

A Generic Scheme for Qualified Logic Programming ^{*}

Technical Report SIC-1-08

Mario Rodríguez-Artalejo and Carlos A. Romero-Díaz ^{**}

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

Abstract. Uncertainty in Logic Programming has been investigated since about 25 years, publishing papers dealing with various approaches to semantics and different applications. This report is intended as a first step towards the investigation of *qualified computations* in Constraint Functional Logic Programming, including *uncertain computations* as a particular case. We revise an early proposal, namely van Emden's *Quantitative Logic Programming* [24], and we improve it in two ways. Firstly, we generalize van Emden's *QLP* to a generic scheme $QLP(\mathcal{D})$ parameterized by any given *Qualification Domain* \mathcal{D} , which must be a lattice satisfying certain natural axioms. We present several interesting instances for \mathcal{D} , one of which corresponds to van Emden's *QLP*. Secondly, we generalize van Emden's results by providing stronger ones, concerning both semantics and goal solving. We present *Qualified SLD Resolution* over \mathcal{D} , a sound and strongly complete goal solving procedure for $QLP(\mathcal{D})$, which is applicable to open goals and can be efficiently implemented using *CLP* technology over any constraint domain $\mathcal{C}_{\mathcal{D}}$ able to deal with qualification constraints over \mathcal{D} . We have developed a prototype implementation of some instances of the $QLP(\mathcal{D})$ scheme (including van Emden's *QLP*) on top of the *CFLP* system *TCV*.

Keywords: Quantitative Logic Programming, Qualification Domains, Qualification Constraints.

1 Introduction

The investigation of uncertainty in logic programming has proceeded along various lines during the last 25 years. A recent recollection by V. S. Subrahmanian [23] highlights some phases in the evolution of the topic from the viewpoint of a committed researcher.

Research on the field has dealt with various approaches to semantics, as well as different applications. One of the earliest approaches was *Quantitative Logic Programming*, *QLP* for short. This can be traced back to a paper by Shapiro

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

^{**} E-mail: mario@sip.ucm.es and cromdia@fdi.ucm.es

[19], who proposed to use real numbers in the interval $(0, 1]$ as *certainty factors*, as well as *certainty functions* for propagating certainty factors from the bodies to the heads of program clauses. Subsequently, van Emden [24] considered *QLP* with an *attenuation factor* $f \in (0, 1]$ attached to the implication of each program clause and restricted his attention to the certainty function which propagates to a clause head the certainty factor $f \times b$, where f is the clause's attenuation factor and b is the minimum of the certainty factors known for the body atoms. Van Emden's approach was less general than Shapiro's because of the fixed choice of a particular certainty function, but it allowed to prove more general results on model theoretic and fixpoint semantics, similar to those previously obtained in [25, 1] for classical Logic Programming. Moreover, [24] gave a procedure for computing the certainty of atoms in the least Herbrand model of a given program, by applying an alpha-beta heuristic to the atoms' and/or search trees. This procedure worked only for ground atoms having a finite search tree.

Following these beginnings, logic programming with uncertainty developed in various directions. Subrahmanian [21] proposed an alternative to [24], using a different lattice of numeric values (aiming at a separate representation of certainty degrees for truth and falsity) as well as clauses whose atoms were annotated with values from this lattice. Neither certainty functions nor attenuation factors were used in this approach, which was extended in [22] to provide goal solving procedures enjoying stronger soundness and completeness results. As a brief summary of some significant later contributions let us mention: generalized *annotated logic programs* [11], a quite general framework which will be discussed in more detail in Section 6; semantics based on *bilattices* of generalized truth values with both a 'knowledge' order and a 'truth' order [8]; logic programming with probabilistic semantics and applications to deductive databases [14, 15]; quantitative and probabilistic constraint logic programming and applications to natural language processing [16]; *hybrid probabilistic programs* [5]; probabilistic agent programs [7] and their extension to deal with both time and uncertainty [6]; logic programs with similarity based unification and applications to flexible data retrieval [2, 18, 9, 12]; and functional logic programming with similarity based unification [13].

We are interested in a long-term research project aiming at a generalization of existing work on logic programming with uncertainty. The generalization we plan to develop will operate in two directions: a) extending logic programming languages to more expressive multi-paradigm declarative languages supporting *functions* and *constraints*; and b) generalizing uncertain truth values to so-called *qualification values*, attached to computed answers and intended to measure the degree in which such computed answers satisfy various user's expectations. In this setting, (constraint) logic programming with uncertainty becomes the particular case in which no functional programming features are used and qualification values are just uncertain truth values. As a first step, we present in this report a generalization of the early *QLP* proposal by van Emden [24], which is still appealing because of its neat semantics. Syntactically, our proposal is very close to van Emden's *QLP*: we use qualified definite Horn clauses $A \leftarrow d - \bar{B}$ with

an attenuation value d attached to the implication and no annotations attached to the atoms. However, we improve [24] in the two ways summarized in the abstract: firstly, we replace numeric certainty values (in particular, those playing the role of attenuation factors in program clauses) by qualification values belonging to a parametrically given *Qualification Domain* \mathcal{D} with a lattice structure, which provides abstract operations generalizing the use of \min (minimum) and \times (product) in [24]. In this way we get a *generic scheme* $QLP(\mathcal{D})$. Secondly, we present stronger semantic results and a sound and strongly complete goal solving procedure called *Qualified SLD Resolution* over \mathcal{D} (in symbols, $SLD(\mathcal{D})$), which extends SLD resolution using *annotated atoms* and *qualification constraints* over \mathcal{D} . The $QLP(\mathcal{D})$ scheme enjoys nice semantic properties and has interesting instances that can be efficiently implemented using CLP technology: $QLP(\mathcal{D})$ programs and goals can be easily translated into $CLP(\mathcal{C}_{\mathcal{D}})$ for any choice of a constraint domain $\mathcal{C}_{\mathcal{D}}$ able to compute with qualification constraints over \mathcal{D} .

We have developed a prototype implementation of some instances of the $QLP(\mathcal{D})$ scheme (including van Emden's QLP) on top of the $CFLP$ system \mathcal{TOY} .

After this introduction, the rest of the report is structured as follows: Section 2 presents the axioms for qualification domains \mathcal{D} , showing some basic instances and proving that the class of such domains is closed under cartesian product. Section 3 presents the syntax and declarative semantics of the $QLP(\mathcal{D})$ scheme. Section 4 presents qualified SLD resolution over \mathcal{D} with its soundness and strong completeness properties. Section 5 presents the general implementation technique for $QLP(\mathcal{D})$ that we have used to implement some instances of the scheme (including van Emden's QLP) on top of the $CFLP$ system \mathcal{TOY} . Finally, Section 6 presents our conclusions and plans for future work. Appendix A includes detailed proofs for the main results. Other proofs that have been omitted or sketched can be found in [17] (in Spanish).

2 Qualification Domains

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

1. $\langle \mathcal{D}, \sqsubseteq, \perp, \top \rangle$ is a lattice with extreme points \perp 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 $d \sqsubseteq e \wedge d \neq e$.
2. $\circ : D \times D \rightarrow D$, called *attenuation operation*, verifies the following axioms:
 - (a) \circ is associative, commutative and monotonic w.r.t. \sqsubseteq .
 - (b) $\forall d \in D : d \circ \top = d$.
 - (c) $\forall d \in D : d \circ \perp = \perp$.
 - (d) $\forall d, e \in D \setminus \{\perp, \top\} : d \circ e \sqsubset e$.
 - (e) $\forall d, e_1, e_2 \in D : d \circ (e_1 \sqcap e_2) = d \circ e_1 \sqcap d \circ e_2$.

In the rest of the report, \mathcal{D} will generally denote an arbitrary qualification domain. For any finite $S = \{e_1, e_2, \dots, e_n\} \subseteq D$, the *glb* of S (noted as $\prod S$) exists and can be computed as $e_1 \prod e_2 \prod \dots \prod e_n$ (which reduces to \top in the case $n = 0$). As an easy consequence of the axioms, one gets the identity $d \circ \prod S = \prod \{d \circ e \mid e \in S\}$. We generalize van Emden's *QLP* to a generic scheme $QLP(\mathcal{D})$ which uses qualification values $d \in D \setminus \{\perp\}$ in place of certainty values $d \in (0, 1]$, the *glb* operator \prod in place of the minimum operator *min*, and the attenuation operator \circ in place of the multiplication operator \times . Three interesting instances of qualification domains are shown below.

The Domain of Classical Boolean Values: $\mathcal{B} = (\{0, 1\}, \leq, 0, 1, \wedge)$, where 0 and 1 stand for the two classical truth values *false* and *true*, \leq is the usual numerical ordering over $\{0, 1\}$, and \wedge stands for the classical conjunction operation over $\{0, 1\}$. The instance $QLP(\mathcal{B})$ of our $QLP(\mathcal{D})$ scheme behaves as classical Logic Programming.

The Domain of van Emden's Uncertainty Values: $\mathcal{U} = (U, \leq, 0, 1, \times)$, where $U = [0, 1] = \{d \in \mathbb{R} \mid 0 \leq d \leq 1\}$, \leq is the usual numerical ordering, and \times is the multiplication operation. In this domain, the top element \top is 1 and the greatest lower bound $\prod S$ of a finite $S \subseteq U$ is the minimum value $\min(S)$, which is 1 if $S = \emptyset$. For this reason, the instance $QLP(\mathcal{U})$ of our $QLP(\mathcal{D})$ scheme behaves as van Emden's *QLP*.

The Domain of Weight Values: $\mathcal{W} = (P, \geq, \infty, 0, +)$, where $P = [0, \infty] = \{d \in \mathbb{R} \cup \{\infty\} \mid d \geq 0\}$, \geq is the reverse of the usual numerical ordering (with $\infty \geq d$ for any $d \in P$), and $+$ is the addition operation (with $\infty + d = d + \infty = \infty$ for any $d \in P$). In this domain, the top element \top is 0 and the greatest lower bound $\prod S$ of a finite $S \subseteq P$ is the maximum value $\max(S)$, which is 0 if $S = \emptyset$. When working in the instance $QLP(\mathcal{W})$ of our $QLP(\mathcal{D})$ scheme one propagates to a clause head the qualification value $f + b$, where f 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 $QLP(\mathcal{W})$ of our $QLP(\mathcal{D})$ scheme behave as a weighted measure of the depth 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 operation \times in \mathcal{U} . In general, the values belonging to a qualification domain are intended to qualify logical assertions by measuring the degree in which they satisfy some kind of user's expectations. In this way, one can think of \mathcal{U} values as measuring the degree of truth, \mathcal{W} values as measuring proofs sizes, etc.

Given two qualification domains $\mathcal{D}_i = \langle D_i, \sqsubseteq_i, \perp_i, \top_i, \circ_i \rangle$ ($i \in \{1, 2\}$), their *cartesian product* $\mathcal{D}_1 \times \mathcal{D}_2$ is defined as $\mathcal{D} =_{\text{def}} \langle D, \sqsubseteq, \perp, \top, \circ \rangle$, where $D =_{\text{def}} D_1 \times D_2$, the partial ordering \sqsubseteq is defined as $(d_1, d_2) \sqsubseteq (e_1, e_2) \iff_{\text{def}} d_1 \sqsubseteq_1 e_1$ and $d_2 \sqsubseteq_2 e_2$, $\perp =_{\text{def}} (\perp_1, \perp_2)$, $\top =_{\text{def}} (\top_1, \top_2)$, and the attenuation operator \circ is defined as $(d_1, d_2) \circ (e_1, e_2) =_{\text{def}} (d_1 \circ_1 e_1, d_2 \circ_2 e_2)$. The class of the qualification domains is closed under cartesian products, as stated in the following result.

Proposition 1. *The cartesian product $\mathcal{D} = \mathcal{D}_1 \times \mathcal{D}_2$ of two given qualification domains is always another qualification domain.*

Proof. According to the axiomatic definition of qualification domains, one must prove two items:

1. \mathcal{D} is a lattice with extreme points \perp and \top w.r.t. the partial ordering \sqsubseteq . This is easily checked using the definition of \sqsubseteq in the product domain. In particular, one gets the equalities $(d_1, d_2) \sqcap (e_1, e_2) = (d_1 \sqcap_1 e_1, d_2 \sqcap_2 e_2)$ and $(d_1, d_2) \sqcup (e_1, e_2) = (d_1 \sqcup_1 e_1, d_2 \sqcup_2 e_2)$.
2. \circ satisfies the five axioms required for attenuation operators, i.e.:
 - (a) \circ is associative, commutative and monotonic w.r.t. \sqsubseteq .
 - (b) $\forall (d_1, d_2) \in D_1 \times D_2 : (d_1, d_2) \circ \top = (d_1, d_2)$.
 - (c) $\forall (d_1, d_2) \in D_1 \times D_2 : (d_1, d_2) \circ \perp = \perp$.
 - (d) $\forall (d_1, d_2), (e_1, e_2) \in D_1 \times D_2 \setminus \{\perp, \top\} : (d_1, d_2) \circ (e_1, e_2) \sqsubset (e_1, e_2)$.
 - (e) $\forall (d_1, d_2), (e_1, e_2), (e'_1, e'_2) \in D_1 \times D_2 : (d_1, d_2) \circ ((e_1, e_2) \sqcap (e'_1, e'_2)) = ((d_1, d_2) \circ (e_1, e_2)) \sqcap ((d_1, d_2) \circ (e'_1, e'_2))$.

All these conditions are easily proved, using the hypothesis that both \mathcal{D}_1 and \mathcal{D}_2 are qualification domains as well as the construction of \mathcal{D} as cartesian product of \mathcal{D}_1 and \mathcal{D}_2 . \square

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 . In particular, values (c, d) belonging to the product domain $\mathcal{U} \times \mathcal{W}$ impose two qualifications, namely: a certainty value greater or equal than c and a proof tree with depth less or equal than d . These intuitions indeed correspond to the declarative and operational semantics formally defined in Sections 3 and 4.

3 Syntax and Semantics of $QLP(\mathcal{D})$

3.1 Programs, Interpretations and Models

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{Var} , disjoint from Σ . *Atoms* are of the form $p(t_1, \dots, t_n)$ (abbreviated as $p(\overline{t}_n)$) where p is a n -ary predicate symbol and t_i are terms. We write At_Σ for the set of all the atoms, called the *open Herbrand base*. A $QLP(\mathcal{D})$ program \mathcal{P} is a set of *qualified definite Horn clauses* of the form $A \leftarrow d - \overline{B}$ where A is an atom, \overline{B} a finite conjunction of atoms and $d \in D \setminus \{\perp\}$ is the *attenuation value* attached to the clause's implication. In $QLP(\mathcal{B})$ programs, the only choice for d is 1, standing for *true*, and therefore $QLP(\mathcal{B})$ behaves as classical LP . The following example presents two simple programs over the domains \mathcal{U} and \mathcal{W} . It is not meant as a realistic application, but just as an illustration.

Example 1.

1. The $QLP(\mathcal{U})$ program $\mathcal{P}_\mathcal{U}$ displayed below 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. The clauses for the predicate **human** specify the human beings as

the ancestors of `adam` and `eve`, with the certainty of being an actual human decreasing as one moves back along the ancestors' chain. Therefore, the certainty of being a cruel human also decreases when moving from descendants to ancestors.

```

cruel(X) <-0.90- human(X), eats(X,Y), animal(Y)
cruel(X) <-0.40- human(X), eats(X,Y), plant(Y)

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

eats(adam, X) <-0.80-
eats(eve,X) <-0.30- animal(X)
eats(eve,X) <-0.60- plant(X)
eats(father(X),Y) <-0.80- eats(X,Y)
eats(mother(X),Y) <-0.70- eats(X,Y)

```

2. The $QLP(\mathcal{W})$ program $\mathcal{P}_{\mathcal{W}}$ is very similar to $\mathcal{P}_{\mathcal{U}}$, except that the attenuation value 1 is attached to all the clauses. Therefore, each clause is intended to convey the 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. As we will see, qualification constraints over \mathcal{W} can be used to impose upper bounds to the depths of proof trees when solving goals w.r.t. $\mathcal{P}_{\mathcal{W}}$.

Note that the two programs in this example are different qualified versions of the classical LP program \mathcal{P} obtained by dropping all the annotations. Due to the left recursion in the clauses for the predicates `human` and `eats`, some goals for \mathcal{P} have an infinite search space where SLD resolution with a leftmost selection strategy would fail to compute some expected answers. For instance, the answer $\{X \mapsto \text{mother}(\text{eve}), Y \mapsto \text{apple}\}$ would not be computed for the goal `eats(X,Y)`. However, when solving goals for the qualified programs $\mathcal{P}_{\mathcal{U}}$ and $\mathcal{P}_{\mathcal{W}}$ using the resolution method presented in Section 4, qualification constraints can be used for imposing bounds to the search space, so that even the leftmost selection strategy leads to successful computations. \square

As shown in the example, clauses contain classic atoms in both their head and their body. But for our semantics, we will be interested in not only proving that we can infer an atom for a given program, but proving that we can infer it with qualification greater or equal than some given value. For this reason, we introduce \mathcal{D} -annotated atoms $A \# d$, consisting of an atom A with an attached 'annotation' $d \in D \setminus \{\perp\}$. For use in goals to be solved, we consider also *open annotated atoms* of the form $A \# W$, where W is a *qualification variable* intended to take values over $D \setminus \{\perp\}$. We postulate a countably infinite set $\mathcal{W}ar$ of qualification variables, disjoint from $\mathcal{V}ar$ and Σ .

The *annotated Herbrand base* over \mathcal{D} is defined as the set $At_{\Sigma}(\mathcal{D})$ of all \mathcal{D} -annotated atoms. The *\mathcal{D} -entailment relation* over $At_{\Sigma}(\mathcal{D})$ is defined as follows: $A \# d \succ_{\mathcal{D}} A' \# d'$ iff there is some substitution θ such that $A' = A\theta$ and $d' \sqsubseteq d$. Finally, we define an *open Herbrand interpretation* over \mathcal{D} as any subset $\mathcal{I} \subseteq At_{\Sigma}(\mathcal{D})$ which is closed under \mathcal{D} -entailment. That is, an open Herbrand interpretation \mathcal{I} including a given annotated atom $A \# d$ is required to include all the ‘instances’ $A' \# d'$ such that $A \# d \succ_{\mathcal{D}} A' \# d'$, because we intend to formalize a semantics such that all such instances are valid whenever $A \# d$ is valid.

In the sequel we refer to open Herbrand interpretations just as Herbrand interpretations, and we write $\text{Int}_{\Sigma}(\mathcal{D})$ for the family of all Herbrand interpretations over \mathcal{D} . The following proposition is easy to prove from the definition of a Herbrand interpretation and the definitions of the union and intersection of a family of sets.

Proposition 2. *The family $\text{Int}_{\Sigma}(\mathcal{D})$ of all Herbrand interpretations over \mathcal{D} is a complete lattice under the inclusion ordering \subseteq , whose extreme points are $\text{Int}_{\Sigma}(\mathcal{D})$ as maximum and \emptyset as minimum. Moreover, given any family of interpretations $I \subseteq \text{Int}_{\Sigma}(\mathcal{D})$, its lub and glb are $\bigsqcup I = \bigcup \{\mathcal{I} \in \text{Int}_{\Sigma}(\mathcal{D}) \mid \mathcal{I} \in I\}$ and $\bigsqcap I = \bigcap \{\mathcal{I} \in \text{Int}_{\Sigma}(\mathcal{D}) \mid \mathcal{I} \in I\}$, respectively. \square*

Let C be any clause $A \leftarrow d - B_1, \dots, B_k$ in the program \mathcal{P} , and $\mathcal{I} \in \text{Int}_{\Sigma}(\mathcal{D})$ any interpretation over \mathcal{D} . We say that \mathcal{I} is a *model* of C if and only if for any substitution θ and any qualification values $d_1, \dots, d_k \in D \setminus \{\perp\}$ such that $B_i\theta \# d_i \in \mathcal{I}$ for all $1 \leq i \leq k$, one has $A\theta \# (d \circ \prod\{d_1, \dots, d_k\}) \in \mathcal{I}$. And we say that \mathcal{I} is a model of the *QLP*(\mathcal{D}) program \mathcal{P} (in symbols, $\mathcal{I} \models \mathcal{P}$) if and only if \mathcal{I} is a model of each clause in \mathcal{P} .

3.2 Declarative Semantics

As in any logic language, we need some technique to infer formulas (in our case, \mathcal{D} -annotated atoms) from a given *QLP*(\mathcal{D}) program \mathcal{P} . Following traditional ideas, we consider two alternative ways of formalizing an inference step which goes from the body of a clause to its head: an operator $T_{\mathcal{P}}$ and a qualified variant of Horn Logic, noted as *QHL*(\mathcal{D}) and called *Qualified Horn Logic*. The operator $T_{\mathcal{P}} : \text{Int}_{\Sigma}(\mathcal{D}) \rightarrow \text{Int}_{\Sigma}(\mathcal{D})$ is defined as:

$$T_{\mathcal{P}}(\mathcal{I}) =_{\text{def}} \{A' \# d' \mid (A \leftarrow d - B_1, \dots, B_k) \in \mathcal{P}, \\ \theta \text{ subst.}, B_i\theta \# d_i \in \mathcal{I} \text{ for all } 1 \leq i \leq k, A' = A\theta, \\ d' \in D \setminus \{\perp\}, d' \sqsubseteq d \circ \prod\{d_1, \dots, d_k\}\}$$

Intuitively, we can see that for a given interpretation \mathcal{I} , $T_{\mathcal{P}}(\mathcal{I})$ is the set of those \mathcal{D} -annotated atoms obtained by considering \mathcal{D} -annotated bodies of clause instances that are included in \mathcal{I} and propagating an annotation to the head via the clause’s qualification value.

The logic *QHL*(\mathcal{D}) is defined as a deductive system consisting just of one inference rule *QMP*(\mathcal{D}), called *Qualitative Modus Ponens* over \mathcal{D} . If there are some $(A \leftarrow d - B_1, \dots, B_k) \in \mathcal{P}$, some substitution θ such that $A' = A\theta$ and

$B'_i = B_i\theta$ for all $1 \leq i \leq k$ and $d' \sqsubseteq d \circ \prod\{d_1, \dots, d_k\}$, the following inference step is allowed:

$$\frac{B'_1 \# d_1 \quad \dots \quad B'_k \# d_k}{A' \# d'} \quad \text{QMP}(\mathcal{D})$$

We will use the notations $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} A \# d$ (resp. $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^n A \# d$) to indicate that $A \# d$ can be inferred from the clauses in program \mathcal{P} in finitely many steps (resp. n steps). Note that $\text{QHL}(\mathcal{D})$ proofs can be naturally represented as upwards growing *proof trees* with \mathcal{D} -annotated 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 $\text{QLP}(\mathcal{D})$ scheme. We just sketch some key proof ideas. The full proofs are given in [17]. As in [24], they can be developed in analogy to the classical papers [25, 1], except that our Herbrand interpretations are open, as first suggested by Clark in [4]. Our use of the $\text{QHL}(\mathcal{D})$ calculus is obviously related to the classical $\text{T}_{\mathcal{P}}$ operator, although it has no direct counterpart in the historical papers we are aware of.

Proposition 3. *The following assertions hold for any $\text{QLP}(\mathcal{D})$ program \mathcal{P} :*

1. $\mathcal{I} \models \mathcal{P} \iff \text{T}_{\mathcal{P}}(\mathcal{I}) \subseteq \mathcal{I}$.
2. $\text{T}_{\mathcal{P}}$ is monotonic and continuous.
3. The least fixpoint $\mu(\text{T}_{\mathcal{P}})$ is the least Herbrand model of \mathcal{P} , noted as $\mathcal{M}_{\mathcal{P}}$.
4. $\mathcal{M}_{\mathcal{P}} = \bigcup_{n \in \mathbb{N}} \text{T}_{\mathcal{P}} \uparrow^n (\emptyset) = \{A \# d \mid \mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} A \# d\}$.

Proof (Sketch). Item (1) is easy to prove from the definition of $\text{T}_{\mathcal{P}}$. In item (2), monotonicity ($\mathcal{I} \subseteq \mathcal{J} \implies \text{T}_{\mathcal{P}}(\mathcal{I}) \subseteq \text{T}_{\mathcal{P}}(\mathcal{J})$) follows easily from the definition of $\text{T}_{\mathcal{P}}$ and continuity ($\text{T}_{\mathcal{P}}(\bigcup_{n \in \mathbb{N}} \mathcal{I}_n) = \bigcup_{n \in \mathbb{N}} \text{T}_{\mathcal{P}}(\mathcal{I}_n)$ for any chain $\{\mathcal{I}_n \mid n \in \mathbb{N}\} \subseteq \text{Int}_{\Sigma}(\mathcal{D})$ 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 2 and some known properties about lattices. Finally, item (4) follows from proving the two implications $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^n A \# d \implies \exists m (A \# d \in \text{T}_{\mathcal{P}} \uparrow^m (\emptyset))$ and $A \# d \in \text{T}_{\mathcal{P}} \uparrow^n (\emptyset) \implies \exists m (\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^m A \# d)$ by induction on n . \square

The next example presents proofs deriving annotated atoms that belong to the least models of the programs $\mathcal{P}_{\mathcal{U}}$ and $\mathcal{P}_{\mathcal{V}}$ from Example 1.

Example 2.

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

$$\frac{\frac{\text{human(eve)\#1.0}}{\text{human(mother(eve))\#0.90}} \quad \frac{\frac{\text{animal(bird)\#1.0}}{\text{eats(eve,bird)\#0.30}}}{\text{eats(mother(eve),bird)\#0.21}} \quad \text{animal(bird)\#1.0}}{\text{cruel(mother(eve))\#0.15}}$$

It is easy to see which clause was used in each inference step. Note that the atom at the root could have been proved even with the greater certainty value 0.189. However, since $0.15 \leq 0.189$, the displayed inference it is also correct (albeit less informative).

2. A proof tree quite similar to the previous one, but with different annotations, can be easily built to show that $\text{cruel}(\text{mother}(\text{eve}))\#4$ can be deduced from \mathcal{P}_W in $QHL(W)$. Therefore, this annotated atom belongs to $\mathcal{M}_{\mathcal{P}_W}$. It conveys the information that $\text{cruel}(\text{mother}(\text{eve}))$ has a proof tree of depth 4 w.r.t. to the classical LP program \mathcal{P} obtained by dropping \mathcal{P}_W 's annotations. \square

4 Goal Solving by SLD(\mathcal{D}) Resolution

4.1 Goals and Solutions

In classical logic programming a goal is presented as a conjunction of atoms. In our setting, proving atoms with arbitrary qualifications may be unsatisfactory, since qualification values too close to \perp may not ensure sufficient information. For this reason, we present goals as conjunctions of open \mathcal{D} -annotated atoms and we indicate the minimum qualification value required each of them. Hence initial goals look like: $A_1 \# W_1, \dots, A_n \# W_n \sqcap W_1 \sqsupseteq \beta_1, \dots, W_n \sqsupseteq \beta_n$, where $W_i \in \mathcal{W}ar$ and $\beta_i \in D \setminus \{\perp\}$. Observe that we have annotated all atoms in the goal with qualification variables W_i instead of plain values because we are interested in any solution that satisfies the *qualification constraints* $W_i \sqsupseteq \beta_i$, used to impose lower bounds to the atoms' qualifications.

As explained in the next Subsection, goal resolution proceeds from an initial goal through intermediate goals until reaching a final solved goal. The intermediate goals have a more general form, consisting of a composition of three items: a conjunction of \mathcal{D} -annotated atoms \bar{A} waiting to be solved, a substitution σ computed in previous steps, and a set of qualification constraints Δ . We consider two kinds of qualification constraints:

1. $\alpha \circ W \sqsupseteq \beta$, where $W \in \mathcal{W}ar$ is qualification variable and $\alpha, \beta \in D \setminus \{\perp\}$ are such that $\alpha \sqsupseteq \beta$. This is called a *threshold constraint* for W .
2. $W = d \circ \sqcap\{W_1, \dots, W_k\}$, where $W, W_1, \dots, W_k \in \mathcal{W}ar$ are qualification variables and $d \in D \setminus \{\perp\}$. This is called a *defining constraint* for W .

In order to understand why these two kinds of constraints are needed, think of an annotated atom $A \# W$ within an initial goal which includes also an initial threshold constraint $\top \circ W \sqsupseteq \beta$ (i.e. $W \sqsupseteq \beta$) for W . Applying a resolution step with a program clause whose head unifies with A and whose attenuation value is $d \in D \setminus \{\perp\}$ will lead to a new goal including a defining constraint $W = d \circ \sqcap\{W_1, \dots, W_k\}$ for W and a threshold constraint $d \circ \top \circ W_i \sqsupseteq \beta$ for each $1 \leq i \leq k$, where the new qualification variables W_i correspond to the atoms in the clause's body. This explains the need to introduce defining constraints as well as more general threshold constraints $\alpha \circ W \sqsupseteq \beta$. Intuitively, the values α and β within such constraints play the role of an *upper* and a *lower* bound, respectively. As we will see, our goal solving procedure takes advantage of these bounds for pruning useless parts of the computation search space.

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

- $\text{var}(\bar{A}) =_{\text{def}} \bigcup \{\text{var}(A) \mid A \# W \in \bar{A}\}$.
- $\text{war}(\bar{A}) =_{\text{def}} \bigcup \{W \mid A \# W \in \bar{A}\}$.
- $\text{war}(\Delta)$ as the set of qualification variables that appears in any qualification constraint in Δ .
- $\text{dom}(\Delta)$ as the set of qualification variables that appear in the left hand side of any qualification constraint in Δ .

We also say that Δ is *satisfiable* iff there is some $\omega \in \text{Subst}_{\Sigma}(\mathcal{D})$ –the set of all the substitutions of values in $D \setminus \{\perp\}$ for variables in Var – such that $\omega \in \text{Sol}(\Delta)$, what means that ω satisfies every qualification constraint in Δ , i.e. ω is a *solution* of Δ . Moreover, we say that Δ is *admissible* iff it satisfies the following three conditions:

1. Δ is satisfiable,
2. for every $W \in \text{war}(\Delta)$ there exists one and only one constraint for W in Δ (this implies $\text{dom}(\Delta) = \text{war}(\Delta)$), and
3. the relation $>_{\Delta}$ defined by $W >_{\Delta} W_i$ iff there is some defining constraint $W = \alpha \circ \prod \{W_1, \dots, W_i, \dots, W_k\}$ 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 1 (Goals and its Variables). *Given a conjunction of \mathcal{D} -annotated atoms \bar{A} , a substitution $\sigma \in \text{Subst}_{\Sigma}$ –the set of all substitutions of terms for variables in Var – and a set of qualification constraints Δ , we say that $G \equiv \bar{A} \parallel \sigma \parallel \Delta$ is a goal iff*

- i. $\sigma \in \text{Subst}_{\Sigma}$ is idempotent and such that $\text{dom}(\sigma) \cap \text{var}(\bar{A}) = \emptyset$.*
- ii. Δ is admissible.*
- iii. For every qualification variable in $\text{war}(\bar{A})$ 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) then G is called initial, and if \bar{A} is empty and Δ is solved, then G is called solved. For any goal G , we define the set of variables of G as $\text{var}(G) =_{\text{def}} \text{var}(\bar{A}) \cup \text{dom}(\sigma)$ and the set of qualification variables of G as $\text{war}(G) =_{\text{def}} \text{war}(\bar{A}) \cup \text{dom}(\Delta)$. \square

Definition 2 (Goal Solutions). *A pair of substitutions (θ, ρ) such that $\theta \in \text{Subst}_{\Sigma}$ and $\rho \in \text{Subst}_{\Sigma}(\mathcal{D})$ is called a solution of a goal $G \equiv \bar{A} \parallel \sigma \parallel \Delta$ iff:*

1. $\theta = \sigma\theta$.
2. $\rho \in \text{Sol}(\Delta)$.
3. $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} A\theta \# W\rho$ for all $A \# W \in \bar{A}$.

In addition, a solution (σ, μ) for a goal G is said to be more general than another solution (θ, ρ) for the same goal G (one also says in this case that (θ, ρ) is subsumed by (σ, μ)) iff $\sigma \preceq \theta [\text{var}(G)]$ and $\mu \supseteq \rho [\text{war}(G)]$, where $\sigma \preceq \theta [\text{var}(G)]$ means that there is some substitution η such that the composition $\sigma\eta$ behaves the same as θ over any variable in the set $\text{var}(G)$ and $\mu \supseteq \rho [\text{war}(G)]$ means that $\mu(W) \supseteq \rho(W)$ holds for any $W \in \text{war}(G)$. \square

Any solved goal $G' \equiv \sigma \sqcup \Delta$ 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 $W \in \text{dom}(\Delta)$, and $\omega_\Delta(W) = \perp$ for any $W \in \text{War} \setminus \text{dom}(\Delta)$. Note that for any $W \in \text{dom}(\Delta)$ there exists one unique defining constraint $W = d \circ \sqcap \{W_1, \dots, W_k\}$ for W in Δ , and then $\omega_\Delta(W)$ can be recursively computed as $d \circ \sqcap \{\omega_\Delta(W_1), \dots, \omega_\Delta(W_k)\}$. The solutions associated to solved goals are called *computed answers*.

Example 3.

1. A possible goal for program \mathcal{P}_U in Example 1 is `eats(father(X), Y)#W1, human(father(X))#W2 | W1>=0.4, W2>=0.6`; and a valid solution for it is `{X ↦ adam, Y ↦ apple} | {W1 ↦ 0.50, W2 ↦ 0.75}`.
2. A goal for program \mathcal{P}_W in Example 1 may be `eats(X, Y)#W | W<=5.0`; and a valid solution is `{X ↦ father(adam), Y ↦ apple} | {W ↦ 4.0}`. \square

Note that the goal for \mathcal{P}_U in the previous example imposes lower bounds to the certainties to be computed, while the goal for \mathcal{P}_W imposes an upper bound to the proof depth. In general, goal solving in $QLP(W)$ corresponds to depth-bound goal-golving in classical Logic Programming.

4.2 SLD(\mathcal{D}) Resolution

We propose a sound and strongly complete goal solving procedure called *Qualified SLD Resolution* parameterized over a given qualification domain \mathcal{D} , written as $SLD(\mathcal{D})$, which makes use of *annotated atoms* and *qualification constraints* over \mathcal{D} . The implementation of this goal solving procedure using *CLP* technology will be discussed in the next section. Resolution computations are written $G_0 \Vdash_{C_1, \sigma_1} G_1 \Vdash_{C_2, \sigma_2} \dots \Vdash_{C_n, \sigma_n} G_n$, abbreviated as $G_0 \Vdash_\sigma^* G_n$ with $\sigma = \sigma_1 \sigma_2 \dots \sigma_n$. They are finite sequences of resolution steps $G_{i-1} \Vdash_{C_i, \sigma_i} G_i$, starting with an initial goal G_0 and ending up with a solved goal G_n . One single resolution step is formally defined as follows:

Definition 3 (Resolution step). A resolution step has the form $\bar{L}, A \# W, \bar{R} \sqcup \sigma \sqcup \alpha \circ W \supseteq \beta, \Delta \Vdash_{C_1, \sigma_1} (\bar{L}, B_1 \# W_1, \dots, B_k \# W_k, \bar{R}) \sigma_1 \sqcup \sigma \sigma_1 \sqcup \Delta_1$ where $A \# W$ is the selected atom, $\Delta_1 = d \circ \alpha \circ W_1 \supseteq \beta, \dots, d \circ \alpha \circ W_k \supseteq \beta, W = d \circ \sqcap \{W_1, \dots, W_k\}, \Delta, C_1 \equiv (H \leftarrow d - B_1, \dots, B_k) \in_{\text{var}} \mathcal{P}$ is chosen as a variant of a clause in \mathcal{P} with fresh variables and such that $d \circ \alpha \supseteq \beta$, σ_1 is the m.g.u. between A and H , and $W_1, \dots, W_k \in \text{War}$ are fresh qualification variables. \square

The notation $\alpha \circ W \sqsupseteq \beta, \Delta$ represents a set of qualification constraints including the threshold constraint $\alpha \circ W \sqsupseteq \beta$ plus those in Δ , with no particular ordering assumed. Notice that the condition $d \circ \alpha \sqsupseteq \beta$ is required for the resolution step to be enabled. In this way, threshold constraints $\alpha \circ W \sqsupseteq \beta$ are actively used for pruning parts of the computation search space where no solutions can be found. In the instance of $QLP(\mathcal{B})$ it is easily checked that all the qualification values and constraints become trivial, so that $SLD(\mathcal{B})$ boils down to classical SLD resolution. In the rest of this section we present the main properties of $SLD(\mathcal{D})$ resolution in the general case.

Proposition 4. *If G is a goal and $G_0 \Vdash_{C_1, \sigma_1} G_1$, then G_1 is also a goal.*

Proof (Sketch). Assume a goal G_0 and a $SLD(\mathcal{D})$ resolution step $G_0 \Vdash_{C_1, \sigma_1} G_1$, as in Definition 3. Then G_0 satisfies the conditions required for goals in Definition 1, and we must show that G_1 also satisfies such conditions. This is not difficult to check, using the fact that C_1 has been chosen without variables in common with G_0 . In particular, note that the threshold constraint for W in G_0 is absent in G_1 , which includes a defining constraint for W and threshold constraints for the new qualification variables W_i . \square

The next two theorems are the main theoretical results in this report. The Soundness Theorem 1 guarantees that every computed answer is correct in the sense that it is a solution of the given goal. The Strong Completeness Theorem 2 ensures that, for any solution of a given goal and any fixed selection strategy, $SLD(\mathcal{D})$ resolution is able to compute an equal, if not better, solution. The proofs, given in Appendix A, use inductive techniques similar to those presented in [20] for classical SLD resolution. Example 4 below illustrates the Completeness Theorem.

Theorem 1 (Soundness). *Assume $G_0 \Vdash^* G$ and $G = \sigma \sqcap \Delta$ solved. Let (σ, μ) be the solution associated to G . Then (σ, μ) –called the computed answer– is a solution of G_0 .* \square

Theorem 2 (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) .* \square

Example 4.

1. The following $SLD(\mathcal{U})$ computation solves the goal for program $\mathcal{P}_{\mathcal{U}}$ presented in Example 3:

$$\begin{array}{l}
\text{eats}(\text{father}(X), Y) \# W1, \\
\text{human}(\text{father}(X)) \# W2 \mid \\
W1 \geq 0.4, W2 \geq 0.6 \qquad \qquad \qquad \Vdash_{\text{eats.4}, \{X \mapsto \text{adam}\}} \\
\text{eats}(\text{adam}, Y) \# W3, \\
\text{human}(\text{father}(\text{adam})) \# W2 \mid \{X \mapsto \text{adam}\} \mid \\
W1 = 0.8 * \min\{W3\}, \\
W2 \geq 0.6, 0.8 * W3 \geq 0.4 \qquad \qquad \qquad \Vdash_{\text{eats.1}, \epsilon} \\
\text{human}(\text{father}(\text{adam})) \# W2 \mid \{X \mapsto \text{adam}\} \mid \\
W1 = 0.8 * \min\{W3\}, \\
W2 \geq 0.6, W3 = 0.8 \qquad \qquad \qquad \Vdash_{\text{human.3}, \epsilon} \\
\text{human}(\text{adam}) \# W4 \mid \{X \mapsto \text{adam}\} \mid \\
W1 = 0.8 * \min\{W3\}, \\
W2 = 0.9 * \min\{W4\}, \\
W3 = 0.8, 0.90 * W4 \geq 0.6 \qquad \qquad \qquad \Vdash_{\text{human.1}, \epsilon} \\
\mid \{X \mapsto \text{adam}\} \mid \\
W1 = 0.8 * \min\{W3\}, \\
W2 = 0.9 * \min\{W4\}, \\
W3 = 0.8, W4 = 1.0
\end{array}$$

Note that the computed answer $\{X \mapsto \text{adam}\} \mid \{W1 \mapsto 0.64, W2 \mapsto 0.90\}$ subsumes the solution for the same goal given in Example 3.

2. Similarly, $SLD(\mathcal{W})$ resolution can solve the goal $\text{eats}(X, Y) \# W \mid W \leq 5.0$ for $\mathcal{P}_{\mathcal{W}}$, obtaining a computed answer $\{X \mapsto \text{father}(\text{adam})\} \mid \{W \mapsto 3.0\}$ which subsumes the solution for the same goal given in Example 3. \square

5 Towards an Implementation

In this section we assume a qualification domain \mathcal{D} and a constraint domain $\mathcal{C}_{\mathcal{D}}$ such that the qualification constraints used in $SLD(\mathcal{D})$ resolution can be expressed as $\mathcal{C}_{\mathcal{D}}$ constraints, and we describe a translation of $QLP(\mathcal{D})$ programs \mathcal{P} and goals G into $CLP(\mathcal{C}_{\mathcal{D}})$ programs \mathcal{P}^t and goals G^t , such that solving G with $SLD(\mathcal{D})$ 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 $SLD(\mathcal{D})$ resolution for the $QLP(\mathcal{D})$ language on top of any CLP or $CFLP$ system that supports $\mathcal{C}_{\mathcal{D}}$ constraints. In particular, if \mathcal{D} is any of the two qualification domains \mathcal{U} or \mathcal{W} , the constraint domain $\mathcal{C}_{\mathcal{D}}$ can be chosen as \mathcal{R} , which supports arithmetic constraints over the real numbers [10]. We have developed prototype implementations for $QLP(\mathcal{U})$, $QLP(\mathcal{W})$ and $QLP(\mathcal{U} \times \mathcal{W})$ on top of the $CFLP$ system *TOY* [3], that supports \mathcal{R} constraints. Note that although the use of a $CLP(\mathcal{R})$ system could lead to a more efficient implementation, we have chosen a $CFLP(\mathcal{R})$ system instead of a $CLP(\mathcal{R})$ one due to our interest in a future extension of the $QLP(\mathcal{D})$ scheme to support qualified $CFLP$ programming.

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

$QLP(\mathcal{D})$ clause

$$C \equiv p(\bar{t}) \leftarrow d - q_1(\bar{s}_1), \dots, q_k(\bar{s}_k)$$

its head is translated as $p(\bar{t}, Alpha, W, Beta)$, where the new variables $Alpha$, W and $Beta$ correspond, respectively, to α , W and β in the threshold constraint $\alpha \circ W \sqsupseteq \beta$ related to a \mathcal{D} -annotated atom $A \# W$ which could be selected for a $SLD(\mathcal{D})$ 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:

$$\begin{aligned} C^t \equiv & p(\bar{t}, Alpha, W, Beta) \leftarrow d \circ Alpha \sqsupseteq Beta, \\ & W_1 \sqsupseteq \perp, W_1 \sqsubseteq \top, q_1(\bar{s}_1, d \circ Alpha, W_1, Beta), \\ & \vdots \\ & W_k \sqsupseteq \perp, W_k \sqsubseteq \top, q_k(\bar{s}_k, d \circ Alpha, W_k, Beta), \\ & W = d \circ \bigsqcap \{W_1, \dots, W_k\} \end{aligned}$$

The conditions in the body of C^t do indeed correspond to the performance of a $SLD(\mathcal{D})$ resolution step with clause C . In fact, $d \circ Alpha \sqsupseteq Beta$ checks that C is eligible for such a step; the conditions in the next k lines using new variables W_i correspond to placing the annotated atoms from C 's body into the new goal; and the last condition introduces the proper defining constraint for W .

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

$$q_1(\bar{t}_1) \# W_1, \dots, q_m(\bar{t}_m) \# W_m \sqcap W_1 \sqsupseteq \beta_1, \dots, W_m \sqsupseteq \beta_m$$

where $\beta_1, \dots, \beta_m \in D \setminus \{\perp\}$, the translated goal G^t is

$$q_1(\bar{t}_1, \top, W_1, \beta_1), \dots, q_m(\bar{t}_m, \top, W_m, \beta_m)$$

where the three additional arguments at each atom are used to encode the initial threshold constraints $W_i \sqsupseteq \beta_i$, that are equivalent to $\top \circ W_i \sqsupseteq \beta_i$.

Example 5. As an example of the translation process we present the translation of the program $\mathcal{P}_{\mathcal{U}}$ from Example 1 into a \mathcal{TOY} program which uses \mathcal{R} constraints.

```

min1 [] = 1
min1 [X|Xs] = min2 X (min1 Xs)
min2 W1 W2 = if W1 <= W2 then W1 else W2
data being = adam | eve | bird | cat | oak | apple
           | father being | mother being
cruel(X,F,W,M) :- F*0.9>=M, W1>0, W1<=1.0, human(X,F*0.9,W1,M),
                  W2>0, W2<=1.0, eats(X,Y,F*0.9,W2,M),
                  W3>0, W3<=1.0, animal(Y,F*0.9,W3,M),
                  W == 0.9 * min1 [W1,W2,W3]
cruel(X,F,W,M) :- F*0.4>=M, W1>0, W1<=1.0, human(X,F*0.4,W1,M),
                  W2>0, W2<=1.0, eats(X,Y,F*0.4,W2,M),
                  W3>0, W3<=1.0, plant(Y,F*0.4,W3,M),
                  W == 0.4 * min1 [W1,W2,W3]

```

```

animal(bird,F,W,M) :- F*1.0>=M, W == 1.0 * min1 []
animal(cat,F,W,M)  :- F*1.0>=M, W == 1.0 * min1 []
plant(oak,F,W,M)  :- F*1.0>=M, W == 1.0 * min1 []
plant(apple,F,W,M) :- F*1.0>=M, W == 1.0 * min1 []
human(adam,F,W,M) :- F*1.0>=M, W == 1.0 * min1 []
human(eve,F,W,M)  :- F*1.0>=M, W == 1.0 * min1 []
human(father(X),F,W,M) :- F*0.9>=M, W1>0, W1<=1.0,
    human(X,F*0.9, W1, M), W == 0.9 * min1 [W1]
human(mother(X),F,W,M) :- F*0.8>=M, W1>0, W1<=1.0,
    human(X,F*0.8, W1, M), W == 0.8 * min1 [W1]
eats(adam,X,F,W,M) :- F*0.8>=M, W == 0.8 * min1 []
eats(eve,X,F,W,M) :- F*0.3>=M, W1>0, W1<=1.0,
    animal(X,F*0.3,W1,M), W == 0.3 * min1 [W1]
eats(eve,X,F,W,M) :- F*0.6>=M, W1>0, W1<=1.0,
    plant(X,F*0.6,W1,M), W == 0.6 * min1 [W1]
eats(father(X),Y,F,W,M) :- F*0.8>=M, W1>0, W1<=1.0,
    eats(X,Y,F*0.8,W1,M), W == 0.8 * min1 [W1]
eats(mother(X),Y,F,W,M) :- F*0.7>=M, W1>0, W1<=1.0,
    eats(X,Y,F*0.7,W1,M), W == 0.7 * min1 [W1]

```

To understand this example it is important to notice the following:

1. Since *glbs* in \mathcal{U} are computed as minimums, translated programs must include functions for this task. Here, `min1` resp. `min2` compute the minimum of a list of numbers resp. two numbers.
2. As \mathcal{TCY} need types for every constructor, we must include suitable datatype declarations in translated programs.
3. The resulting code could be simplified and optimized, but our aim here is to illustrate the literal application of the general translation rules. For this reason, no optimizations have been performed. \square

6 Conclusions and Future Work

We have generalized the early *QLP* proposal by van Emden [24] to a generic scheme $QLP(\mathcal{D})$ parameterized by a qualification domain \mathcal{D} , which must be a lattice with extreme points and equipped with an attenuation operator. The values belonging to a qualification domain are intended to qualify logical assertions, ensuring that they satisfy certain user's expectations. Qualification domains include \mathcal{B} (classical truth values of two-valued logic), \mathcal{U} (van Emden's certainty values) and \mathcal{W} (numeric values representing proof weights), as well as arbitrary cartesian products of given qualification domains. As shown by instances such as $QLP(\mathcal{W})$ and $QLP(\mathcal{U} \times \mathcal{W})$, the $QLP(\mathcal{D})$ scheme can express uncertainty in Logic Programming and more, since the user's expectations qualified by \mathcal{W} do not correspond to uncertain truth values.

The semantic results obtained for $QLP(\mathcal{D})$ are stronger than those in [24]. Each program \mathcal{P} has a least open Herbrand model $\mathcal{M}_{\mathcal{P}}$ with two equivalent

characterizations: as the least fixpoint of the operator $T_{\mathcal{P}}$, and as the set of qualified atoms deducible from \mathcal{P} in the logic calculus $QHL(\mathcal{D})$. Moreover, the goal solving calculus $SLD(\mathcal{D})$, based on an extension of SLD resolution with qualification constraints, is sound and strongly complete for arbitrary open goals. $SLD(\mathcal{B})$ boils down to classical SLD resolution.

As implementation technique, we have proposed a translation of $QLP(\mathcal{D})$ 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} . If \mathcal{D} is \mathcal{U} , \mathcal{B} , or $\mathcal{U} \times \mathcal{B}$, the constraint domain $\mathcal{C}_{\mathcal{D}}$ can be chosen as \mathcal{R} , and $QLP(\mathcal{D})$ can be implemented on top of any CLP or $CFLP$ system which supports constraint solving over \mathcal{R} . We have implemented prototypes of $QLP(\mathcal{U})$, $QLP(\mathcal{W})$ and $QLP(\mathcal{U} \times \mathcal{W})$ on top of the $CFLP$ system \mathcal{TOY} .

In comparison to the theory of generalized annotated logic programs (GAP for short) presented in [11], our results in this report also include some interesting contributions. With respect to the syntax and goal solving procedure, the $QLP(\mathcal{D})$ scheme can be made to fit into the GAP framework by viewing our attenuation operators as annotation functions. However, our resolution procedure $SLD(\mathcal{D})$ can be implemented more efficiently than the constrained SLD resolution used in GAP , due to an optimized treatment of qualification constraints and, more importantly, because the costly computation of so-called *reductants* between variants of program clauses is needed in GAP resolution but not in $SLD(\mathcal{D})$. The purpose of reductants in GAP is to explicitly compute the *lubs* of several lattice values (qualification values in the case of $QLP(\mathcal{D})$) which would result from finitely many different computations if no reductants were used. In GAP 's declarative semantics, interpretations are required to be closed w.r.t. finite *lubs* of lattice values assigned to the same atom, and for this reason reductants are needed for the completeness of goal resolution. In $QLP(\mathcal{D})$ interpretations as defined in Section 3 no closure condition w.r.t. *lubs* is required, and therefore the completeness result stated in Theorem 2 can be proved without reductants. Of course, the $QLP(\mathcal{D})$ approach to semantics means that a user has to observe *several* computed answers for one and the same goal and think of the *lub* of the various \mathcal{D} elements provided by the different computations by himself instead of getting the *lub* computed by one single $SLD(\mathcal{D})$ derivation. In our opinion, this is a reasonable scenario because even in GAP the \mathcal{T} value provided by any single computed answer always corresponds to some *lub* of finitely many \mathcal{T} values, and it may be not the highest possible \mathcal{T} value w.r.t. to the program's declarative semantics. Moreover, our Theorem 2 is much stronger than the one given in [11], which only ensures the possibility of computing *some* solution for any goal whose solvability holds in the least program model. We strongly conjecture that a stronger completeness theorem could be proved also for GAP by using a proof technique more similar to our's.

As possible lines of future work we consider: to improve the current prototype implementations of the instances $QLP(\mathcal{U})$, $QLP(\mathcal{W})$ and $QLP(\mathcal{U} \times \mathcal{W})$; to extend the $QLP(\mathcal{D})$ scheme and its implementation to a more expressive scheme which can support qualitative programming with features such as disjunctive

goals, negation, lazy functions and parametrically given constraint domains; to explore alternative semantic approaches, considering annotations, bilattices, probabilistic semantics and similarity based unification; and to investigate applications to the computation of qualified answers for web search queries.

Acknowledgements

The authors are thankful to their colleagues Paco López and Rafa Caballero for their valuable hints concerning bibliography and implementation techniques. They also appreciate the constructive comments of the anonymous reviewers, that were helpful for improving the presentation.

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. F. Arcelli and F. Formato. Likelog: a logic programming language for flexible data retrieval. In *Proceedings of the 1999 ACM Symposium on Applied computing (SAC'99)*, pages 260–267, New York, NY, USA, 1999. ACM Press.
3. P. Arenas, A. J. Fernández, A. Gil, F. J. López-Fraguas, M. Rodríguez-Artalejo, and F. Sáenz-Pérez. *TOY*, a multiparadigm declarative language. version 2.3.1, 2007. R. Caballero and J. Sánchez (Eds.), Available at <http://toy.sourceforge.net>.
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. A. Dekhtyar and V. S. Subrahmanian. Hybrid probabilistic programs. *Journal of Logic Programming*, 43(3):187–250, 2000.
6. J. Dix, S. Kraus, and V. S. Subrahmanian. Heterogeneous temporal probabilistic agents. *ACM Transactions on Computational Logic*, 7(1):151–198, 2006.
7. J. Dix, M. Nanni, and V. S. Subrahmanian. Probabilistic agent programs. *ACM Transactions on Computational Logic*, 1(2):208–246, 2000.
8. M. Fitting. Bilattices and the semantics of logic programming. *Journal of Logic Programming*, 11:91–116, 1991.
9. 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.
10. J. Jaffar, S. Michaylov, P. J. Stuckey, and R. H. C. Yap. The CLP(R) language and system. *ACM Transactions on Programming Languages and Systems*, 14(3):339–395, 1992.
11. 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.
12. 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.
13. 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.

14. R. T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1992.
15. R. T. Ng and V. S. Subrahmanian. A semantical framework for supporting subjective and conditional probability in deductive databases. *Journal of Automated Reasoning*, 10(2):191–235, 1993.
16. S. Riezler. *Probabilistic Constraint Logic Programming*. PhD thesis, Neuphilologischen Fakultät der Universität Tübingen, 1998.
17. C. A. Romero-Díaz. Programación lógica cuantitativa y su implementación en *TOY*. Proyecto de Fin de Máster, DSIC, Universidad Complutense de Madrid.
18. M. I. Sessa. Approximate reasoning by similarity-based SLD resolution. *Theoretical Computer Science*, 275(1-2):389–426, 2002.
19. E. Y. Shapiro. Logic programs with uncertainties: A tool for implementing rule-based systems. In A. Bundy, editor, *Proceedings of the 8th International Joint Conference on Artificial Intelligence (IJCAI'83)*, pages 529–532, Karlsruhe, Germany, 1983.
20. 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.
21. 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.
22. 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.
23. V. S. Subrahmanian. Uncertainty in logic programming: Some recollections. *Association for Logic Programming Newsletter*, 20(2), 2007.
24. M. H. van Emden. Quantitative deduction and its fixpoint theory. *Journal of Logic Programming*, 3(1):37–53, 1986.
25. 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.

A Appendix: Additional Proofs

This appendix contains the proofs of the Soundness Theorem 1 and the Completeness Theorem 2. In order to prove them, we will previously prove an auxiliary lemma for each of the two cases. In the rest of the Appendix, we assume a given program \mathcal{P} over a qualification domain \mathcal{D} .

A.1 Proof of the Soundness Theorem

Lemma 1 (Soundness). *Assume two goals G_0 and G_1 and a pair of substitutions (θ, ρ) such that $G_0 \Vdash_{C_1, \sigma_1} G_1$ and $(\theta, \rho) \in QSol_{\mathcal{P}}(G_1)$ (the set of all solutions of G_1). Then we have that $(\theta, \rho) \in QSol_{\mathcal{P}}(G_0)$.*

Proof. Assume $G_0 \equiv \bar{L}, A \# W, \bar{R} \parallel \sigma_0 \parallel \alpha \circ W \sqsupseteq \beta, \Delta$; $C_1 \equiv (H \leftarrow d - B_1, \dots, B_k) \in_{\text{var}} \mathcal{P}$ a variant of a program clause without variables in common with G_0 ; and σ_1 the m.g.u. between A and H . Then

$$G_1 \equiv (\bar{L}, B_1 \# W_1, \dots, B_k \# W_k, \bar{R})\sigma_1 \parallel \sigma_0\sigma_1 \parallel \Delta_1$$

where $\Delta_1 \equiv W = d \circ \prod\{W_1, \dots, W_k\}, d \circ \alpha \circ W_1 \sqsupseteq \beta, \dots, d \circ \alpha \circ W_k \sqsupseteq \beta, \Delta$. As $(\theta, \rho) \in QSol_{\mathcal{P}}(G_1)$ we know

- (1) $\sigma_0\sigma_1\theta = \theta$,
- (2) $\rho \in Sol(\Delta_1)$, and
- (3) $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} (\bar{L}, B_1 \# W_1, \dots, B_k \# W_k, \bar{R})\sigma_1^{\wedge}(\theta, \rho)^1$.

And for (θ, ρ) to be a solution of G_0 we need the following:

- (4) $\sigma_0\theta = \theta$,
- (5) $\rho \in Sol(\alpha \circ W \sqsupseteq \beta, \Delta)$, and
- (6) $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} (\bar{L}, A \# W, \bar{R})^{\wedge}(\theta, \rho)$.

Therefore we have to prove (4), (5) and (6).

Proof of (4). First, we can see that for every variable $y \in \text{Var}$, if $y \in \text{vran}(\sigma_0)$, then $y \notin \text{dom}(\sigma_0)$ because σ_0 is idempotent. Hence, $y \in \text{vran}(\sigma_0) \implies y\sigma_1\theta = y\sigma_0\sigma_1\theta \stackrel{(1)}{=} y\theta$, and it is true that (7) $\sigma_1\theta = \theta [\text{vran}(\sigma_0)]$. Now, for any variable x we can prove $x\sigma_0\theta = x\theta$ by distinguishing two cases: a) if $x \notin \text{dom}(\sigma_0)$ then $x\sigma_0\theta = x\theta$; and b) if $x \in \text{dom}(\sigma_0)$ then $\text{var}(x\sigma_0) \subseteq \text{vran}(\sigma_0) \implies x\sigma_0\theta \stackrel{(7)}{=} x\sigma_0\sigma_1\theta \stackrel{(1)}{=} x\theta$.

Proof of (5). We have to prove that $\alpha \circ W\rho \sqsupseteq \beta$ and $\rho \in Sol(\Delta)$. $\alpha \circ W\rho \stackrel{(2)}{=} \alpha \circ d \circ \prod\{W_1\rho, \dots, W_k\rho\} = \prod\{\alpha \circ d \circ W_1\rho, \dots, \alpha \circ d \circ W_k\rho\}$. It is enough proving $\alpha \circ d \circ W_i\rho \sqsupseteq \beta$ for $1 \leq i \leq k$. But $\{\alpha \circ d \circ W_i \sqsupseteq \beta \mid 1 \leq i \leq k\} \subseteq \Delta_1$ and $\rho \in Sol(\Delta_1)$. $\rho \in Sol(\Delta)$ is trivial because $\Delta \subseteq \Delta_1$.

¹ $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} (A \# W)^{\wedge}(\theta, \rho) \iff_{\text{def}} \mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} A\theta \# W\rho$.

Proof of (6). We can split (6) in the following three cases:

- (6a) $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} \overline{L}^\wedge(\theta, \rho)$. We prove that $\overline{L}^\wedge(\theta, \rho) = \overline{L}\sigma_1^\wedge(\theta, \rho)$ which, because of (3), can be inferred in $\text{QHL}(\mathcal{D})$ from \mathcal{P} . We know that $\text{dom}(\sigma_0) \cap \text{var}(G_0) = \emptyset$, therefore, $\overline{L}\sigma_1^\wedge(\theta, \rho) = \overline{L}\sigma_0\sigma_1^\wedge(\theta, \rho) = \overline{L}^\wedge(\sigma_0\sigma_1\theta, \rho) \stackrel{(1)}{=} \overline{L}^\wedge(\theta, \rho)$.
- (6b) $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} A \# W^\wedge(\theta, \rho)$. Using $W\rho = d \circ \prod\{W_1\rho, \dots, W_k\rho\}$ which holds because of (2), (3) and one inference step with clause C_1 and substitution $\sigma_1\theta$, we obtain $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} H\sigma_1\theta \# W\rho$. Now, because σ_1 is the m.g.u. between A and H , we have $H\sigma_1\theta = A\sigma_1\theta$. Therefore, we have $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} A\sigma_1\theta \# W\rho$. Finally, we note that $A\sigma_1\theta = A\sigma_0\sigma_1\theta \stackrel{(1)}{=} A\theta$ because $\text{dom}(\sigma_0) \cap \text{var}(A) = \emptyset$.
- (6c) $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} \overline{R}^\wedge(\theta, \rho)$. As in (6a). \square

Proof (of Soundness Theorem). Assume $G_0 \Vdash_\sigma^n G$ where $G \equiv \sigma \parallel \Delta$ is solved. Let (σ, μ) be the solution associated to G . We prove $(\sigma, \mu) \in \text{QSol}_{\mathcal{P}}(G_0)$ by induction on n .

Base. In this case, $n = 0$ and $G_0 = G$ is solved and $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} \overline{A}^\wedge(\sigma, \mu)$ is trivial because the sequence of atoms \overline{A} of G is empty. Moreover, $\mu \in \text{Sol}(\Delta)$ because $\mu = \omega_\Delta$.

Induction. In this case we have $n > 0$ and $G_0 \Vdash G_1 \Vdash^{n-1} G$. Then we obtain $(\sigma, \mu) \in \text{QSol}_{\mathcal{P}}(G_1)$ by induction hypothesis, and therefore $(\sigma, \mu) \in \text{QSol}_{\mathcal{P}}(G_0)$ because of Lemma 1. \square

A.2 Proof of the Completeness Theorem

Before going into the proof, just a note on notation: as said, $(\theta, \rho) \in \text{QSol}_{\mathcal{P}}(G)$ means that the pair of substitutions (θ, ρ) is a solution of the goal $G \equiv \overline{A} \parallel \sigma \parallel \Delta$. Now, writing $(\theta, \rho) \in \text{QSol}_{\mathcal{P}}^n(G)$ we are expressing that the exact number of inference steps in $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})} \overline{A}^\wedge(\theta, \rho)$ is n , written as $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^n \overline{A}^\wedge(\theta, \rho)$.

Lemma 2 (Completeness). *Let $G_0 \equiv \overline{A}_0 \parallel \sigma_0 \parallel \Delta_0$ be a goal not solved, and $(\theta_0, \rho_0) \in \text{QSol}_{\mathcal{P}}^n(G_0)$. Let also V_0 be any finite set of variables such that $\text{var}(G_0) \cup \text{dom}(\theta_0) \subseteq V_0$. For any arbitrary selection of an atom $A \# W$ of \overline{A}_0 , there exists some resolution step $G_0 \Vdash_{\sigma_1} G_1$ selecting the chosen atom and, in addition, some (θ_1, ρ_1) satisfying the following properties:*

- a. $\theta_1 = \theta_0[V_0]$
- b. $\sigma_1\theta_1 = \theta_1$
- c. $\sigma_0\sigma_1\theta_1 = \theta_1$
- d. $\rho_1 \supseteq \rho_0[\text{var}(G_0)]$
- e. $\rho_1 \in \text{Sol}(\Delta_1)$
- f. $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^{n-1} \overline{A}_1^\wedge(\theta_1, \rho_1)$

In particular, (c), (e) and (f) mean that $(\theta_1, \rho_1) \in \text{QSol}_{\mathcal{P}}^{n-1}(G_1)$.

Proof. Assume $A \# W$ to be the selected atom in G_0 . Then, $G_0 \equiv \overline{L}_0, A \# W, \overline{R}_0 \parallel \sigma_0 \parallel \alpha \circ W \supseteq \beta, \Delta$. Because of the lemma's hypothesis we can also assume the following:

- (0) $\sigma_0\theta_0 = \theta_0$,
- (1) $\rho_0 \in \text{Sol}(\Delta_0)$,
- (2) $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^{m_1} \overline{L_0}^\wedge(\theta_0, \rho_0)$,
- (3) $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^{m_2} A \# W^\wedge(\theta_0, \rho_0)$, and
- (4) $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^{m_3} \overline{R_0}^\wedge(\theta_0, \rho_0)$

with $m_1 + m_2 + m_3 = n > 0$.

Because of (3) there must exist some clause $C_1 \equiv (H \leftarrow d - B_1, \dots, B_k) \in_{\text{var}} \mathcal{P}$ and some substitution η_0 such that

- (5) $A\theta_0 = H\eta_0$ and $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^{m_2-1} B_1\eta_0 \# d_1, \dots, B_k\eta_0 \# d_k$ with $d_1, \dots, d_k \in D \setminus \{\perp\}$ such that $W\rho_0 \sqsubseteq d \circ \prod\{d_1, \dots, d_k\}$.

It is possible to choose C_1 and η_0 so that $\text{var}(C_1) \cap V_0 = \emptyset$ and $\text{dom}(\eta_0) \subseteq \text{var}(C_1)$. Therefore, it is guaranteed that $\text{dom}(\eta_0) \cap \text{dom}(\theta_0) = \emptyset$ and then:

- (6) $\theta_1 \stackrel{\text{def}}{=} \theta_0 \uplus \eta_0$ is a well-founded substitution that satisfies: $\text{dom}(\theta_1) = \text{dom}(\theta_0) \uplus \text{dom}(\eta_0)$; $\theta_1 = \theta_0 [V_0]$; $\theta_1 = \eta_0 [\setminus V_0] \implies (a)$ of lemma.

From (5) and (6) we know that θ_1 is a unifier of A and H . Choosing σ_1 as the m.g.u. (in the Robinson's sense) between A and H we will have:

- (7) $A\sigma_1 = H\sigma_1$ and $\sigma_1\theta_1 = \theta_1 \implies (b)$ of lemma.

Then, taking ρ_1 such that

- (8) $W'\rho_1 \stackrel{\text{def}}{=} \begin{cases} d_i & \text{if } W' = W_i \text{ for some } 1 \leq i \leq k \\ d \circ \prod\{d_1, \dots, d_k\} & \text{if } W' = W \\ W'\rho_0 & \text{otherwise} \end{cases}$

we will have, by (8) and (5), $\rho_1 \sqsupseteq \rho_0 [\text{var}(G_0)] \implies (d)$ of lemma.

Now, doing a resolution step with σ_1 and C_1 we get $G_0 \Vdash_{\sigma_1, C_1} G_1 \equiv \overline{A_1} \parallel \sigma_0\sigma_1 \parallel \Delta_1$ where $\overline{A_1} = (\overline{L_0}, B_1 \# W_1, \dots, B_k \# W_k, \overline{R_0})\sigma_1$ and $\Delta_1 \equiv d \circ \alpha \circ W_1 \sqsupseteq \beta, \dots, d \circ \alpha \circ W_k \sqsupseteq \beta, W = d \circ \prod\{W_1, \dots, W_k\}, \Delta$. Note that we can deduce $d \circ \alpha \sqsupseteq \beta$ from (5), (1) and the axioms required for the attenuation operation (\circ) in any qualification domain, because we have $W\rho_0 = d \circ \prod\{d_1, \dots, d_k\}$ and $\alpha \circ W\rho_0 \sqsupseteq \beta \implies \alpha \circ d \circ \prod\{d_1, \dots, d_k\} \sqsupseteq \beta \implies \alpha \circ d \sqsupseteq \beta$. Remember from Definition 3 that the condition $\alpha \circ d \sqsupseteq \beta$ is required for the resolution step to be enabled.

To finish the proof we only need to prove (c), (e) and (f).

Proof of (c). $\sigma_0\sigma_1\theta_1 \stackrel{(b)}{=} \sigma_0\theta_1 \stackrel{(6)}{=} \sigma_0(\theta_0 \uplus \eta_0) \stackrel{(*)}{=} \sigma_0\theta_0 \uplus \eta_0 \stackrel{(0)}{=} \theta_0 \uplus \eta_0 = \theta_1$.
 (*) Because $\text{vran}(\sigma_0) \subseteq V_0$ and $\text{dom}(\eta_0) \cap V_0 = \emptyset$.

Proof of (e). We have to see that ρ_1 satisfies every constraint in Δ_1 :

1. $W = d \circ \prod\{W_1, \dots, W_k\}$. This is satisfied by definition of ρ_1 .

2. $d \circ \alpha \circ W_i \sqsupseteq \beta$ for $1 \leq i \leq k$. We know from (1) that $\alpha \circ W\rho_0 \sqsupseteq \beta$, and from (d) follows $W\rho_1 \sqsupseteq W\rho_0$. Therefore, $\alpha \circ W\rho_1 \sqsupseteq \beta$. Because of (8) we also know $W\rho_1 = d \circ \prod\{W_1\rho_1, \dots, W_k\rho_1\}$ that implies $\alpha \circ W\rho_1 = \prod\{d \circ \alpha \circ W_1\rho_1, \dots, d \circ \alpha \circ W_k\rho_1\}$. Hence $\alpha \circ W\rho_1 \sqsupseteq \beta$ implies that $d \circ \alpha \circ W_i\rho_1 \sqsupseteq \beta$ is satisfied for every $1 \leq i \leq k$.
3. Δ . From (7) follows that $\rho_1 = \rho_0 [\text{war}(\Delta)]$, and because of (1) $\rho_0 \in \text{Sol}(\Delta) \implies \rho_1 \in \text{Sol}(\Delta)$.

Proof of (f). First we can deduce:

- (9) $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^{m_1} \overline{L_0}^\wedge(\theta_1, \rho_1)$ by (2), (6) and (8).
- (10) $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^{m_2-1} (B_1 \# W_1, \dots, B_k \# W_k)^\wedge(\theta_1, \rho_1)$ by (5), (6) and (8).
- (11) $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^{m_3} \overline{R_0}^\wedge(\theta_1, \rho_1)$ by (4), (6) and (8).

Considering that $m_1 + (m_2 - 1) + m_3 = n - 1$, (9), (10) and (11) imply that $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^{n-1} (\overline{L_0}, B_1 \# W_1, \dots, B_k \# W_k, \overline{R_0})^\wedge(\theta_1, \rho_1)$; and this is (f) due to (b). \square

Proof (of Completeness Theorem). As G_0 is a goal, we know that σ_0 is idempotent and that $\overline{A_0}\sigma_0 = \overline{A_0}$. Now, as $(\theta_0, \rho_0) \in \text{QSOL}_{\mathcal{P}}(G_0)$, we can choose a number $n \in \mathbb{N}$ such that $(\theta_0, \rho_0) \in \text{QSOL}_{\mathcal{P}}^n(G_0)$ and therefore we have (1) $\sigma_0\theta_0 = \theta_0$, (2) $\rho_0 \in \text{Sol}(\Delta_0)$ and (3) $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^n \overline{A_0}^\wedge(\theta_0, \rho_0)$. We can also choose a finite set of variables V_0 satisfying (4) $\text{var}(G_0) \cup \text{dom}(\theta_0) \subseteq V_0$. Given the conditions (1) to (4), we will prove the following:

- (†) There exist some resolution computation $G_0 \Vdash_{\sigma}^* \sigma_0\sigma \sqsupseteq \Delta$ (that we can build making use of any selection strategy) ending in a solved goal, and some substitution θ satisfying (5) $\theta = \theta_0 [V_0]$, (6) $\sigma\theta = \theta$ and (7) $\sigma_0\sigma\theta = \theta$.

From (†) follows the theorem's thesis (except (8) $\mu \sqsupseteq \rho_0 [\text{war}(G_0)]$) because:

- (6) $\implies_{(5)} \sigma\theta = \theta_0 [V_0] \implies \sigma \preceq \theta_0 [\text{var}(G_0)]$
- (7) $\implies_{(5)} \sigma_0\sigma\theta = \theta_0 [V_0] \implies \sigma_0\sigma \preceq \theta_0 [\text{var}(G_0)]$

We simultaneously prove (†) and (8) by induction on n :

Base. If $n = 0$, (2) implies that $\overline{A_0}$ is empty. Then, taking $\sigma = \epsilon$ and $\theta = \theta_0$ we have that $G_0 \Vdash_{\epsilon}^0 \sigma_0 \sqsupseteq \Delta_0$ with is a trivial resolution of 0 steps; and in addition:

- (5) reduces to $\theta_0 = \theta_0 [V_0]$, which is trivial.
- (6) reduces to $\theta_0 = \theta_0$, which also is trivial.
- (7) reduces to $\sigma_0\theta_0 = \theta_0$ which is true given (1).
- (8) is satisfied because $\mu = \rho_0 [\text{war}(G_0)]$ is true, given that $\text{war}(G_0) = \text{war}(\Delta_0)$. Now, as Δ_0 is solved, it only contains defining constraints and therefore (2) implies that for any $W \in \text{war}(\Delta_0)$ it is true that $W\rho_0 = \omega_{\Delta_0}(W) = W\mu$.

Induction. If $n > 0$, (2) implies that $\overline{A_0}$ is not empty. Hence, selecting an atom $A \# W$ in $\overline{A_0}$ with any selection strategy and using the Completeness Lemma 2, we can perform a resolution step

$$(9) \quad G_0 \equiv \overline{A_0} \parallel \sigma_0 \parallel \Delta_0 \Vdash_{\sigma_1} \overline{A_1} \parallel \sigma_0 \sigma_1 \parallel \Delta_1 \equiv G_1$$

having that there exists some solution $(\theta_1, \rho_1) \in QSol_{\mathcal{P}}^{n-1}(G_1)$ satisfying all 6 conditions guaranteed by the lemma: (10) $\theta_1 = \theta_0 [V_0]$, (11) $\sigma_1 \theta_1 = \theta_1$, (1') $\sigma_0 \sigma_1 \theta_1 = \theta_1$, (12) $\rho_1 \sqsupseteq \rho_0 [\text{war}(G_0)]$, (2') $\rho_1 \in Sol(\Delta_1)$, and (3') $\mathcal{P} \vdash_{\text{QHL}(\mathcal{D})}^{n-1} \overline{A_1}^{\wedge}(\theta_1, \rho_1)$.

Let V_1 be any finite set of variables such that

$$(4') \quad V_0 \cup \text{var}(G_1) \cup \text{dom}(\theta_1) \subseteq V_1 \quad .$$

Conditions (1'), (2'), (3') and (4') are similar, respectively, to (1), (2), (3) and (4), but now for $(\theta_1, \rho_1) \in QSol_{\mathcal{P}}^{n-1}(G_1)$. By induction hypothesis we can obtain a resolution computation

$$(13) \quad G_1 \Vdash_{\sigma'}^* \sigma_0 \sigma_1 \sigma' \parallel \Delta'$$

and a substitution θ such that

$$(5') \quad \theta = \theta_1 [V_1] \quad (6') \quad \sigma' \theta = \theta \quad (7') \quad \sigma_0 \sigma_1 \sigma' \theta = \theta$$

$$(8') \quad \mu' \sqsupseteq \rho_1 [\text{war}(G_1)] \text{ with } (\sigma_0 \sigma_1 \sigma', \mu') \text{ the associated solution to } \sigma_0 \sigma_1 \sigma' \parallel \Delta'.$$

From (9) and (13) results

$$G_0 \equiv \overline{A_0} \parallel \sigma_0 \parallel \Delta_0 \Vdash_{\sigma_1} G_1 \equiv \overline{A_1} \parallel \sigma_0 \sigma_1 \parallel \Delta_1 \Vdash_{\sigma'}^* \sigma_0 \sigma_1 \sigma' \parallel \Delta' \quad .$$

Now there is only left to check that (5), (6) and (7) are satisfied given the same θ that satisfies (5'), (6') and (7') and $\sigma = \sigma_1 \sigma'$; and that (8) is also satisfied when $\mu = \mu'$. In fact:

- (5) trivially follows from (5'), (4') and (10).
- (6) comes from the following: by (4') we can assume $\theta = \theta_1 \uplus \eta'$, with η' such that $\text{dom}(\eta') \cap (V_1) = \emptyset$. Then: $\underline{\sigma} \theta = \sigma_1 \underline{\sigma} \theta \stackrel{(6')}{=} \sigma_1 \underline{\theta} = \sigma_1 (\theta_1 \uplus \eta') \stackrel{(*)}{=} \underline{\sigma_1 \theta_1} \uplus \eta' \stackrel{(11)}{=} \theta_1 \uplus \eta' = \theta$. The step (*) is correct because $\text{vran}(\sigma_1) \subseteq V_1$ and $\text{dom}(\eta') \cap V_1 = \emptyset$.
- (7) trivially follows from (7'), given that $\sigma = \sigma_1 \sigma'$.
- (8) is consequence of (8') and (12), because $\text{war}(G_0) \subseteq \text{war}(G_1)$. □