Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 248 (2009) 67–82

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Qualified Logic Programming with Bivalued Predicates [⋆]

Mario Rodríguez-Artalejo[1] Carlos A. Romero-Díaz[2]

Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid
Madrid, Spain

Abstract

Research on the field of uncertainty in logic programming has evolved during the last 25 years. In a recent paper [13] we have revised a classical approach by van Emden's to *Quantitative Logic Programming* [19], generalizing it to a generic scheme QLP($\mathcal{D}$) for so-called *Qualified Logic Programming* over a parametrically given domain $\mathcal{D}$, whose elements play the role of generalized truth values and can be used to qualify logical assertions. In this paper we present an extension of QLP($\mathcal{D}$) yielding a more expressive scheme BQLP($\mathcal{D}$), which supports a simple kind of negation based on bivalued predicates and allows *threshold constraints* in clause bodies in order to impose lower bounds to the qualifications computed by program clauses. The new scheme has a rigorous declarative semantics and a sound and strongly complete goal resolution procedure which can be efficiently implemented using constraint logic programming technology.

Keywords: Bivalued Predicates, Qualification Domains, Qualified Logic Programming, Threshold Constraints.

1 Introduction

The historical evolution of research on uncertainty in logic programming has been described in a recent recollection by V. S. Subrahmanian [18] and briefly summarized in the introductory section of [13]. Early approaches include the quantitative treatment of uncertainty in the spirit of fuzzy logic, as in van Emden's classical paper [19], and two subsequent papers by Subrahmanian [16,17]. The main contribution of [19] was a rigorous declarative semantics for a LP language with program clauses of the form $A \leftarrow f - \overline{B}$, where the head A is an atom, the body $\overline{B}$ is a conjunction of atoms, and the so-called *attenuation* factor $f \in (0, 1]$ attached to the clause's implication is used to propagate to the head the certainty factor $f \times b$,

⋆ Research partially supported by projects MERIT-FORMS (TIN2005-09027-C03-03) and PROMESAS-CAM(S-0505/TIC/0407)

1 Email: mario@sip.ucm.es

2 Email: cromdia@fdi.ucm.es

doi:10.1016/j.entcs.2009.07.060

where b is the minimum of the certainty factors previously computed for the various atoms occurring in the body. The papers [16,17] proposed to use a special lattice $\mathcal{S}$ in place of the lattice of the real numbers in the interval $[0, 1]$ under their natural ordering. $\mathcal{S}$ includes two isomorphic copies of $[0, 1]$ whose elements are incomparable under $\mathcal{S}$'s ordering and can be separately used to represent degrees of *truth* and *falsity*, respectively, thus enabling a simple treatment of negation. Other main contributions of [16,17] were the introduction of annotated program clauses and goals (later generalized to a much more expressive framework in [6]) and the introduction of goal solving procedures more convenient and powerful than those given in [19].

In a recent paper [13] we have revised the approach to *Quantitative Logic Programming* (QLP for short) in [19], generalizing it to a generic scheme QLP($\mathcal{D}$) for so-called *Qualified Logic Programming* over a parametrically given domain $\mathcal{D}$, which must be a lattice satisfying certain natural axioms. The class of qualification domains includes the lattice $[0, 1]$ used in [19] as well as other lattices whose elements can be used to qualify logical assertions in other ways. In this paper we present an extension of QLP($\mathcal{D}$) to a more expressive scheme BQLP($\mathcal{D}$) which provides two main novelties w.r.t. [13]. Firstly, so-called *threshold constraints* are used to impose lower bounds to the qualifications computed for the individual atoms in clause bodies, with the aim of preventing inferences from insufficiently qualified premises. Secondly, a simple kind of negation is supported by the use of *marked atoms* $A \sharp \mathtt{tt}$, and $A \sharp \mathtt{ff}$, where $\mathtt{tt}$ and $\mathtt{ff}$ stand for the two classical truth values. Marked atoms can be viewed as logical assertions associated to bivalued predicates, and their degree of validity can be qualified by elements of a parametrically given qualification domain $\mathcal{D}$. In particular, if the given $\mathcal{D}$ is the lattice $[0, 1]$, qualifying marked atoms with numeric degrees $d \in (0, 1]$ is similar to annotating atoms with values of the lattice $\mathcal{S}$, as proposed in [16,17]. Nevertheless, the present approach differs from [16,17] in two major aspects: a) our framework can operate with any parametrically given qualification domain $\mathcal{D}$ instead of the fixed lattice $\mathcal{S}$; and b) we use attenuated clauses with threshold constraints, whose expressivity is quite different from that of the annotated clauses used in [16,17]. More comparisons to related work can be found in Section 6 and in the concluding section of [13].

As it was the case for QLP($\mathcal{D}$), the new scheme BQLP($\mathcal{D}$) has a rigorous declarative semantics and a sound and strongly complete goal resolution procedure which can be efficiently implemented using constraint logic programming technology. The rest of this paper is structured as follows: Section 2 presents the axioms for qualification domains $\mathcal{D}$ and its basic properties, including closure under cartesian product. The axioms have been revised w.r.t. [13] with the aim of enabling the technical treatment of threshold constraints in subsequent sections. Section 3 presents the syntax and declarative semantics of the BQLP($\mathcal{D}$) scheme. Section 4 presents a goal solving procedure for BQLP($\mathcal{D}$) along with its soundness and strong completeness properties. Section 5 sketches a general implementation technique for BQLP($\mathcal{D}$) which can be used to implement useful instances of the scheme on top of any system that supports sufficient CLP technology. Finally, Section 6 summarizes conclusions and plans for future work.

2 Qualification Domains and their Properties

By definition, a *Qualification Domain* is any structure $\mathcal{D} = \langle D, \sqsubseteq, \bot, \top, \otimes, \oslash \rangle$ such that:

(i) $\langle D, \sqsubseteq, \bot, \top \rangle$ is a lattice with extreme points $\bot$ and $\top$ w.r.t. the partial ordering $\sqsubseteq$. For given elements $d, e \in D$, we write $d \sqcap e$ for the *greatest lower bound* (*glb*) of d and e and $d \sqcup e$ for the *least upper bound* (*lub*) of d and e. We also write $d \sqsubset e$ as abbreviation for $d \sqsubseteq e \wedge d \neq e$.

(ii) $\otimes : D \times D \longrightarrow D$, called *attenuation operation*, verifies the following axioms:
  (a) $\otimes$ is associative, commutative and monotonic w.r.t. $\sqsubseteq$.
  (b) $\forall d \in D : d \otimes \top = d$.
  (c) $\forall d \in D : d \otimes \bot = \bot$.
  (d) $\forall d, e \in D \setminus \{\bot, \top\} : d \otimes e \sqsubset e$.
  (e) $\forall d, e_1, e_2 \in D : d \otimes (e_1 \sqcap e_2) = d \otimes e_1 \sqcap d \otimes e_2$.

(iii) $\oslash : D \times D \dashrightarrow D$ is a partial operation defined as the inverse of $\otimes$, that satisfies the following axioms: Given $d \in D \setminus \{\bot\}$ and $e, e' \in D$,
  (a) (Inverse) $e \oslash d$ is defined and $e \oslash d = e' \iff e' \otimes d = e$.
  (b) (Polarity) $e \sqsubseteq e'$, $d \sqsupseteq d'$, $e' \oslash d'$ defined $\Longrightarrow e \oslash d \sqsubseteq e' \oslash d'$ also defined.

In the rest of the paper, $\mathcal{D}$ will always denote an arbitrary qualification domain. The axioms stated above are like those in [13] except that no $\oslash$ operation was required there. For any finite $S = \{e_1, e_2, \ldots, e_n\} \subseteq D$, the *glb* of S (noted as $\bigsqcap S$) exists and can be computed as $e_1 \sqcap e_2 \sqcap \cdots \sqcap e_n$ (which reduces to $\top$ in the case $n = 0$). As an easy consequence of the axioms, one gets the identity $d \otimes \bigsqcap S = \bigsqcap \{d \otimes e \mid e \in S\}$. Three interesting instances of qualification domain are shown below.

The Domain of Classical Boolean Values. $\mathcal{B} = (\{0,1\}, \leq, 0, 1, \wedge, \oslash)$, where 0 and 1 stand for the two classical truth values *false* and *true*, $\leq$ is the usual numerical ordering over $\{0,1\}$, $\wedge$ stands for the classical conjunction operation over $\{0,1\}$, and $\oslash$ is defined by the two equations $0 \oslash 1 = 0$ and $1 \oslash 1 = 1$. The instance BQLP($\mathcal{B}$) of our BQLP($\mathcal{D}$) scheme will behave as classical logic programming extended with bivalued predicates.

The Domain of Uncertainty Values. $\mathcal{U} = (\mathrm{U}, \leq, 0, 1, \times, /)$, where $\mathrm{U} = [0,1] = \{d \in \mathbb{R} \mid 0 \leq d \leq 1\}$, $\leq$ is the usual numerical ordering, $\times$ is the multiplication operation, and $/$ is the division operation (with $e/0$ undefined). In this domain, the top element $\top$ is 1 and the greatest lower bound $\bigsqcap S$ of a finite $S \subseteq \mathrm{U}$ is the minimum value min(S), which is 1 if $S = \emptyset$. Therefore, the instance BQLP($\mathcal{U}$) of our BQLP($\mathcal{D}$) scheme will behave as *van Emden*'s QLP extended with bivalued predicates.

The Domain of Weight Values. $\mathcal{W} = (\mathrm{P}, \geq, \infty, 0, +, -)$, where $\mathrm{P} = [0, \infty] = \{d \in \mathbb{R} \cup \{\infty\} \mid d \geq 0\}$, $\geq$ is the reverse of the usual numerical ordering (with $\infty \geq d$ for any $d \in \mathrm{P}$), $+$ is the addition operation (with $\infty + d = d + \infty = \infty$ for any $d \in \mathrm{P}$), and $-$ is the substraction operation (with $e - \infty$ undefined and $\infty - d = \infty$ for $d \neq \infty$). In this domain, the top element $\top$ is 0 and the greatest lower bound $\bigsqcap S$ of a finite $S \subseteq \mathrm{P}$ is the maximum value max(S), which is 0 if $S = \emptyset$. When working in the instance BQLP($\mathcal{W}$) of our BQLP($\mathcal{D}$) scheme, one propagates to a clause head the qualification value $\alpha + b$, where α is the clause's *attenuation factor* and b is the maximum of the qualification values known for the body atoms. Therefore, qualification values in the instance BQLP($\mathcal{W}$) of our BQLP($\mathcal{D}$) scheme behave as a weighted measure of depths of proof trees.

It is easily checked that the axioms of qualification domains are satisfied by $\mathcal{B}$, $\mathcal{U}$ and $\mathcal{W}$. In fact, the axioms have been chosen as a natural generalization of some basic properties satisfied by the ordering $\leq$ and the operations $\times$ and $/$ in $\mathcal{U}$. The following result states some intuitive properties of $\oslash$ as the inverse of $\otimes$. The proof is an easy exercise.

Proposition 2.1 *The following properties hold in any qualification domain $\mathcal{D}$, for any $d, d_1, d_2 \in D \setminus \{\bot\}$ and any $e \in D$:*

(i) $(d \otimes e) \oslash d = (e \otimes d) \oslash d = e$.

(ii) $d \otimes (e \oslash d) = (e \oslash d) \otimes d = e$.

(iii) $(e \oslash d_1) \oslash d_2 = e \oslash (d_1 \otimes d_2) = e \oslash (d_2 \otimes d_1)$. □

Given two qualification domains $\mathcal{D}_i = \langle D_i, \sqsubseteq_i, \bot_i, \top_i, \otimes_i, \oslash_i \rangle$ $(i \in \{1, 2\})$, their *cartesian product* $\mathcal{D}_1 \times \mathcal{D}_2$ is defined as $\mathcal{D} =_{\mathrm{def}} \langle D, \sqsubseteq, \bot, \top, \otimes, \oslash \rangle$, where $D =_{\mathrm{def}} D_1 \times D_2$, the partial ordering $\sqsubseteq$ is defined as $(d_1, d_2) \sqsubseteq (e_1, e_2) \Longleftrightarrow_{\mathrm{def}} d_1 \sqsubseteq_1 e_1$ and $d_2 \sqsubseteq_2 e_2$, $\bot =_{\mathrm{def}} (\bot_1, \bot_2)$, $\top =_{\mathrm{def}} (\top_1, \top_2)$, the attenuation operator $\otimes$ is defined as $(d_1, d_2) \otimes (e_1, e_2) =_{\mathrm{def}} (d_1 \otimes_1 e_1, d_2 \otimes_2 e_2)$ and its inverse $\oslash$ is defined as $(d_1, d_2) \oslash (e_1, e_2) =_{\mathrm{def}} (d_1 \oslash_1 e_1, d_2 \oslash_2 e_2)$. Intuitively, each value (d_1, d_2) belonging to a product domain $\mathcal{D}_1 \times \mathcal{D}_2$ imposes the qualification d_1 *and also* the qualification d_2. The class of the qualification domains is closed under cartesian products, as stated in the following result. The proof is a simple extension of that found in [12], adding the arguments needed for the axioms of the operation $\oslash$.

Proposition 2.2 *The cartesian product $\mathcal{D} = \mathcal{D}_1 \times \mathcal{D}_2$ of two given qualification domains is always another qualification domain.* □

3 Syntax and Semantics of the BQLP($\mathcal{D}$) Scheme

3.1 Signature and Programs

We assume a signature Σ providing free function symbols (a.k.a. constructors) and predicate symbols. *Terms* are built from constructors and variables from a countably infinite set $\mathcal{V}ar$, disjoint from Σ. The set Subst_Σ of all substitutions of terms for variables in $\mathcal{V}ar$ is defined in the usual way. *Atoms* are of the form $p(t_1, \ldots, t_n)$, shortened as $p(\overline{t_n})$ or simply $p(\overline{t})$, where p is a n-ary predicate symbol and t_i are terms. We write At_Σ, called the *open Herbrand base*, for the set of all atoms. We call *marked atom* to $A \sharp v$ where $A \in \mathrm{At}_\Sigma$ and $v \in \{\mathtt{tt}, \mathtt{ff}\}$, and $\mathcal{D}$-*annotated atom* to $A \sharp (v, w)$ where $A \in \mathrm{At}_\Sigma$, $v \in \{\mathtt{tt}, \mathtt{ff}\}$ and $w \in (D \setminus \{\bot\}) \uplus \{?\}$ (note that $\mathtt{tt}, \mathtt{ff} \notin \Sigma$ and $? \notin D$). A marked atom $A \sharp v$ where A is $p(\overline{t})$ is intended as a logical assertion related to a *bivalued predicate* p, and a $\mathcal{D}$-annotated atom

$A \sharp (v, w)$ is intended as the requirement that $A \sharp v$ holds with at least a qualification $e \in D \setminus \{\bot\}$ such that $e \sqsupseteq^? w$, where $e \sqsupseteq^? w \Longleftrightarrow_{\mathrm{def}} w = ?$ or else $w \in D \setminus \{\bot\}$ and $e \sqsupseteq w$. Note that when $w = ?$ no effective constraint is imposed over e. Requirements of the form $e \sqsupseteq^? w$ will be called *threshold constraints* in the rest of the paper.

A BQLP($\mathcal{D}$)-program $\mathcal{P}$ is a finite set of *program rules* of the form $A \sharp v \leftarrow \alpha - B_1 \sharp (v_1, w_1), \ldots, B_k \sharp (v_k, w_k)$ where $A \sharp v$ is a marked atom, $B_i \sharp (v_i, w_i)$ with $1 \leq i \leq k$ are $\mathcal{D}$-annotated atoms and $\alpha \in D \setminus \{\bot\}$. Such a rule is also called an *attenuated definite Horn clause* with *attenuation value* α attached to its implication and *threshold constraints* attached to its atoms bodies as indicated by their annotations. The behavior of a program clause as inference rule will be formalized in Subsection 3.2 below. Roughly, the idea is to propagate an annotation (v, d) to the head atom whenever annotations (v_i, d_i) for all $1 \leq i \leq k$, satisfying their corresponding threshold constraints $d_i \sqsupseteq^? w_i$, are known to hold for the body atoms and $d \sqsubseteq \alpha \otimes \bigsqcap\{d_1, \ldots, d_k\}$.

Example 3.1 The simple programs over the domains $\mathcal{U}$, $\mathcal{U}\times\mathcal{W}$ and $\mathcal{B}$ shown below are not intended as realistic applications but just as illustrations. In each case, the program can be understood as a *knowledge base* given by the facts for the predicates `animal`, `plant`, `human` and `eats`, along with *knowledge inference rules* corresponding to the clauses with non-empty body. Due to the attenuation values attached to clause implications, qualification degrees can decrease when moving from a clause's body to its head. Note the differences between them when the qualification domain varies.

(i) The BQLP($\mathcal{U}$)-program $\mathcal{P}_{\mathcal{U}}$ contains the clauses you can see in Figure 1.

(ii) The BQLP($\mathcal{U}\times\mathcal{W}$)-program $\mathcal{P}_{\mathcal{U}\times\mathcal{W}}$ is similar to $\mathcal{P}_{\mathcal{U}}$, except that the attenuation value `(c,1)` $\in \mathcal{U}\times\mathcal{W}$ replaces the attenuation value `c` $\in \mathcal{U}$ at the implication sign of every clause in $\mathcal{P}_{\mathcal{U}}$ and the $\mathcal{U}\times\mathcal{W}$-annotation `(v,(c,1))` replaces the $\mathcal{U}$-annotation `(v,c)` (where `c` $\in \mathcal{U}$) at every body atom in $\mathcal{P}_{\mathcal{U}}$ (note that the remaining $\mathcal{U}$-annotations `(v,?)` are also valid $\mathcal{U}\times\mathcal{W}$-annotations). Therefore, each clause is now intended to convey the additional information that the depth of a proof tree for the head is 1 plus the maximum depth of proof trees for the atoms in the body.

(iii) The only possible attenuation value in the domain $\mathcal{B}$ is `1`, which conveys no significant information. Therefore, the BQLP($\mathcal{B}$) program $\mathcal{P}_{\mathcal{B}}$ obtained form $\mathcal{P}_{\mathcal{U}}$ by placing `1` as attenuation value at all the clauses is essentially a classical logic program, where the marks `tt` and `ff` can be thought as additional predicate arguments. Due to the left recursion in the clauses for `human` and `eats`, some goals for $\mathcal{P}_{\mathcal{B}}$ have an infinite search space where SLD resolution with a leftmost selection strategy fails to compute some expected answers. For instance, the answer $\{$`X` $\mapsto$ `mother(eve)`, `Y` $\mapsto$ `apple`$\}$ would not be computed for the goal `eats(X,Y)#tt`. However, when solving goals for the qualified programs $\mathcal{P}_{\mathcal{U}}$ and $\mathcal{P}_{\mathcal{U}\times\mathcal{W}}$ using the resolution method presented in Section 4, threshold constraints can be used for pruning the search space, so that even the leftmost

```
cruel(X)#tt <-0.90- human(X)#(tt,?), eats(X,Y)#(tt,?), animal(Y)#(tt,?)
cruel(X)#tt <-0.40- human(X)#(tt,?), eats(X,Y)#(tt,?), plant(Y)#(tt,?)
cruel(X)#ff <-0.90- human(X)#(tt,?), eats(X,Y)#(ff,?), animal(Y)#(tt,?)
cruel(X)#ff <-0.40- human(X)#(tt,?), eats(X,Y)#(ff,?), plant(Y)#(tt,?)

animal(bird)#tt <-1.0-
animal(cat)#tt <-1.0-
plant(oak)#tt <-1.0-
plant(apple)#tt <-1.0-
human(adam)#tt <-1.0-
human(eve)#tt <-1.0-
human(father(X))#tt <-0.90- human(X)#(tt,0.50)
human(mother(X))#tt <-0.90- human(X)#(tt,0.50)

eats(adam,X)#tt <-0.80-
eats(eve,X)#tt <-0.30- animal(X)#(tt,?)
eats(eve,X)#tt <-0.60- plant(X)#(tt,?)
eats(father(X),Y)#tt <-0.80- eats(X,Y)#(tt,0.40)
eats(mother(X),Y)#tt <-0.70- eats(X,Y)#(tt,0.40)
eats(adam,X)#ff <-0.20-
eats(eve,X)#ff <-0.70- animal(X)#(tt,?)
eats(eve,X)#ff <-0.40- plant(X)#(tt,?)
eats(father(X),Y)#ff <-0.80- eats(X,Y)#(ff,0.40)
eats(mother(X),Y)#ff <-0.70- eats(X,Y)#(ff,0.40)
```

Fig. 1. BQLP($\mathcal{U}$) program $\mathcal{P}_\mathcal{U}$

selection strategy leads to successful computations. $\square$

3.2 Declarative Semantics

In order to formalize program semantics we define *qualification values* as pairs (v, d) where $v \in \{\mathtt{tt}, \mathtt{ff}\}$ and $d \in D \setminus \{\bot\}$, and *$\mathcal{D}$-qualified atoms* as $\mathcal{D}$-annotated atoms $A \sharp (v, d)$ such that (v, d) is a qualification value (i.e., $d \neq ?$). The *$\mathcal{D}$-qualified Herbrand base* is defined as the set $\mathrm{At}_\Sigma(\mathcal{D})$ of all $\mathcal{D}$-qualified atoms. The *$\mathcal{D}$-entailment* relation over $\mathrm{At}_\Sigma(\mathcal{D})$ is defined as follows: $A \sharp (v, d) \succcurlyeq_\mathcal{D} A' \sharp (v', d')$ iff there is some substitution θ such that $A' = A\theta$, $v' = v$ and $d' \sqsubseteq d$. Finally, we define an *open Herbrand interpretation* over $\mathcal{D}$ as any subset $\mathcal{I} \subseteq \mathrm{At}_\Sigma(\mathcal{D})$ which is closed under $\mathcal{D}$-entailment. That is, a Herbrand interpretation $\mathcal{I}$ including a given $\mathcal{D}$-qualified atom $A \sharp (v, d)$ must also include all the "instances" $A' \sharp (v', d')$ such that $A \sharp (v, d) \succcurlyeq_\mathcal{D} A' \sharp (v', d')$. From now on we will write $\mathrm{Int}_\Sigma(\mathcal{D})$ for the family of all Herbrand interpretations over $\mathcal{D}$. The following proposition is easy to prove from the definition of an Herbrand interpretation and the definitions of the union and intersection of a family of sets.

Proposition 3.2 *The family* $\mathrm{Int}_\Sigma(\mathcal{D})$ *of all Herbrand interpretation over* $\mathcal{D}$ *is a complete lattice under the inclusion ordering* $\subseteq$*, whose extreme points are* $\mathrm{Int}_\Sigma(\mathcal{D})$ *as maximum and* $\emptyset$ *as minimum. Moreover, given any family of interpretations* $I \subseteq \mathrm{Int}_\Sigma(\mathcal{D})$*, its* lub *and* glb *are* $\bigsqcup I = \bigcup\{\mathcal{I} \in \mathrm{Int}_\Sigma(\mathcal{D}) \mid \mathcal{I} \in I\}$ *and* $\bigsqcap I = \bigcap\{\mathcal{I} \in$

$\mathrm{Int}_\Sigma(\mathcal{D}) \mid \mathcal{I} \in I\}$, *respectively.* □

Let C be any clause $A \sharp v \leftarrow \alpha - B_1 \sharp (v_1, w_1), \ldots, B_k \sharp (v_k, w_k)$ in the program $\mathcal{P}$ and $\mathcal{I} \in \mathrm{Int}_\Sigma(\mathcal{D})$ any interpretation over $\mathcal{D}$. We say that $\mathcal{I}$ is a *model* of C (and write $\mathcal{I} \models C$) iff for any substitution θ, $\mathcal{I} \models C\theta$. Assuming $C\theta \equiv A' \sharp v \leftarrow \alpha - B'_1 \sharp (v_1, w_1), \ldots, B'_k \sharp (v_k, w_k)$, then $\mathcal{I} \models C\theta$ iff for every $d_1, \ldots, d_k \in D \setminus \{\bot\}$ such that $d_i \sqsupseteq^? w_i$ and $B'_i \sharp (v_i, d_i) \in \mathcal{I}$ for all $1 \le i \le k$, one has $A' \sharp (v, d) \in \mathcal{I}$ for the value $d = \alpha \otimes \bigsqcap\{d_i, \ldots, d_k\}$. We say that $\mathcal{I}$ is a model of $\mathcal{P}$ (and write $\mathcal{I} \models \mathcal{P}$) iff $\mathcal{I} \models C$ holds for every clause $C \in \mathcal{P}$.

As in any logic language, we need some technique to infer formulas (in our case, $\mathcal{D}$-qualified atoms) from a given BQLP($\mathcal{D}$)-program $\mathcal{P}$. We consider two alternative ways of formalizing an inference step which goes from the body of a clause to its head: an *interpretation transformer* $\mathrm{T}_\mathcal{P}$ and a qualified variant of Horn Logic, noted as QHL($\mathcal{D}$) and called *Qualified Horn Logic over* $\mathcal{D}$. The interpretation transformer $\mathrm{T}_\mathcal{P} : \mathrm{Int}_\Sigma(\mathcal{D}) \to \mathrm{Int}_\Sigma(\mathcal{D})$ is defined as follows:

$$\begin{aligned}\mathrm{T}_\mathcal{P}(\mathcal{I}) =_{\mathrm{def}} \{&A\theta \sharp (v, d) \mid (A \sharp v \leftarrow \alpha - B_1 \sharp (v_1, w_1), \ldots, B_k \sharp (v_k, w_k)) \in \mathcal{P},\\ &\theta \in \mathrm{Subst}_\Sigma,\ d_i \in D \setminus \{\bot\} \text{ verifying } d_i \sqsupseteq^? w_i\ (1 \le i \le k),\\ &B_i\theta \sharp (v_i, d_i) \in \mathcal{I} \text{ and } d \sqsubseteq \alpha \otimes \textstyle\bigsqcap\{d_1, \ldots, d_k\}\}\end{aligned}$$

The logic QHL($\mathcal{D}$) is defined as a deductive system consisting just of one inference rule QMP($\mathcal{D}$), called *Qualified Modus Ponens* over $\mathcal{D}$. If there are some $(A \sharp v \leftarrow \alpha - B_1 \sharp (v_1, w_1), \ldots, B_k \sharp (v_k, w_k)) \in \mathcal{P}$, some $\theta \in \mathrm{Subst}_\Sigma$ such that $A' = A\theta$ and $B'_i = B_i\theta$ for all $1 \le i \le k$, and some $d_1, \ldots, d_k \in D \setminus \{\bot\}$ such that $d_i \sqsupseteq^? w_i$ for all $1 \le i \le k$, then the following inference step is allowed for any $d \sqsubseteq \alpha \otimes \bigsqcap\{d_1, \ldots, d_k\}$:

$$\frac{B'_1 \sharp (v_1, d_1) \quad \cdots \quad B'_k \sharp (v_k, d_k)}{A' \sharp (v, d)} \quad \mathrm{QMP}(\mathcal{D})$$

We will use the notations $\mathcal{P} \vdash_{\mathrm{QHL}(\mathcal{D})} A \sharp (v, d)$ (resp. $\mathcal{P} \vdash^n_{\mathrm{QHL}(\mathcal{D})} A \sharp (v, d)$) to indicate that $A \sharp (v, d)$ can be inferred from the clauses in program $\mathcal{P}$ in finitely many steps (resp. n steps). Note that QHL($\mathcal{D}$) proofs can be naturally represented as upwards growing *proof trees* with $\mathcal{D}$-qualified atoms at their nodes, each node corresponding to one inference step having the children nodes as premises.

The following proposition collects the main results concerning the declarative semantics of the BQLP($\mathcal{D}$) scheme. A full proof can be developed in analogy to the classical papers [20,1], except that our Herbrand interpretations are open, as first suggested by Clark in [4]. Our use of the QHL($\mathcal{D}$) calculus is obviously related to the classical $\mathrm{T}_\mathcal{P}$ operator, although it has no direct counterpart in the historical papers we are aware of.

Proposition 3.3 *The following assertions hold for any BQLP($\mathcal{D}$) program $\mathcal{P}$:*

(i) $\mathcal{I} \models \mathcal{P} \iff \mathrm{T}_\mathcal{P}(\mathcal{I}) \subseteq \mathcal{I}$.

(ii) $\mathrm{T}_\mathcal{P}$ *is monotonous and continuous.*

(iii) *The least fixpoint* $\mu(\mathrm{T}_{\mathcal{P}})$ *is the least Herbrand model of* $\mathcal{P}$, *noted as* $\mathcal{M}_{\mathcal{P}}$.

(iv) $\mathcal{M}_{\mathcal{P}} = \bigcup_{n\in\mathbb{N}} \mathrm{T}_{\mathcal{P}}{\uparrow}^n\,(\emptyset) = \{A \sharp (v,d) \mid \mathcal{P} \vdash_{\mathrm{QHL}(\mathcal{D})} A \sharp (v,d)\}$. □

Proof (Sketch) Item (1) is easy to prove from the definition of $\mathrm{T}_{\mathcal{P}}$. In item (2), monotonicity ($\mathcal{I} \subseteq \mathcal{J} \Longrightarrow \mathrm{T}_{\mathcal{P}}(\mathcal{I}) \subseteq \mathrm{T}_{\mathcal{P}}(\mathcal{J})$) follows easily from the definition of $\mathrm{T}_{\mathcal{P}}$ and continuity ($\mathrm{T}_{\mathcal{P}}(\bigcup_{n\in\mathbb{N}}\mathcal{I}_n) = \bigcup_{n\in\mathbb{N}} \mathrm{T}_{\mathcal{P}}(\mathcal{I}_n)$ for any chain $\{\mathcal{I}_n \mid n \in \mathbb{N}\} \subseteq \mathrm{Int}_{\Sigma}(\mathcal{D})$ with $\mathcal{I}_n \subseteq \mathcal{I}_{n+1}$ for all $n \in \mathbb{N}$) follows from monotonicity and properties of chains and sets of interpretations. Item (3) follows from (1), (2), Proposition 3.2 and some known properties about lattices. Finally, item (4) follows from proving the two implications $\mathcal{P} \vdash^n_{\mathrm{QHL}(\mathcal{D})} A \sharp (v,d) \Longrightarrow \exists m\,(A \sharp (v,d) \in \mathrm{T}_{\mathcal{P}} \uparrow^m (\emptyset))$ and $A \sharp (v,d) \in \mathrm{T}_{\mathcal{P}} \uparrow^n (\emptyset) \Longrightarrow \exists m\,(\mathcal{P} \vdash^m_{\mathrm{QHL}(\mathcal{D})} A \sharp (v,d))$ by induction on n. □

The following example presents QHL($\mathcal{D}$) proofs related to the programs shown in Example 3.1 above.

Example 3.4

(i) The proof tree displayed below shows that the $\mathcal{U}$-annotated atom at its root can be deduced from $\mathcal{P}_{\mathcal{U}}$ in QHL($\mathcal{U}$). Therefore, the atom belongs to $\mathcal{M}_{\mathcal{P}_{\mathcal{U}}}$.

$$\dfrac{\dfrac{\overline{\texttt{human(eve)\#(tt,1.0)}}}{\texttt{human(mother(eve))\#(tt,0.90)}} \quad \dfrac{\dfrac{\overline{\texttt{animal(cat)\#(tt,1.0)}}}{\texttt{eats(eve,cat)\#(ff,0.70)}}}{\texttt{eats(mother(eve),cat)\#(ff,0.49)}} \quad \overline{\texttt{animal(cat)\#(tt,1.0)}}}{\texttt{cruel(mother(eve))\#(ff,0.25)}}$$

It is easy to find out which clause was used in each inference step. Note that the atom at the root could have been proved for a greater certainty value of up to `0.441`. However, since `0.25` $\leq$ `0.441`, the displayed inference is also correct (albeit less informative). Note also that inferring `eats(mother(eve), cat)#(ff,0.49)` by means of an instance of the last program rule for `eats`, `eats(eve,cat)#(ff,d)` must be proved with some certainty `d` $\geq$ `0.40`, as required by the threshold constraint in the clause. Actually, the inference is allowed because `eats(eve,cat)#(ff,0.70)` can be proved.

(ii) A proof tree quite similar to the previous one, but with different annotations, can be easily built to show that `cruel(mother(eve))#(ff,(0.25,4))` can be deduced from $\mathcal{P}_{\mathcal{U}\times\mathcal{W}}$ in QHL($\mathcal{U}\times\mathcal{W}$). Therefore, this annotated atom belongs to $\mathcal{M}_{\mathcal{P}_{\mathcal{U}\times\mathcal{W}}}$, and it carries information concerning both the certainty degree `0.25` $\in \mathcal{U}$ and the proof tree depth `4` $\in \mathcal{W}$. □

4 Goal Solving by Resolution in BQLP($\mathcal{D}$)

4.1 Goals and Solutions

In classical logic programming a goal is presented as a conjunction of atoms. In our setting, goals include *threshold constraints* intended to impose lower bounds to the qualifications of individual atoms. In the sequel we assume a countably infinite set $\mathcal{W}\!ar$, disjoint from Σ and $\mathcal{V}\!ar$, of qualification variables W intended to take values over $D\backslash\{\bot\}$. We consider *open* $\mathcal{D}$*-annotated atoms* $A \sharp (v,W)$ with $v \in \{\texttt{tt},\texttt{ff}\}$ and $W \in \mathcal{W}\!ar$ and *threshold constraints* $W \sqsupseteq^? \beta$ with $W \in \mathcal{W}\!ar$ and $\beta \in (D\backslash\{\bot\}) \uplus \{?\}$.

Goal resolution will be formalized in the next subsection. It proceeds from an initial goal through intermediate goals until reaching a final solved goal. Initial goals look like: $A_1 \sharp (v_1, W_1), \ldots, A_n \sharp (v_n, W_n) \;\square\; W_1 \sqsupseteq^? \beta_1, \ldots, W_n \sqsupseteq^? \beta_n$, where $W_i \in \mathcal{W}ar$ and $\beta_i \in (D \setminus \{\bot\}) \uplus \{?\}$. Intermediate goals have a more general form, consisting of a composition of three items: a conjunction of open $\mathcal{D}$-annotated atoms $\overline{A}$ waiting to be solved, a substitution $\sigma \in \mathrm{Subst}_\Sigma$ computed in previous steps, and a set of qualification constraints Δ. We consider two kinds of qualification constraints:

(i) $W \sqsupseteq^? \beta$, where $W \in \mathcal{W}ar$ is qualification variable and $\beta \in (D \setminus \{\bot\}) \uplus \{?\}$. This is called a *threshold constraint* for W.

(ii) $W = \alpha \otimes \bigsqcap\{W_1, \ldots, W_k\}$, where $W, W_1, \ldots, W_k \in \mathcal{W}ar$ are qualification variables and $\alpha \in D \setminus \{\bot\}$. This is called a *defining constraint* for W.

In order to understand why these two kinds of constraints are needed, let us anticipate the expected behavior of a resolution step. Given an initial goal including an open $\mathcal{D}$-annotated atom $A \sharp (v, W)$ and a threshold constraint $W \sqsupseteq^? \beta$ for W, a resolution step with a program clause whose head unifies with A and whose attenuation value is $\alpha \in D \setminus \{\bot\}$ will be enabled only if $\alpha \sqsupseteq^? \beta$ (thereby pruning useless parts of the computation search space) and it will lead to a new goal including a defining constraint $W = \alpha \otimes \bigsqcap\{W_1, \ldots, W_k\}$ for W and a threshold constraint $W_i \sqsupseteq^? \beta_i$ for each $1 \le i \le k$, where the new qualification variables W_i correspond to the atoms in the clause's body, and the β_i are computed as a function of the previous lower bound β, the clause's attenuation factor α and the lower bounds w_i present in the threshold constraints of the clause's body.

Let us now present some notations needed for a formal definition of goals. Given a conjunction of open annotated atoms $\overline{A}$ and a set of qualification constraints Δ, we define the following sets of variables:

- $\mathrm{var}(\overline{A}) =_{\mathrm{def}} \bigcup\{\mathrm{var}(A) \mid A \sharp (v, W) \in \overline{A}\}$,
- $\mathrm{war}(\overline{A}) =_{\mathrm{def}} \{W \mid A \sharp (v, W) \in \overline{A}\}$,
- $\mathrm{war}(\Delta)$ as the set of all the qualification variables occurring in Δ, and
- $\mathrm{dom}(\Delta)$ as the set of all $W \in \mathcal{W}ar$ such that W occurs as the left hand side of some qualification constraint in Δ.

We say that Δ is *satisfiable* iff there is some $\omega \in \mathrm{Subst}_\Sigma(\mathcal{D})$ –the set of all the substitutions of qualification values in $D \setminus \{\bot\}$ for variables in $\mathcal{W}ar$– such that ω is a *solution* of Δ –written $\omega \in Sol(\Delta)$–, meaning that ω satisfies every qualification constraint in Δ. We also say that Δ is *admissible* iff it satisfies the following three conditions:

(i) Δ is satisfiable,

(ii) for every $W \in \mathrm{war}(\Delta)$ there exists one and only one constraint for W in Δ (this implies $\mathrm{dom}(\Delta) = \mathrm{war}(\Delta)$), and

(iii) the relation $>_\Delta$, defined by $W >_\Delta W_i$ iff there is some defining constraint $W = \alpha \otimes \bigsqcap\{W_1, \ldots, W_i, \ldots, W_k\}$ in Δ, satisfies that $>^*_\Delta$ is irreflexive.

Finally, we say that Δ is *solved* iff Δ is admissible and only contains defining constraints. Now we are in a position to define *goals* and their *solutions*.

Definition 4.1 (Goals) Given a conjunction of open $\mathcal{D}$-annotated atoms $\overline{A}$, a substitution $\sigma \in \mathrm{Subst}_\Sigma$, and a set of qualification constraints Δ, we say that $G \equiv \overline{A} \ \| \ \sigma \ \| \ \Delta$ is a *goal* iff

(i) $\sigma \in \mathrm{Subst}_\Sigma$ is idempotent and such that $\mathrm{dom}(\sigma) \cap \mathrm{var}(\overline{A}) = \emptyset$,

(ii) Δ is admisible, and

(iii) for every qualification variable in $\mathrm{war}(\overline{A})$ there is one and only one threshold constraint for W in Δ. And there are no more threshold constraints in Δ.

Furthermore, if $\sigma = \epsilon$ (the identity substitution) and Δ contains only threshold constraints, then G is called *initial*; and if $\overline{A}$ is empty and Δ is solved, then G is called *solved*. For any goal G, we also define:

- $\mathrm{var}(G) =_{\mathrm{def}} \mathrm{var}(\overline{A}) \cup \mathrm{dom}(\sigma)$, and
- $\mathrm{war}(G) =_{\mathrm{def}} \mathrm{war}(\overline{A}) \cup \mathrm{dom}(\Delta)$. □

Definition 4.2 (Goal Solutions) A pair of substitutions (θ, ρ) such that $\theta \in \mathrm{Subst}_\Sigma$ and $\rho \in \mathrm{Subst}_\Sigma(\mathcal{D})$ is called a *solution* of a goal $G \equiv \overline{A} \ \| \ \sigma \ \| \ \Delta$ w.r.t. a BQLP($\mathcal{D}$)-program $\mathcal{P}$ iff:

(i) $\theta = \sigma\theta$,

(ii) $\rho \in \mathit{Sol}(\Delta)$, and

(iii) $\mathcal{P} \vdash_{\mathrm{QHL}(\mathcal{D})} A\theta \sharp (v, W\rho)$ for all $A \sharp (v, W) \in \overline{A}$.

In addition, a solution (σ, μ) for a goal G is said to be more general than (or to subsume) another solution (θ, ρ) for the same goal G iff $\sigma \preccurlyeq \theta$ $[\mathrm{var}(G)]$ and $\mu \sqsupseteq \rho$ $[\mathrm{war}(G)]$, where $\sigma \preccurlyeq \theta$ $[\mathrm{var}(G)]$ means that there is some substitution η such that the composition $\sigma\eta$ behaves the same as θ over any variable in the set $\mathrm{var}(G)$ and $\mu \sqsupseteq \rho$ $[\mathrm{war}(G)]$ means that $\mu(W) \sqsupseteq \rho(W)$ holds for any $W \in \mathrm{war}(G)$. □

Any solved goal $G' \equiv \sigma \ \| \ \Delta$ has the *associated solution* (σ, μ) where $\mu = \omega_\Delta$ is the qualification substitution given by Δ such that $\omega_\Delta(W)$ is the qualification value determined by the defining constraints in Δ for all $W \in \mathrm{dom}(\Delta)$ and $\omega_\Delta(W) = \bot$ for any $W \in \mathcal{W}\!ar \setminus \mathrm{dom}(\Delta)$. Note that for any $W \in \mathrm{dom}(\Delta)$ there exists one unique defining constraint $W = \alpha \otimes \bigsqcap\{W_1, \ldots, W_k\}$ for W in Δ and then $\omega_\Delta(W)$ can be recursively computed as $\alpha \otimes \bigsqcap\{\omega_\Delta(W_1), \ldots, \omega_\Delta(W_k)\}$. The solutions associated to solved goals are called *computed answers*. The next example illustrates solutions for goals related to the programs in Example 3.1.

Example 4.3

(i) A possible goal for program $\mathcal{P}_\mathcal{U}$ in Example 3.1 is `eats(father(X),Y)#(ff, W1), human(father(X))#(tt,W2) | W1>=0.4, W2>=0.6`; and a valid solution for it is $\{$`X` $\mapsto$ `eve, Y` $\mapsto$ `bird`$\}$ `|` $\{$`W1` $\mapsto$ `0.5, W2` $\mapsto$ `0.8`$\}$.

(ii) A goal for $\mathcal{P}_{\mathcal{U}\times\mathcal{W}}$ in Example 3.1 may be `eats(X,Y)#(tt,W) | W` $\sqsupseteq$ `(0.5,4)`; and a valid solution is $\{$`X` $\mapsto$ `mother(adam), Y` $\mapsto$ `bird`$\}$ `|` $\{$`W` $\mapsto$ `(0.6,3)`$\}$.

Note that the threshold constraint $W \sqsupseteq (0.5, 4)$ in $\mathcal{U}\times\mathcal{W}$ imposes a qualification value $W = (C, S)$ such that $C \geq 0.5$ and $S \leq 4$. □

4.2 QSLD($\mathcal{D}$) Resolution

As goal solving procedure we propose *Qualified SLD Resolution*, abbreviated as QSLD($\mathcal{D}$), which extends classical *SLD* resolution with qualification constraints over $\mathcal{D}$. We write $G_0 \Vdash_{C_1,\sigma_1} G_1 \Vdash_{C_2,\sigma_2} \cdots \Vdash_{C_n,\sigma_n} G_n$, abbreviated as $G_0 \Vdash^*_\sigma G_n$ with $\sigma = \sigma_1\sigma_2\cdots\sigma_n$, to indicate a computation in n resolution steps starting at goal G_0. One single resolution step is formally defined as follows:

Definition 4.4 (Resolution step) A *resolution step* has the form $G = \overline{L}, A \sharp (v, W), \overline{R} \,[\![\, \sigma \,]\!]\, W \sqsupseteq^? \beta, \Delta \Vdash_{C_1,\sigma_1} G_1 = (\overline{L}, B_1 \sharp (v_1, W_1), \ldots, B_k \sharp (v_k, W_k), \overline{R})\sigma_1 \,[\![\, \sigma\sigma_1 \,]\!]\, \Delta_1$ where $A \sharp (v, W)$ is called the selected atom; $C_1 \equiv (H \sharp v \xleftarrow{\alpha} B_1 \sharp (v_1, w_1), \ldots, B_k \sharp (v_k, w_k)) \in_{\text{var}} \mathcal{P}$ is chosen as a variant of a clause in $\mathcal{P}$ with fresh variables and such that $\alpha \sqsupseteq^? \beta$; σ_1 is a m.g.u. between A and H; $W_1, \ldots, W_k \in \mathcal{W}\!ar$ are fresh qualification variables; and $\Delta_1 \equiv W_1 \sqsupseteq^? \beta_1, \ldots, W_k \sqsupseteq^? \beta_k, W = \alpha \otimes \bigsqcap\{W_1, \ldots, W_k\}, \Delta$, where $\beta_i = \mathit{newThreshold}(\beta, \alpha, w_i)$ is defined as follows for all $1 \leq i \leq k$:

$$\mathit{newThreshold}(\beta, \alpha, w_i) = \begin{cases} \beta \oslash \alpha \sqcup w_i & \text{if } \beta \neq ? \text{ and } w_i \neq ? \\ \beta \oslash \alpha & \text{if } \beta \neq ? \text{ and } w_i = ? \\ w_i & \text{if } \beta = ? \text{ and } w_i \neq ? \\ ? & \text{if } \beta = ? \text{ and } w_i = ? \end{cases}$$

Note that "$W \sqsupseteq^? \beta, \Delta$" represents a set of qualification constraints including the threshold constraint $W \sqsupseteq^? \beta$ plus those in Δ with no particular ordering assumed. From the threshold constraint for W in G and the new constraints in Δ_1 of G_1 (particularly the new defining constraint for W) easily follows that $\alpha \sqsupseteq^? \beta$ must hold, therefore such condition can be required, without loss of completeness, to actually enable the resolution step. Moreover, the values β_i are computed by means of the auxiliary operation *newThreshold* so that the new threshold constraints $W_i \sqsupseteq^? \beta_i$ in conjunction with the defining constraint for W in G_1 imply the threshold constraint $W \sqsupseteq^? \beta$ in G and the threshold constraints $W_i \sqsupseteq^? w_i$ encoded in the body of C_1. For instance, in the case that $\beta \neq$? and $w_i \neq$?, one must have $\alpha \otimes W_i \sqsupseteq \beta$ due to $W \sqsupseteq \beta$ and the defining constraint for W in G_1. But $\alpha \otimes W_i \sqsupseteq \beta$ is equivalent to $W_i \sqsupseteq \beta \oslash \alpha$ (see Proposition 2.1), and $W_i \sqsupseteq \beta \oslash \alpha$ in conjunction with $W_i \sqsupseteq w_i$ yields $W_i \sqsupseteq \beta \oslash \alpha \sqcup w_i$. The other three cases can be argued similarly. □

It is easy to check that G_1 is again a legal goal whenever G is a goal and $G \Vdash_{C_1,\sigma_1} G_1$. Moreover, in the instance BQLP($\mathcal{B}$) all the qualification values and constraints become trivial, so that QSLD($\mathcal{B}$) boils down to classical SLD resolution. The next two theorems relate QSLD($\mathcal{D}$) resolution to the declarative semantics of BQLP($\mathcal{D}$)-programs. The Soundness Theorem 4.5 guarantees that every computed answer is correct in the sense that it is a solution of the given goal. The Strong Completeness Theorem 4.6 ensures that, for any solution of a given goal and any fixed selection strategy, QSLD($\mathcal{D}$) resolution is able to compute an equal, if not better, solution. The proofs are analogous to those given for the QLP($\mathcal{D}$) scheme in [12,13], using inductive techniques similar to those presented in [15] for classical *SLD* resolution. Example 4.7 below illustrates the Completeness Theorem.

Theorem 4.5 (Soundness) *Assume $G_0 \Vdash^* G$ and $G = \sigma \, [\!] \, \Delta$ solved. Let (σ, μ) be the solution associated to G. Then (σ, μ) –called the computed answer– is a solution of G_0.* □

Theorem 4.6 (Strong Completeness) *Assume a given solution (θ, ρ) for G_0 and any fixed strategy for choosing the selected atom at each resolution step. Then there is some computed answer (σ, μ) for G_0 which subsumes (θ, ρ).* □

Example 4.7

(i) The following QSLD($\mathcal{U}$) computation solves the goal for program $\mathcal{P}_\mathcal{U}$ presented in Example 4.3.

```
eats(father(X),Y)#(ff,W1), human(father(X))#(tt,W2) |
     W1 >= 0.4, W2 >= 0.6                              ⊩eats.9,{X↦eve}
eats(eve,Y)#(ff,W3), human(father(eve))#(tt,W2) |
     { X ↦ eve } |
     W1 = 0.8 * min {W3}, W2 >= 0.6, W3 >= 0.4/0.8     ⊩eats.7,{Y↦bird}
animal(bird)#(tt,W4), human(father(eve))#(tt,W2) |
     { X ↦ eve, Y ↦ bird } |
     W1 = 0.8 * min {W3}, W2 >= 0.6,
     W3 = 0.7 * min {W4}, W4 >= (0.4/0.8)/0.7          ⊩animal.1,ε
human(father(eve))#(tt,W2) |
     { X ↦ eve, Y ↦ bird } |
     W1 = 0.8 * min { W3 }, W2 >= 0.6,
     W3 = 0.7 * min {W4}, W4 = 1.0                     ⊩human.3,ε
human(eve)#(tt,W5) |
     { X ↦ eve, Y ↦ bird } |
     W1 = 0.8 * min {W3}, W2 = 0.9 * min {W5},
     W3 = 0.7 * min {W4}, W4 = 1.0, W5 >= 0.6/0.7      ⊩human.2,ε
| { X ↦ eve, Y ↦ bird } |
     W1 = 0.8 * min {W3}, W2 = 0.9 * min {W5},
     W3 = 0.7 * min {W4}, W4 = 1.0, W5 = 1.0
```

Note that the computed answer {`X ↦ eve, Y ↦ bird`} `|` {`W1 ↦ 0.56, W2 ↦ 0.9`} subsumes the solution for the same goal given in Example 4.3, because it has a higher certainty for both atoms.

(ii) Similarly, QSLD($\mathcal{U} \times \mathcal{W}$) resolution can solve the goal `eats(X,Y)#(tt,W) |` `W` $\sqsupseteq$ `(0.5,4)` for $\mathcal{P}_{\mathcal{U}\times\mathcal{W}}$, obtaining a computed answer {`X ↦ mother(adam)`} `|` {`W ↦ (0.56,2)`} which subsumes the solution for the same goal given in Example 4.3. □

5 Towards an Implementation

The implementation technique proposed in [13] for the QLP($\mathcal{D}$) scheme can be easily adapted to BQLP($\mathcal{D}$). Assuming a qualification domain $\mathcal{D}$ and a constraint domain $\mathcal{C}_\mathcal{D}$ such that the qualification constraints used in QSLD($\mathcal{D}$) resolution can be expressed as $\mathcal{C}_\mathcal{D}$ constraints, a translation of a given BQLP($\mathcal{D}$)-program $\mathcal{P}$ with goal G into a CLP($\mathcal{C}_\mathcal{D}$)-program $\mathcal{P}^t$ with goal G^t can be specified in such a way that solving G with QSLD($\mathcal{D}$) resolution using $\mathcal{P}$ corresponds to solving G^t with constrained SLD resolution using $\mathcal{P}^t$ and a solver for $\mathcal{C}_\mathcal{D}$. The translation can be used to develop an implementation of QSLD($\mathcal{D}$) resolution for the BQLP($\mathcal{D}$) language on top of any CLP or CFLP system that supports $\mathcal{C}_\mathcal{D}$ constraints.

The translation of a BQLP($\mathcal{D}$) program works by adding three extra arguments to all predicates and translating each clause independently. Given the BQLP($\mathcal{D}$) clause

$$C \equiv p(\bar{t}) \sharp v \leftarrow\!\alpha\!- q_1(\bar{s}_1) \sharp (v_1, w_1), \ldots, q_k(\bar{s}_k) \sharp (v_k, w_k) \ ,$$

its head is translated as $p(\bar{t}, v, W, B)$, where the new variables W and B correspond, respectively, to W and β in the threshold constraint $W \sqsupseteq^? \beta$ related to an open annotated atom $A \sharp (v, W)$ which could be selected for a QSLD($\mathcal{D}$) resolution step using the clause C. The clause's body is translated with the aim of emulating such a resolution step, and the translated clause becomes:

$$\begin{aligned} C^t \equiv p(\bar{t}, v, W, B) \quad \leftarrow \quad & \alpha \sqsupseteq^? B, \\ & B_1 = \mathit{newThreshold}(B, \alpha, w_1),\ q_1(\bar{s}_1, v_1, W_1, B_1), \\ & \vdots \\ & B_k = \mathit{newThreshold}(B, \alpha, w_k),\ q_k(\bar{s}_k, v_k, W_k, B_k), \\ & W = \alpha \otimes \textstyle\bigsqcap\{W_1, \ldots, W_k\} \ . \end{aligned}$$

The idea for translating goals is similar. Given the initial QLP($\mathcal{D}$) goal

$$G \equiv q_1(\bar{t}_1) \sharp (v_1, W_1), \ldots, q_m(\bar{t}_m) \sharp (v_m, W_m) \ [\!] \ W_1 \sqsupseteq^? \beta_1, \ldots, W_m \sqsupseteq^? \beta_m$$

where $\beta_1, \ldots, \beta_m \in (D \setminus \{\bot\}) \uplus \{?\}$, the translated goal becomes

$$G^t \equiv q_1(\bar{t}_1, v_1, W_1, \beta_1), \ldots, q_m(\bar{t}_m, v_m, W_m, \beta_m) \ .$$

For three particular choices for $\mathcal{D}$, namely $\mathcal{U}$, $\mathcal{W}$ and $\mathcal{U} \times \mathcal{W}$, we have implemented the instance QLP($\mathcal{D}$) on top of the CFLP system $\mathcal{TOY}$ [2], which supports constraint solving over the *real constraint domain* $\mathcal{R}$. The current implementation is expected to be distributed within the $\mathcal{TOY}$ system itself (as well as any further development), but until its next release, a special distribution of $\mathcal{TOY}$ with QLP($\mathcal{D}$) embedded is available at `http://gpd.sip.ucm.es/cromdia/qlpd`. There you will also find specific instructions for its installation and some examples for different instances to try it out. These three prototypes could be easily extended to support the corresponding BQLP($\mathcal{D}$) instances.

6 Conclusions and Future Work

In [13] we had proposed a generic scheme QLP($\mathcal{D}$) for *Qualified Logic Programming* over a parametrically given qualification domain $\mathcal{D}$, which generalized and improved a classical approach by van Emden [19] to *Quantitative Logic Programming.* In this paper, we have presented an extension of QLP($\mathcal{D}$) to a more expressive scheme BQLP($\mathcal{D}$) supporting *threshold constraints* in clause bodies and a simple kind of negation based on bivalued predicates. The new scheme BQLP($\mathcal{D}$) has a rigorous declarative semantics and a sound and strongly complete goal resolution procedure which can be implemented using constraint logic programming technology. As implementation technique, we have proposed a translation of BQLP($\mathcal{D}$) programs and goals into CLP($\mathcal{C}_{\mathcal{D}}$), choosing a constraint domain $\mathcal{C}_{\mathcal{D}}$ able to compute with qualification constraints over $\mathcal{D}$. In our opinion, this implementation technique is efficient because it can support some interesting instances of our scheme (namely, BQLP($\mathcal{U}$), BQLP($\mathcal{W}$) and BQLP($\mathcal{U}\times\mathcal{W}$)) just by solving simple arithmetic constraints and avoiding the costly computation of so-called *reductant clauses* needed in other approaches to logic programming with uncertainty, as e.g. the *GAP framework* [6] or the *multi-adjoint approach* in [8,9].

As it was already the case for QLP($\mathcal{D}$), the BQLP($\mathcal{D}$) scheme improves the semantic results given in [19]. With respect to the alternative to [19] proposed in [16,17], our approach is more general in that it can operate with any parametrically given qualification domain, and our attenuated clauses with threshold constraints in the body have a quite different expressivity in comparison to the simple annotated clauses used in [16,17]. The theory of generalized annotated logic programs (GAP for short) presented more recently in [6] allows to express attenuated clauses, but the comparisons between GAP and QLP($\mathcal{D}$) given in the concluding section of [13] apply *mutatis mutandis* to BQLP($\mathcal{D}$), showing that our scheme has some points of advantage w.r.t. GAP.

An even more recent line of related work is *logic programming with similarity-based unification* [14,7], which can be applied to *flexible data retrieval* problems. In this approach, programs just consist of definite Horn clauses as in classical logic programming, but SLD resolution is modified to work with a generalized unification algorithm, so that a given *similarity relation* (roughly, the fuzzy analogon of an equivalence relation) permits to unify not identical but similar symbols. Unifiers become substitutions paired with a number $d \in (0, 1]$ which measures the degree of similarity between the (not necessarily identical) unified terms or atoms. In recent joint work with Rafael Caballero [3] we have presented an extension SQLP($\mathcal{R}, \mathcal{D}$) of the QLP($\mathcal{D}$) scheme, which supports similarity-based reasoning using any similarity relation $\mathcal{R}$ over any qualification domain $\mathcal{D}$. A main result given in [3] is a semantics preserving translation of SQLP($\mathcal{R}, \mathcal{D}$) programs into QLP($\mathcal{D}$) programs, showing that implementations of QLP($\mathcal{D}$) instances can be used to support similarity-based reasoning. The approach in [3] could be easily extended to accommodate bivalued predicates in the sense of the current paper.

Some more or less close relations to our work can be also found in existing research on fuzzy logic programming. For instance, the approach to Fuzzy Prolog presented in [5] is similar to our approach in using CLP with real arithmetic constraints as an implementation tool, but rather different in other respects, since it uses elements of the Borel algebra over the interval $[0, 1]$ as a sophisticated kind of fuzzy truth values. Some further comparisons between our approach and other approaches to computing with uncertainty and similarity in LP can be found in [3].

We plan future work along several lines. Firstly, we would like to improve our current implementation of QLP($\mathcal{D}$) instances, possibly incorporating bivalued predicates and unification modulo a given similarity relation. We also plan to perform benchmarks in order to check the implementation's performance, in particular the execution overload introduced by adding qualifications to ordinary logic programs. Next, we plan to extend our current schemes for similarity-based qualified LP to a more expressive scheme which supports multiparadigm declarative programming with lazy functions, predicates and constraints. Some work on functional logic programming with similarity-based unification is already available [10,11]. Finally, we would like to test the usefulness of our approach, focusing on applications to solving flexible information retrieval problems.

Acknowledgement

The authors are thankful to their colleagues Paco López and Rafa Caballero for their valuable help concerning bibliography and implementation techniques as well as to the anonymous reviewers whose constructive comments aided us to improve this work.

References

[1] K. R. Apt and M. H. van Emden. Contributions to the theory of logic programming. *Journal of the Association for Computing Machinery (JACM)*, 29(3):841–862, 1982.

[2] P. Arenas, A. J. Fernández, A. Gil, F. J. López-Fraguas, M. Rodríguez-Artalejo, and F. Sáenz-Pérez. $\mathcal{TOY}$, a multiparadigm declarative language. version 2.3.1, 2007. R. Caballero and J. Sánchez (Eds.), Available at `http://toy.sourceforge.net`.

[3] R. Caballero, M. Rodríguez-Artalejo, and C. A. Romero-Díaz. Similarity-based reasoning in qualified logic programming. In *PPDP '08: Proceedings of the 10th international ACM SIGPLAN conference on Principles and Practice of Declarative Programming*, pages 185–194, New York, NY, USA, 2008. ACM.

[4] K. L. Clark. Predicate logic as a computational formalism (res. report doc 79/59). Technical report, Imperial College, Dept. of Computing, London, 1979.

[5] S. Guadarrama, S. Muñoz, and C. Vaucheret. Fuzzy prolog: A new approach using soft constraint propagation. *Fuzzy Sets and Systems*, 144(1):127–150, 2004.

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

[7] V. Loia, S. Senatore, and M. I. Sessa. Similarity-based SLD resolution and its role for web knowledge discovery. *Fuzzy Sets and Systems*, 144(1):151–171, 2004.

[8] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Multi-adjoint logic programming with continuous semantics. In T. Eiter, W. Faber, and M. Truszczyinski, editors, *Logic Programming and Non-Monotonic Reasoning (LPNMR'01)*, volume 2173 of *LNAI*, pages 351–364. Springer Verlag, 2001.

[9] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. A procedural semantics for multi-adjoint logic programming. In P. Brazdil and A. Jorge, editors, *Progress in Artificial Intelligence (EPIA'01)*, volume 2258 of *LNAI*, pages 290–297. Springer Verlag, 2001.

[10] G. Moreno and V. Pascual. Programming with fuzzy logic and mathematical functions. In A. P. I. Bloch and A. Tettamanzi, editors, *Proceedings of the 6th International Workshop on Fuzzy Logic and Applications (WILF'05)*, volume 3849 of *LNAI*, pages 89–98. Springer Verlag, 2006.

[11] G. Moreno and V. Pascual. Formal properties of needed narrowing with similarity relations. *Electronic Notes in Theoretical Computer Science*, 188:21–35, 2007.

[12] M. Rodríguez-Artalejo and C. A. Romero-Díaz. A generic scheme for qualified logic programming (Technical Report SIC-1-08). Technical report, Universidad Complutense, Departamento de Sistemas Informáticos y Computación, Madrid, Spain, 2008.

[13] M. Rodríguez-Artalejo and C. A. Romero-Díaz. Quantitative logic programming revisited. In J. Garrigue and M. Hermenegildo, editors, *Functional and Logic Programming (FLOPS'08)*, volume 4989 of *LNCS*, pages 272–288. Springer Verlag, 2008.

[14] M. I. Sessa. Approximate reasoning by similarity-based SLD resolution. *Theoretical Computer Science*, 275(1-2):389–426, 2002.

[15] R. F. Stärk. A direct proof for the completeness of SLD-resolution. In E. Börger, H. K. Büning, and M. M. Richter, editors, *Proceedings of the 3rd Workshop on Computer Science Logic (CSL'89)*, volume 440 of *LNCS*, pages 382–383. Springer Verlag, 1990.

[16] V. S. Subrahmanian. On the semantics of quantitative logic programs. In *Proceedings of the 4th IEEE Symposium on Logic Programming*, pages 173–182, San Francisco, 1987.

[17] V. S. Subrahmanian. Query processing in quantitative logic programming. In *Proceedings of the 9th International Conference on Automated Deduction*, volume 310 of *LNCS*, pages 81–100, London, UK, 1988. Springer-Verlag.

[18] V. S. Subrahmanian. Uncertainty in logic programming: Some recollections. *Association for Logic Programming Newsletter*, 20(2), 2007.

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

[20] M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the Association for Computing Machinery (JACM)*, 23(4):733–742, 1976.