathbf{P}}(I, X) = 1$.
2. If $\mathbf{P}_I(X) = \{R_1, \dots, R_n\}$, $n > 0$, let $[v, w]$ be the result of applying Φ_X to the belief factors of the rules R_1, \dots, R_n . Then
 - If $I(X) = \mathbf{t}$, then $s_{\mathbf{P}}(I, X) = v$;
 - If $I(X) = \mathbf{f}$, then $s_{\mathbf{P}}(I, X) = 1 - w$;
 - If $I(X) = \mathbf{u}$, then $s_{\mathbf{P}}(I, X) = w - v$. □

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

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

Definition 9. If I is a truth valuation for an acyclic blp \mathbf{P} , then

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

Theorem 1. For any acyclic blp \mathbf{P} , $\sum_{I \in T\text{Val}(\mathbf{P})} \hat{m}_{\mathbf{P}}(I) = 1$. □

This theorem is crucial, as it makes the following definition well-founded.

Definition 10. The **model** of an acyclic blp \mathbf{P} is the following belief function:

$$\text{model}(F) = \sum_{I \in T\text{Val}(\mathbf{P}) \text{ such that } I \models F} \hat{m}_{\mathbf{P}}(I), \quad \text{where } F \in \text{Bool}(B_{\mathbf{P}}). \quad \square$$

In [26] we showed that **model** is a “correct” (and unique) belief function that one should expect: it provides each atom in the Herbrand base with precisely the right amount of support from all the applicable rules. Namely, if S is a suitable set of the rules that support A (see [26] for a precise formulation) then

$$\frac{\text{model}(A \wedge \bigwedge_{R \in S} \text{Body}(R))}{\text{model}(\bigwedge_{R \in S} \text{Body}(R))} = v \quad \frac{\text{model}(\bar{A} \wedge \bigwedge_{R \in S} \text{Body}(R))}{\text{model}(\bigwedge_{R \in S} \text{Body}(R))} = 1 - w$$

where $[v, w] = \Phi_X(BF_S)$ and BF_S is the multiset of belief factors of rules in S .

4 Semantics for General BLP

We introduce some necessary notions first.

Definition 11. Let \mathbf{P} be a blp. $B_{\mathbf{P}}$ can always be partitioned into disjoint atom sets $\mathcal{C}_1, \dots, \mathcal{C}_k$, such that two atoms are in the same set if and only if they depend on each other. We call $\mathcal{C}_1, \dots, \mathcal{C}_k$ the **atom cliques** of \mathbf{P} and use $\text{clique}(A)$ to denote the atom clique that contains atom A .

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

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 $[v, w] A :- A$.

4.1 Transformational Semantics for General 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 \mathbf{P} into an acyclic blp \mathbf{P}' , which captures all the non-trivial and non-redundant belief derivations. Then the minimal model of \mathbf{P} is defined to be the minimal model of the acyclic blp \mathbf{P}' .

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

Definition 12. The **dependency graph**, \mathcal{H} , of a ground blp \mathbf{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 \mathbf{P} , \mathcal{H} has an a-node labeled A .
- For each rule R in \mathbf{P} , \mathcal{H} has an r-node labeled with proposition ID_R .
- For each rule R in \mathbf{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 \mathbf{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 . \square

The dependency graph \mathcal{H} describes the dependency relation over $B_{\mathbf{P}}$. 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 13. Let \mathbf{P} be a blp and \mathcal{H} be \mathbf{P} 's dependency graph. A directed graph \mathcal{G} is called a **partial-proof DAG of \mathbf{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 $\text{clique}(A)$ and has exactly one child.
4. If an a-node D belongs to $\text{clique}(A)$, and D 's parent is in \mathcal{G} , then D itself is also in \mathcal{G} . \square

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 1. Returning to the example in Section 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.

$$\begin{aligned} [0.8, 1] \text{disease}(?X) & :- \text{test_pos}(?X). \\ [0.6, 1] \text{disease}(?X) & :- \text{contact}(?X, ?Y) \wedge \text{disease}(?Y). \end{aligned}$$

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 \mathbf{P}_1 after grounding.

$$\begin{aligned} \mathbf{P}_1 \quad r_1 & : [0.8, 1] \text{disease}(p_3) :- \text{test_pos}(p_3). \\ r_2 & : [0.8, 1] \text{disease}(p_4) :- \text{test_pos}(p_4). \\ r_3 & : [0.6, 1] \text{disease}(p_3) :- \text{contact}(p_3, p_4) \wedge \text{disease}(p_4). \\ r_4 & : [0.6, 1] \text{disease}(p_4) :- \text{contact}(p_4, p_3) \wedge \text{disease}(p_3). \\ r_5 & : [1, 1] \text{test_pos}(p_3). \\ r_6 & : [1, 1] \text{test_pos}(p_4). \\ r_7 & : [1, 1] \text{contact}(p_3, p_4). \\ r_8 & : [1, 1] \text{contact}(p_4, p_3). \end{aligned}$$

There are five atom cliques in \mathbf{P}_1 : $\{\text{test_pos}(p_3)\}$, $\{\text{test_pos}(p_4)\}$, $\{\text{contact}(p_3, p_4)\}$, $\{\text{contact}(p_4, p_3)\}$ and $\{\text{disease}(p_3), \text{disease}(p_4)\}$. The dependency graph and the pp-DAGs are shown in Figure 2 and Figure 3, respectively. \square

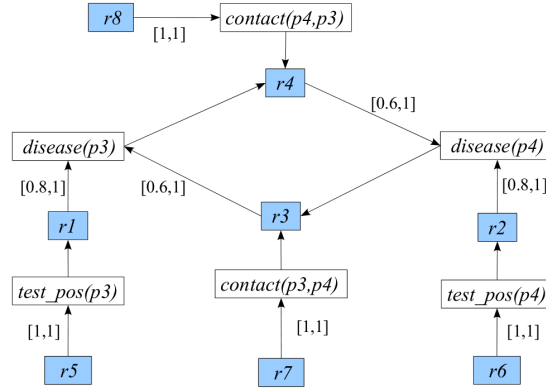


Fig. 2. The dependency graph for Example 1

Definition 14. Let \mathcal{G} be a pp-DAG of \mathbf{P} for the atom A . Another pp-DAG \mathcal{G}' of \mathbf{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} .

\square

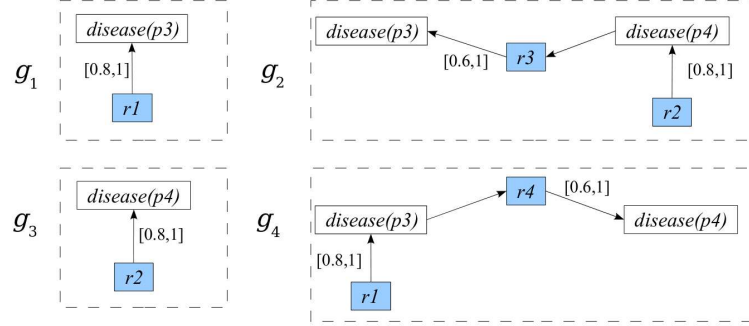


Fig. 3. The pp-DAGs in Example 1: g_1, g_2 are for $disease(p3)$, g_3, g_4 are for $disease(p4)$.

In Example 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.

Definition 15. *Decyclification* of \mathbf{P} , denoted $acyclic(\mathbf{P})$, is obtained from \mathbf{P} as follows. Let \mathcal{S} be the set of new atoms labeled with pp-DAGs of \mathbf{P} :

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

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

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

where $A_i \in clique(A_0), 1 \leq i \leq n, D_j \notin clique(A_0), 1 \leq j \leq m$, replace R with the rule

$$[v, w] A_0 \text{ :- } ID_R$$

where ID_R is the proposition that identifies R , plus, for every list $\mathcal{G}_0, \dots, \mathcal{G}_n$ 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 $\mathcal{G}_0, 1 \leq j \leq n$,

we add the rules of the form

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

$$[1, 1] ID_R \text{ :- } A_1^{\mathcal{G}_1}, \dots, A_k^{\mathcal{G}_k}, \overline{A_{k+1}^{\mathcal{G}_{k+1}}}, \dots, \overline{A_n^{\mathcal{G}_n}}, D_1, \dots, D_l, \overline{D_{l+1}}, \dots, \overline{D_m}. \quad \square$$

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 16. For a (possibly cyclic) blp \mathbf{P} , $\tilde{m}_{\mathbf{P}}$ is a support function for \mathbf{P} such that for any $I \in T\mathcal{V}al(\mathbf{P})$,

$$\tilde{m}_{\mathbf{P}}(I) = \sum_{I' \in T\mathcal{V}al(\text{acyclic}(\mathbf{P})), I'|_{B_{\mathbf{P}}} = I} \hat{m}_{\text{acyclic}(\mathbf{P})}(I')$$

The **model** of \mathbf{P} is a belief function for \mathbf{P} , $\text{model}_c(\mathbf{P})$, such that for any formula F in $\mathcal{B}ool(B_{\mathbf{P}})$, $\text{model}_c(\mathbf{P})(F) = \text{model}(\text{acyclic}(\mathbf{P}))(F)$. (For acyclic blps, \hat{m} and model are defined in Definitions 9 and 10.) \square

In other words, the semantics for acyclic BLP can be applied on $\text{acyclic}(\mathbf{P})$ to compute the model of \mathbf{P} . In practice, the query answering algorithm in [27] can be used on $\text{acyclic}(\mathbf{P})$ to compute $\text{model}_c(\mathbf{P})(F)$ for any F in $\mathcal{B}ool(B_{\mathbf{P}})$.

Example 2. (Example 1 continued.) Applying the decyclification transformation on \mathbf{P}_1 , we get the following acyclic blp \mathbf{P}'_1 .

$$\begin{array}{l} \mathbf{P}'_1 \quad [0.8, 1] \text{disease}(p_3) \text{ :- } r_1. \\ \quad [1, 1] r_1 \text{ :- } \text{test_pos}(p_3). \\ [0.8, 1] \text{disease}(p_4) \text{ :- } r_2. \\ \quad [1, 1] r_2 \text{ :- } \text{test_pos}(p_4). \\ [0.6, 1] \text{disease}(p_3) \text{ :- } r_3. \\ \quad [1, 1] r_3 \text{ :- } \text{contact}(p_3, p_4) \wedge \text{disease}^{g_2}(p_4). \\ [0.6, 1] \text{disease}(p_4) \text{ :- } r_4. \\ \quad [1, 1] r_4 \text{ :- } \text{contact}(p_4, p_3) \wedge \text{disease}^{g_1}(p_3). \\ [0.8, 1] \text{disease}^{g_1}(p_3) \text{ :- } \text{test_pos}(p_3). \\ [0.8, 1] \text{disease}^{g_2}(p_4) \text{ :- } \text{test_pos}(p_4). \\ [0.6, 1] \text{disease}^{g_3}(p_3) \text{ :- } \text{contact}(p_3, p_4) \wedge \text{disease}^{g_2}(p_4). \\ [0.6, 1] \text{disease}^{g_4}(p_4) \text{ :- } \text{contact}(p_4, p_3) \wedge \text{disease}^{g_1}(p_3). \\ \quad [1, 1] \text{test_pos}(p_3). \\ \quad [1, 1] \text{test_pos}(p_4). \\ \quad [1, 1] \text{contact}(p_3, p_4). \\ \quad [1, 1] \text{contact}(p_4, p_3). \end{array}$$

If the combination function associated with *disease* is Φ^{DS} , we get the following conclusions: $\text{bel}(\text{disease}(p_3)) = \text{bel}(\text{disease}(p_4)) = 0.896$. 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 $\text{disease}(p_3)$ and $\text{disease}(p_4)$. Otherwise, $\text{bel}(\text{disease}(p_3))$ and $\text{bel}(\text{disease}(p_4))$ would have ended up close to 1 via these self-supporting feedback loops. \square

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

Theorem 2 (Backward Compatibility). *Let \mathbf{P} be an acyclic blp, and $F \in \text{Bool}(B_{\mathbf{P}})$. Then $\tilde{m}_{\mathbf{P}} = \hat{m}_{\mathbf{P}}$ and $\text{model}_c(\mathbf{P})(F) = \text{model}(\mathbf{P})(F)$. \square*

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 3 (Self-support). *Let \mathbf{P} be a (possibly cyclic) blp, and A an atom in $B_{\mathbf{P}}$. Let \mathbf{P}'_A be the blp obtained from \mathbf{P} by deleting all the rules that contain A in their bodies. Then $\text{model}_c(\mathbf{P})(A) = \text{model}_c(\mathbf{P}'_A)(A)$. \square*

Example 3. (Example 2 continued.) Let \mathbf{P}_2 be $\mathbf{P}_1 - \{R_4\}$ where \mathbf{P}_1 is the program in Example 1 and R_4 is the fourth rule of \mathbf{P}_1 . In the BLP semantics, \mathbf{P}_2 , \mathbf{P}_1 and \mathbf{P}'_1 (the decyclification of \mathbf{P}_1 , as shown in Example 2) give the same amount of support in $\text{disease}(p_3)$. \square

4.2 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 7.

Definition 17. *Let \mathbf{P} be a blp, I a truth valuation, and \mathcal{C} an atom clique in the dependency graph of \mathbf{P} . $\mathbf{P}(\mathcal{C})$ is defined as the set of rules in \mathbf{P} that has an atom from \mathcal{C} in the head. \mathbf{P} 's **reduct under I with respect to \mathcal{C}** , denoted $\mathbf{P}_I(\mathcal{C})$,³ is obtained from $\mathbf{P}(\mathcal{C})$ 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 \mathbf{P} has a fact of the form $[a, b]A$, then delete all the other rules with A in head.

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

Definition 18. *Suppose I is a truth valuation over a set of atoms β and $\alpha \subseteq \beta$. We define the **restriction** of I to α , denoted $I|_{\alpha}$, to be the truth valuation over α such that $\forall X \in \alpha, I|_{\alpha}(X) = I(X)$.*

We write $I|_{\alpha} = \tau$ if $I(X) = \tau$ for all $X \in \alpha$, where $\tau \in \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$. \square

Let I_{\emptyset} be an *empty* truth valuation $\emptyset \longrightarrow \{\mathbf{t}, \mathbf{f}, \mathbf{u}\}$, i.e., a trivial valuation with an empty domain. Since I_{\emptyset} is the only truth valuation over \emptyset , it follows that $\mathcal{TV}al(\emptyset) = \{I_{\emptyset}\}$ and $\forall \alpha \forall I \in \mathcal{TV}al(\alpha) I|_{\emptyset} = I_{\emptyset}$. We also define a special support function $m_{\emptyset} : \mathcal{TV}al(\emptyset) \rightarrow [0, 1]$ to be $m_{\emptyset}(I_{\emptyset}) = 1$; it is the only support function for \emptyset .

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

³ Note that the definitions of $\mathbf{P}(\mathcal{C})$ and $\mathbf{P}_I(\mathcal{C})$ here is different from the definitions of $\mathbf{P}(X)$ and $\mathbf{P}_I(X)$ (in Definition 7): \mathcal{C} is an atom clique, while X is an atom.

Definition 19. Let \mathbf{P} be a blp with n atom cliques and a clique ordering Ord . Let $\mathcal{C}_1, \dots, \mathcal{C}_n$ be the atom cliques of \mathbf{P} , such that $\text{Ord}(\mathcal{C}_i) = i, 1 \leq i \leq n$, and let $\alpha_0 = \emptyset, \alpha_i = \mathcal{C}_1 \cup \dots \cup \mathcal{C}_i, 1 \leq i \leq n$.

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

$$\hat{T}_{\mathbf{P}, \text{Ord}}(m)(I) = m(I|_{\alpha_k}) \cdot \tilde{m}_Q(I|_{\mathcal{C}_{k+1}}) \quad (1)$$

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

Theorem 4 (Equivalence of fixpoint and transformational semantics).

Let \mathbf{P} be a blp with n atom cliques and Ord a clique ordering of \mathbf{P} . Beginning with $m_0 = m_\emptyset$, let m_k be $\hat{T}_{\mathbf{P}, \text{Ord}}^{\uparrow k}(m_0), k = 0, 1, \dots, n$. The support function m_n coincides with $\tilde{m}_{\mathbf{P}}$. \square

The above theorem shows that the fixpoint semantics does not depend on the choice of the clique ordering in \mathbf{P} and that this semantics coincides with the transformational semantics of Section 4.

Next, we will show that the computation in (1) can be simplified for a special class of cyclic blps.

Definition 20. Let \mathbf{P} be a blp with n atom cliques and a clique ordering Ord . Let $\mathcal{C}_1, \dots, \mathcal{C}_n$ be the atom cliques of \mathbf{P} , such that $\text{Ord}(\mathcal{C}_i) = i, 1 \leq i \leq n$, and let $\alpha_0 = \emptyset, \alpha_i = \mathcal{C}_1 \cup \dots \cup \mathcal{C}_i, 1 \leq i \leq n$. Also let $m_k = \hat{T}_{\mathbf{P}, \text{Ord}}^{\uparrow k}(m_\emptyset), k = 0, 1, \dots, n$.

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

If \mathbf{P} is modularly acyclic, it follows from Theorem 2 that $\tilde{m}_Q(I|_{\mathcal{C}_{k+1}})$ in (1) is equivalent to $\hat{m}_Q(I|_{\mathcal{C}_{k+1}})$, where Q is $\mathbf{P}_{I|_{\alpha_k}}(\mathcal{C}_{k+1})$, as defined in Definition 19. (Indeed, it follows from Definition 20 that Q is acyclic if \mathbf{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 1. Let \mathbf{P} be a blp. If in every cycle in \mathbf{P} (i.e., in every cycle in the dependency graph of \mathbf{P}), there is a rule R such that some atom in R 's body is not in the head of any rule, then \mathbf{P} is modularly acyclic. \square

Example 4. Let us return to the diagnosis case for p_1 and p_2 in Section 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 \mathbf{P}_3 is

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

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 1 that \mathbf{P}_3 is modularly acyclic. The belief in $disease(p_1)$ is 0.8 and so is the belief in $disease(p_2)$. \square

More interestingly, a blp can be modularly acyclic even when the condition in Proposition 1 is not satisfied, as shown in the following example.

Example 5. Consider a blp \mathbf{P}_4

$$\begin{array}{ll} [0.8, 1] a :- d. & [1, 1] d. \\ [0.8, 1] b :- e. & [1, 1] e. \\ [0.6, 1] a :- b \wedge c_1. & [0.5, 0.5] c_1. \\ [0.6, 1] b :- a \wedge c_2. & [1, 1] c_2 :- \bar{c}_1. \end{array}$$

According to Definition 20, \mathbf{P}_4 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. \square

4.3 Discussion

In this section, we will contrast our method with an alternative approach of eliminating cycles by adding time parameters, which is utilized in [6] and implicitly in [22]. Adopting a similar methodology, the program of Example 1 can be transformed to the following blp:

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

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 \mathbf{P}'_1 in Example 2 by our approach. One critical question in the time parameter methodology in [6] is, for a query $q(\cdot)$, at which time point t does $q(\cdot, t)$ yield the correct answer. In [22], 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., $[0.9, 1] p(X, T) :- p(X, T - 1)$.

5 Conclusions

In [26] we introduced a novel logic theory, Belief Logic Programming, for reasoning with uncertainty, which can correlate structural information contained in derivation paths for beliefs. In this paper we extended the previous work to cyclic BLP by defining a transformational and a fixpoint semantics in which self-supported belief is discarded. We also showed that the proposed semantics are backward compatible with the semantics for acyclic BLP [26] and has the expected properties.

For future work, we plan to lift the decyclification transformation to non-ground level and extend the query answering algorithm proposed in [27] to cyclic BLP.

6 Acknowledgment

This work is part of the SILK (Semantic Inference on Large Knowledge) project sponsored by Vulcan, Inc. The author thanks Michael Kifer for the insightful discussions.

References

1. J. F. Baldwin. Evidential support logic programming. *Fuzzy Sets and Systems*, 24(1):1–26, 1987.
2. A. Dekhtyar and V. S. Subrahmanian. Hybrid probabilistic programs. *J. of Logic Programming*, 43:391–405, 1997.
3. A. P. Dempster. Upper and lower probabilities induced by a multi-valued mapping. *Ann. Mathematical Statistics*, 38, 1967.
4. Z. Ding, Y. Peng, and R. Pan. BayesOWL: Uncertainty modeling in semantic web ontologies. *Studies in Fuzziness and Soft Computing*, 204:3–29, 2006.
5. M. H. Van Emden. Quantitative deduction and its fixpoint theory. *J. of Logic Programming*, 3(1):37–53, 1986.
6. S. Glesner and D. Koller. Constructing flexible dynamic belief networks from first-order probabilistic knowledge bases. In *European Conf. on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 217–226, 1995.

7. B.N. Grosz. A courteous compiler from generalized courteous logic programs to ordinary logic programs. Technical Report Supplementary Update Follow-On to RC 21472, IBM, July 1999.
8. K. Kersting and L. De Raedt. Bayesian logic programs. Technical report, Albert-Ludwigs University at Freiburg, 2001.
9. M. Kifer and A. Li. On the semantics of rule-based expert systems with uncertainty. In *ICDT '88: Intl. Conf. on Database Theory*, pages 102–117, London, UK, 1988. Springer-Verlag.
10. M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *J. of Logic Programming*, 12(3,4):335–367, 1992.
11. L. V. S. Lakshmanan and N. Shiri. A parametric approach to deductive databases with uncertainty. *IEEE Trans. on Knowledge and Data Engineering*, 13(4):554–570, 2001.
12. T. Lukasiewicz. Probabilistic logic programming under inheritance with overriding. In *Annual Conf. on Uncertainty in Artificial Intelligence (UAI-01)*, pages 329–336, San Francisco, CA, 2001. Morgan Kaufmann Publishers.
13. T. Lukasiewicz. Probabilistic logic programming with conditional constraints. *ACM Trans. on Computational Logic*, 2(3):289–339, 2001.
14. T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *Journal of Web Semantics*, 6(4):291–308, 2008.
15. R. T. Ng. Reasoning with uncertainty in deductive databases and logic programs. *Intl. Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 5(3):261–316, 1997.
16. R. T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1992.
17. R. T. Ng and V. S. Subrahmanian. A semantical framework for supporting subjective probabilities in deductive databases. *J. of Automated Reasoning*, 10(2):191–235, 1993.
18. D. Poole. The independent choice logic and beyond. In *Probabilistic Inductive Logic Programming*, pages 222–243, 2008.
19. L. De Raedt and K. Kersting. Probabilistic inductive logic programming. In *Probabilistic Inductive Logic Programming*, pages 1–27, 2008.
20. M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2):107–136, 2006.
21. G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.
22. Y. Shen. Reasoning with recursive loops under the plp framework. *ACM Trans. Comput. Logic*, 9(4):1–31, 2008.
23. G. Stoilos, G. Stamou, V. Tzouvaras, J. Z. Pan, and I. Horrocks. Fuzzy owl: Uncertainty and the semantic web. In *In Proc. of the International Workshop on OWL: Experiences and Directions*, 2005.
24. V. S. Subrahmanian. On the semantics of quantitative logic programs. In *SLP*, pages 173–182, 1987.
25. H. Wan, B.N. Grosz, M. Kifer, P. Fodor, and S. Liang. Logic programming with defaults and argumentation theories. In *Intl. Conf. on Logic Programming*, 2009.
26. H. Wan and M. Kifer. Belief logic programming: Uncertainty reasoning with correlation of evidence. In *Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 2009.
27. H. Wan and M. Kifer. Query answering in belief logic programming. In *Intl. Conf. on Scalable Uncertainty Management (SUM)*, 2009.
28. H. Wan and M. Kifer. Technical report: Belief logic programming. Technical report, Stony Brook University, 2009. http://www.cs.sunysb.edu/~hwan/BLP_TR.html.