TLP **10** (4–6): 627–642, 2010. © Cambridge University Press 2010
doi:10.1017/S1471068410000323

*A declarative semantics for CLP with qualification and proximity**

MARIO RODRÍGUEZ-ARTALEJO and CARLOS A. ROMERO-DÍAZ

Departamento de Sistemas Informáticos y Computación, Universidad Complutense
Facultad de Informática, 28040 Madrid, Spain
(*e-mail:* mario@sip.ucm.es, cromdia@fdi.ucm.es)

submitted 4 February 2010; revised 30 April 2010; accepted 21 May 2010

Abstract

Uncertainty in Logic Programming has been investigated during the last decades, dealing with various extensions of the classical LP paradigm and different applications. Existing proposals rely on different approaches, such as clause annotations based on uncertain truth values, qualification values as a generalization of uncertain truth values, and unification based on proximity relations. On the other hand, the CLP scheme has established itself as a powerful extension of LP that supports efficient computation over specialized domains while keeping a clean declarative semantics. In this paper we propose a new scheme SQCLP designed as an extension of CLP that supports qualification values and proximity relations. We show that several previous proposals can be viewed as particular cases of the new scheme, obtained by partial instantiation. We present a declarative semantics for SQCLP that is based on observables, providing fixpoint and proof-theoretical characterizations of least program models as well as an implementation-independent notion of goal solutions.

KEYWORDS: constraint logic programming, qualification domains and values, proximity relations

1 Introduction

Many extensions of logic programming (shortly LP) to deal with uncertainty have been proposed in the last decades. A line of research not related to this paper is based on probabilistic extensions of LP such as Ng and Subrahmanian (1992). Other proposals in the field replace classical two-valued logic by some kind of many-valued logic whose truth values can be attached to computed answers and are usually interpreted as certainty degrees. The next paragraphs summarize some relevant approaches of this kind.

There are extensions of LP using annotations in program clauses to compute a certainty degree for the head atom from the certainty degrees previously computed for the body atoms. This line of research includes the seminal proposal of Quantitative Logic Programming by van Emden (1986) and inspired later works such as the

* This work has been partially supported by the Spanish projects STAMP (TIN2008-06622-C03-01), PROMETIDOS–CM (S2009TIC-1465) and GPD–UCM (UCM–BSCH–GR58/08-910502).

Generalized Annotated logic Programs (shortly GAP) by Kifer and Subrahmanian (1992) and the QLP scheme for Qualified LP Rodríguez-Artalejo and Romero-Díaz (2008). While van Emden (1986) and other early approaches used real numbers of the interval $[0, 1]$ as certainty degrees, QLP and GAP take elements from a parametrically given lattice to be used in annotations and attached to computed answers. In the case of QLP, the lattice is called a *qualification domain* and its elements (called *qualification values*) are not always understood as certainty degrees. As argued in Rodríguez-Artalejo and Romero-Díaz (2008), GAP is a more general framework, but QLP's semantics have some advantages for its intended scope.

There are also extended LP languages based on fuzzy logic which can be classified into two major lines. The first line includes Fuzzy LP languages such as Vojtáš (2001) and Guadarrama *et al.* (2004) and the Multi-Adjoint LP (shortly MALP) framework by Medina *et al.* (2001). All these approaches extend classical LP by using clause annotations and a fuzzy interpretation of the connectives and aggregation operators occurring in program clauses and goals. There is a relationship between Fuzzy LP and GAP that has been investigated in Krajči *et al.* (2004). Intended applications of Fuzzy LP languages include expert knowledge representation.

The second line includes Similarity-based LP (shortly SLP) in the sense of Sessa (2002) and related proposals, which keep the classical syntax of LP clauses but use a *similarity relation* over a set of symbols S to allow "flexible" unification of syntactically different symbols with a certain approximation degree. Similarity relations over a given set S have been defined in Zadeh (1971) and Sessa (2002) and related literature as fuzzy relations represented by mappings $\mathscr{S} : S \times S \to [0, 1]$ which satisfy reflexivity, symmetry and transitivity axioms analogous to those required for classical equivalence relations. A more general notion called *proximity relation* was introduced in Dubois and Prade (1980) by omitting the transitivity axiom. As noted by Shenoi and Melton (1999) and other authors, the transitivity property required for similarity relations may conflict with user's intentions in some cases. The Bousi∼Prolog language (Julián-Iranzo and Rubio-Manzano 2009) has been designed with the aim of generalizing SLP to work with proximity relations. A different generalization of SLP is the SQLP scheme (Caballero *et al.* 2008), designed as an extension of the QLP scheme. In addition to clause annotations in QLP style, SQLP uses a given similarity relation $\mathscr{S} : S \times S \to D$ (where D is the carrier set of a parametrically given qualification domain) in order to support flexible unification. In the sequel we use the acronym SLP as including proximity-based LP languages also. Intended applications of SLP include flexible query answering. An analogy of proximity relations in a different context (namely partial constraint satisfaction) can be found in Freuder and Wallace (1992), where several metrics are proposed to measure the proximity between the solution sets of two different constraint satisfaction problems.

Several of the above mentioned LP extensions (including GAP, QLP, the Fuzzy LP language in Guadarrama *et al.* (2004) and SQLP) have used constraint solving as an implementation technique. However, we only know two approaches which have been conceived as extensions of the classical CLP scheme Jaffar and Lassez (1987). Firstly, Riezler (1998) extended the formulation of CLP by Höhfeld and Smolka (1988) with quantitative LP in the sense of van Emden (1986); this work was motivated by problems from the field of natural language processing. Secondly,

https://doi.org/10.1017/S1471068410000323 Published online by Cambridge University Press

Bistarelli *et al.* (2001) proposed a semiring-based approach to CLP, where constraints are solved in a soft way with levels of consistency represented by values of a semiring. This approach was motivated by constraint satisfaction problems and implemented with `clp(FD,S)` in Georget and Codognet (1998) for a particular class of semirings which enable to use local consistency algorithms. The relationships between Riezler (1998) and Bistarelli *et al.* (2001) and the results of this paper will be further discussed in Section 4.

Finally, there are a few preliminary attempts to combine some of the above mentioned approaches with the Functional Logic Programming (shortly FLP) paradigm found in languages such as Curry (Hanus 2006) and $\mathcal{TOY}$ (Arenas *et al.* 2007). Similarity-based unification for FLP languages has been investigated by (Moreno and Pascual 2007), while (Caballero *et al.* 2009) have proposed a generic scheme QCFLP designed as a common extension of the two schemes CLP and QLP with first-order FLP features.

In this paper we propose a new extension of CLP that supports qualification values and proximity relations. More precisely, we define a generic scheme SQCLP whose instances SQCLP($\mathscr{S},\mathscr{D},\mathscr{C}$) are parameterized by a proximity relation $\mathscr{S}$, a qualification domain $\mathscr{D}$ and a constraint domain $\mathscr{C}$. We will show that several previous proposals can be viewed as particular cases of SQCLP, obtained by partial instantiation. Moreover, we will present a declarative semantics for SQCLP that is inspired in the observable CLP semantics by Gabbrielli *et al.* (1995) and provides fixpoint and proof-theoretical characterizations of least program models as well as an implementation-independent notion of goal solution that can be used to specify the expected behavior of goal solving systems.

The reader is assumed to be familiar with the semantic foundations of LP and CLP. The rest of the paper is structured as follows: Section 2 introduces constraint domains, qualification domains and proximity relations. Section 3 presents the SQCLP scheme and the main results on its declarative semantics. Finally, Section 4 concludes by giving a discussion of related approaches (many of which can be viewed as particular cases of SQCLP) and pointing to some lines open for future work. Due to space limits, we have preferred to include examples rather than proofs. A widely extended version including detailed proofs is available as Technical Report (Rodríguez-Artalejo and Romero-Díaz 2010).

2 Computational basis

2.1 Constraint domains

As in the CLP Scheme, we will work with constraint domains related to signatures. We assume a *universal programming signature* $\Gamma = \langle DC, DP\rangle$ where $DC = \bigcup_{n\in\mathbb{N}} DC^n$ and $DP = \bigcup_{n\in\mathbb{N}} DP^n$ are countably infinite and mutually disjoint sets of free function symbols (called *data constructors* in the sequel) and *defined predicate* symbols, respectively, ranked by arities. We will use *domain specific signatures* $\Sigma = \langle DC, DP, PP\rangle$ extending Γ with a disjoint set $PP = \bigcup_{n\in\mathbb{N}} PP^n$ of *primitive predicate* symbols, also ranked by arities. The idea is that primitive predicates come along with constraint domains, while defined predicates are specified in user programs. Each PP^n may be any countable set of n-ary predicate symbols.

https://doi.org/10.1017/S1471068410000323 Published online by Cambridge University Press

Terms have the syntax $t ::= X|u|c(\bar{t}_n)$, where $X \in \mathcal{V}ar$, $u \in B$ and $c \in DC^n$, assuming a countably infinite set of variables $\mathcal{V}ar$ and a set of basic values B and using $\bar{t}_n$ as a shorthand for $t_1, \ldots, t_n$. The set of ground terms is noted $\mathrm{Term}(\Sigma, B)$. As usual, *substitutions* are defined as mappings σ assigning terms to variables and extended to act over other syntactic objects o in the natural way. The result of applying σ to o is noted as $o\sigma$.

Several formal notions of *constraint domain* are known in the literature. In this paper, constraint domains of signature Σ are relational structures of the form $\mathscr{C} = \langle C, \{p^{\mathscr{C}} \mid p \in PP\}\rangle$ consisting of a carrier set $C = \mathrm{Term}(\Sigma, B)$ and an interpretation $p^{\mathscr{C}} : C^n \to \{0,1\}$ for each $p \in PP^n$. For the examples in this paper we will use the *real constraint domain* $\mathscr{R}$ well known as the basis of the CLP($\mathscr{R}$) language and system (Jaffar *et al.* 1992). In our setting we represent $\mathscr{R}$ with set of basic values $B = \mathbb{R}$ and primitive predicates $op_+, op_\times, \ldots \in PP^3$ and $cp_>, cp_\geqslant, \ldots \in PP^2$ defined to represent the usual arithmetic and comparison operations over $\mathbb{R}$. Other useful constraint domains are: the *Herbrand* domain $\mathscr{H}$, intended to work just with equality constraints; and $\mathscr{FD}$, intended to work with constraints involving integer values and finite domain variables.

Given a constraint domain $\mathscr{C}$, we will work with *atoms* of three kinds: defined atoms $A : r(\bar{t}_n)$, where $r \in DP^n$ and t_i are terms; primitive atoms $\kappa : p(\bar{t}_n)$, where $p \in PP^n$ and t_i are terms; and equations $t == s$, where t, s are terms and $==$ is the *equality symbol*. Primitive atoms and equations are called *atomic $\mathscr{C}$-constraints*. More generally, $\mathscr{C}$-constraints π are built from atomic $\mathscr{C}$-constraints using logical conjunction $\wedge$, existential quantification $\exists$, and sometimes other logical operations. Constraints of the form $\exists X_1 \ldots \exists X_n (B_1 \wedge \ldots \wedge B_m)$ –where B_j $(1 \leqslant j \leqslant m)$ are atomic– are called *existential*.

The set of all $\mathscr{C}$-constraints is noted $\mathrm{Con}_{\mathscr{C}}$. Constraints are interpreted by means of *$\mathscr{C}$-valuations* $\eta \in \mathrm{Val}_{\mathscr{C}}$, which are ground substitutions. The set $\mathrm{Sol}_{\mathscr{C}}(\Pi)$ of solutions of $\Pi \subseteq \mathrm{Con}_{\mathscr{C}}$ includes all the valuations η such that $\Pi\eta$ is true when interpreted in $\mathscr{C}$. $\Pi \subseteq \mathrm{Con}_{\mathscr{C}}$ is called *satisfiable* if $\mathrm{Sol}_{\mathscr{C}}(\Pi) \neq \emptyset$ and *unsatisfiable* otherwise. $\pi \in \mathrm{Con}_{\mathscr{C}}$ *is entailed* by $\Pi \subseteq \mathrm{Con}_{\mathscr{C}}$ (noted $\Pi \models_{\mathscr{C}} \pi$) iff $\mathrm{Sol}_{\mathscr{C}}(\Pi) \subseteq \mathrm{Sol}_{\mathscr{C}}(\pi)$.

As a simple illustration consider $\Pi = \{cp_\geqslant(A, 3.0), op_+(A, A, X), op_\times(2.0, A, Y)\} \subseteq \mathrm{Con}_{\mathscr{R}}$. Clearly, $\eta \in \mathrm{Sol}_{\mathscr{R}}(\Pi)$ holds iff $\eta(A)$, $\eta(X)$ and $\eta(Y)$ are real numbers $a, x, y \in \mathbb{R}$ such that $a \geqslant 3.0$, $a + a = x$ and $2.0 \times a = y$. Then $\mathrm{Sol}_{\mathscr{R}}(\Pi) \subseteq \mathrm{Sol}_{\mathscr{R}}(X == Y)$. Therefore, assuming $c \in DC^1$ one has $\Pi \models_{\mathscr{R}} c(X) == c(Y)$.

2.2 Qualification domains

As mentioned in the Introduction, qualification values were introduced as a generalization of certainty values in Rodríguez-Artalejo and Romero-Díaz (2008). They are elements of special lattices called *qualification domains* and defined as structures $\mathscr{D} = \langle D, \preccurlyeq, \mathbf{b}, \mathbf{t}, \circ\rangle$ verifying the following requirements:

1. $\langle D, \preccurlyeq, \mathbf{b}, \mathbf{t}\rangle$ is a lattice with extreme points $\mathbf{b}$ (called *infimum* or *bottom* element) and $\mathbf{t}$ (called *maximum* or *top* element) w.r.t. the partial ordering $\preccurlyeq$ (called *qualification ordering*). For given elements $d, e \in D$, we write $d \sqcap e$ for the *greatest*

https://doi.org/10.1017/S1471068410000323 Published online by Cambridge University Press

lower bound (*glb*) of d and e, and $d \sqcup e$ for the *least upper bound* (*lub*) of d and e. We also write $d \lhd e$ as abbreviation for $d \trianglelefteq e \wedge d \neq e$.

2. $\circ : D \times D \to D$, called *attenuation operation*, verifies the following axioms:

(a) $\circ$ is associative, commutative and monotonic w.r.t. $\trianglelefteq$.
(b) $\forall d \in D : d \circ \mathbf{t} = d$ and $d \circ \mathbf{b} = \mathbf{b}$.
(c) $\forall d, e \in D : d \circ e \trianglelefteq e$ and even $\mathbf{b} \neq d \circ e \trianglelefteq e$ if $d, e \in D \setminus \{\mathbf{b}\}$.
(d) $\forall d, e_1, e_2 \in D : d \circ (e_1 \sqcap e_2) = (d \circ e_1) \sqcap (d \circ e_2)$.

Actually, axioms (2)(b.2) and (2)(c.1) are redundant because they can be derived from the other axioms[1]. For any $S = \{e_1, e_2, \ldots, e_n\} \subseteq D$, the *glb* (also called *infimum* of S) exists and can be computed as $\bigsqcap S = e_1 \sqcap e_2 \sqcap \cdots \sqcap e_n$ (which reduces to $\mathbf{t}$ in the case $n = 0$). The dual claim concerning *lub*s is also true. As an easy consequence of the axioms, one gets the identity $d \circ \bigsqcap S = \bigsqcap \{d \circ e \mid e \in S\}$.

The following basic qualification domains were also introduced in Rodríguez-Artalejo and Romero-Díaz (2008).

The Domain of Classical Boolean Values is $\mathscr{B} =_{\text{def}} \langle \{0,1\}, \leqslant, 0, 1, \wedge \rangle$, where 0 and 1 stand for the two classical truth values *false* and *true*, $\leqslant$ is the usual numerical ordering over $\{0,1\}$, and $\wedge$ stands for the classical conjunction operation over $\{0,1\}$.

The Domain of Uncertainty Values is $\mathscr{U} =_{\text{def}} \langle \mathrm{U}, \leqslant, 0, 1, \times \rangle$, where $\mathrm{U} = [0,1] = \{d \in \mathbb{R} \mid 0 \leqslant d \leqslant 1\}$, $\leqslant$ is the usual numerical ordering, and $\times$ is the multiplication operation. The top element $\mathbf{t}$ is 1 and for any finite $S \subseteq \mathrm{U}$ one has $\bigsqcap S = \min(S)$, which is 1 if $S = \emptyset$. Elements of $\mathscr{U}$ are intended to represent certainty degrees.

The Domain of Weight Values is $\mathscr{W} =_{\text{def}} \langle \mathrm{P}, \geqslant, \infty, 0, + \rangle$, where $\mathrm{P} = [0, \infty] = \{d \in \mathbb{R} \cup \{\infty\} \mid d \geqslant 0\}$, $\geqslant$ is the reverse of the usual numerical ordering (with $\infty \geqslant d$ for any $d \in \mathrm{P}$), and $+$ is the addition operation (with $\infty + d = d + \infty = \infty$ for any $d \in \mathrm{P}$). The top element $\mathbf{t}$ is 0 and for any finite $S \subseteq \mathrm{P}$ one has $\bigsqcap S = \max(S)$, which is 0 if $S = \emptyset$. Elements of $\mathscr{W}$ are intended to represent proof costs, measured as the weighted depth of proof trees.

Given qualification domains $\mathscr{D}_1$ and $\mathscr{D}_2$, their *strict cartesian product* $\mathscr{D}_1 \otimes \mathscr{D}_2$ is $\mathscr{D} =_{\text{def}} \langle D, \trianglelefteq, \mathbf{b}, \mathbf{t}, \circ \rangle$ where $D = D_1 \otimes D_2 =_{\text{def}} ((D_1 \setminus \{\mathbf{b}_1\}) \times (D_2 \setminus \{\mathbf{b}_2\})) \cup \{(\mathbf{b}_1, \mathbf{b}_2)\}$, the partial ordering $\trianglelefteq$ is defined as $(d_1, d_2) \trianglelefteq (e_1, e_2) \iff_{\text{def}} d_1 \trianglelefteq_1 e_1$ and $d_2 \trianglelefteq_2 e_2$, and the attenuation operator $\circ$ is defined as $(d_1, d_2) \circ (e_1, e_2) =_{\text{def}} (d_1 \circ_1 e_1, d_2 \circ_2 e_2)$. It can be proved that $\mathscr{D}_1 \otimes \mathscr{D}_2$ is again a qualification domain[2].

In Section 3 we will need the following definition, that refines a similar one given in (Caballero *et al.* 2009).

Definition 2.1 (*Expressing $\mathscr{D}$ in $\mathscr{C}$*)
A qualification domain $\mathscr{D}$ is expressible in a constraint domain $\mathscr{C}$ if there is an injective embedding mapping $\iota : D \setminus \{\mathbf{b}\} \to C$ and moreover:

1. There is a $\mathscr{C}$-constraint $\mathsf{qVal}(X)$ such that $\mathrm{Sol}_{\mathscr{C}}(\mathsf{qVal}(X))$ is the set of all $\eta \in \mathrm{Val}_{\mathscr{C}}$ verifying $\eta(X) \in \mathit{ran}(\iota)$.

[1] The authors are thankful to G. Gerla for pointing out this fact.
[2] This result refines a similar one for ordinary cartesian products presented in (Rodríguez-Artalejo and Romero-Díaz 2008).

https://doi.org/10.1017/S1471068410000323 Published online by Cambridge University Press

2. There is a $\mathscr{C}$-constraint $\mathsf{qBound}(X, Y, Z)$ encoding "$x \trianglelefteq y \circ z$" in the following sense: any $\eta \in \mathrm{Val}_{\mathscr{C}}$ such that $\eta(X) = \iota(x)$, $\eta(Y) = \iota(y)$ and $\eta(Z) = \iota(z)$ verifies $\eta \in \mathrm{Sol}_{\mathscr{C}}(\mathsf{qBound}(X, Y, Z))$ iff $x \trianglelefteq y \circ z$.

In addition, if $\mathsf{qVal}(X)$ and $\mathsf{qBound}(X, Y, Z)$ can be chosen as existential constraints, we say that $\mathscr{D}$ is *existentially expressible* in $\mathscr{C}$. □

We can prove that $\mathscr{B}$, $\mathscr{U}$, $\mathscr{W}$ and any qualification domain built from these with the help of $\otimes$ is existentially expressible in any constraint domain $\mathscr{C}$ that includes the basic values and computational features of $\mathscr{R}$. For instance, $\mathscr{U} \otimes \mathscr{W}$ can be expressed in $\mathscr{R}$ using a binary data constructor $\mathsf{pair} \in DC^2$ and taking: $\iota(x, y) =_{\mathrm{def}} \mathsf{pair}(x, y)$; $\mathsf{qVal}(X) : \exists X_1 \exists X_2 (X == \mathsf{pair}(X_1, X_2) \wedge cp_{<}(0, X_1) \wedge cp_{\leqslant}(X_1, 1) \wedge cp_{\leqslant}(0, X_2))$; and $\mathsf{qBound}(X, Y, Z)$ built in a suitable way. The interested reader is referred to (Rodríguez-Artalejo and Romero-Díaz 2010) for other examples of qualification domains which can be existentially expressed in $\mathscr{FD}$.

2.3 Proximity relations

Similarity and proximity relations have been introduced in Section 1. In the rest of this paper we will focus on triples $\langle \mathscr{S}, \mathscr{D}, \mathscr{C} \rangle$ fulfilling the following requirements:

Definition 2.2 (*Admissible triples*)
An *admissible triple* $\langle \mathscr{S}, \mathscr{D}, \mathscr{C} \rangle$ consist of a constraint domain $\mathscr{C}$ with signature $\Sigma = \langle DC, DP, PP \rangle$ and set of basic values B, a qualification domain $\mathscr{D}$ expressible in $\mathscr{C}$ and a mapping $\mathscr{S} : S \times S \to D$ satisfying the following properties:

1. $S = \mathscr{V}ar \uplus B \uplus DC \uplus DP \uplus PP$.
2. $\mathscr{S}$ is a *$\mathscr{D}$-valued proximity relation* such that $\mathscr{S}(x, x) = \mathbf{t}$ (reflexivity) and $\mathscr{S}(x, y) = \mathscr{S}(y, x)$ (symmetry) hold for all $x, y \in S$. In the case that $\mathscr{S}$ satifies also $\mathscr{S}(x, z) \trianglerighteq \mathscr{S}(x, y) \sqcap \mathscr{S}(y, z)$ (transitivity) for all $x, y, z \in S$, it is called *$\mathscr{D}$-valued similarity relation.*
3. $\mathscr{S}$ restricted to $\mathscr{V}ar$ behaves as the identity, i.e. $\mathscr{S}(X, X) = \mathbf{t}$ for all $X \in \mathscr{V}ar$ and $\mathscr{S}(X, Y) = \mathbf{b}$ for all $X, Y \in \mathscr{V}ar$ such that $X \neq Y$.
4. For any $x, y \in S$, $\mathscr{S}(x, y) \neq \mathbf{b}$ can happen only if: (a) $x = y$ are identical; or else (b) $x, y \in B$ are basic values; or else (c) $x, y \in DC$ are data constructor symbols with the same arity; or else (d) $x, y \in DP$ are defined predicate symbols with the same arity. □

$\mathscr{D}$-valued proximity relations generalize the $\mathscr{D}$-valued similarity relations first introduced in (Caballero *et al.* 2008). When $\mathscr{D}$ is chosen as the qualification domain $\mathscr{U}$, the previous definition provides proximity and similarity relations in the sense of Zadeh (1971) and Dubois and Prade (1980). In this case, a proximity degree $\mathscr{S}(x, y) = d \in [0, 1]$ can be naturally interpreted as a *certainty degree* for the assertion that x and y are interchangeable. On the other hand, if $\mathscr{S}$ is $\mathscr{W}$-valued, then $\mathscr{S}(x, y) = d \in [0, \infty]$ can be interpreted as a *cost* to be paid for y to play the role of x.

As mentioned in the Introduction, the transitivity property postulated for similarity relations may be counterintuitive in some cases. For instance, assume nullary

https://doi.org/10.1017/S1471068410000323 Published online by Cambridge University Press

constructors `colt`, `cold` and `gold` intended to represent words composed of four letters. Then, measuring the proximity between such words might reasonably lead to a $\mathscr{U}$-valued proximity relation $\mathscr{S}$ such that $\mathscr{S}(\texttt{colt},\texttt{cold}) = 0.9$, $\mathscr{S}(\texttt{cold},\texttt{gold}) = 0.9$ and $\mathscr{S}(\texttt{colt},\texttt{gold}) = 0.4$. On the other hand, insisting on $\mathscr{S}$ to be transitive would enforce the unreasonable condition $\mathscr{S}(\texttt{colt},\texttt{gold}) \geqslant 0.9$. Therefore, a similarity relation would not be appropriate in this case.

The special mapping $\mathscr{S}_{\mathrm{id}} : S \times S \to D$ defined as $\mathscr{S}_{\mathrm{id}}(x,x) = \mathbf{t}$ for all $x \in S$ and $\mathscr{S}_{\mathrm{id}}(x,y) = \mathbf{b}$ for all $x, y \in S$, $x \neq y$ is trivially a $\mathscr{D}$-valued similarity (and therefore, also a proximity) relation called the *identity*.

In the rest of this paper, the notations $\mathscr{S}$, $\mathscr{D}$ and $\mathscr{C}$ are always understood as the components of some given admissible triple and the proximity relation $\mathscr{S}$ is not required to be transitive. As noted in Sessa (2002) and related works, $\mathscr{S}$ can be naturally extended to act over terms. The extension, also noted $\mathscr{S}$, works as specified by the recursive equations displayed below:

- $\mathscr{S}(t,t) = \mathbf{t}$ for every term t.
- $\mathscr{S}(X,t) = \mathscr{S}(t,X) = \mathbf{b}$ for $X \in \mathscr{V}ar$ and for any term $t \neq X$.
- $\mathscr{S}(c(\bar{t}_n), c'(\bar{t'}_n) = \mathbf{b}$ for $c \in DC^n, c' \in DC^m$ with $n \neq m$.
- $\mathscr{S}(c(\bar{t}_n), c'(\bar{t'}_n)) = \mathscr{S}(c,c') \sqcap \mathscr{S}(t_1,t'_1) \sqcap \ldots \sqcap \mathscr{S}(t_n,t'_n)$ for $c, c' \in DC^n$.

Analogously, $\mathscr{S}$ can be extended to work over atoms and other syntactic objects. The following definition combines $\mathscr{S}$ with constraint entailment, leading to a kind of relations over terms which will play a crucial role for the semantics of equations in SQCLP.

Definition 2.3 (*Constraint-based term proximity at level* λ)
Assume $\lambda \in D \setminus \{\mathbf{b}\}$ and $\Pi \subseteq \mathrm{Con}_{\mathscr{C}}$. We will say that two terms t and s are $\mathscr{S}$-*close at level* λ *w.r.t.* Π (in symbols, $t \approx_{\lambda,\Pi} s$) iff there are two terms $\hat{t}$, $\hat{s}$ such that $\Pi \models_{\mathscr{C}} t == \hat{t}$, $\Pi \models_{\mathscr{C}} s == \hat{s}$ and $\mathscr{S}(\hat{t},\hat{s}) \trianglerighteq \lambda$. □

It can be proved that $\approx_{\lambda,\Pi}$ is a reflexive and symmetric relation over the set of terms, that is even transitive in case that $\mathscr{S}$ is a similarity relation. As a simple example, assume $\mathscr{D} = \mathscr{U}$, $\mathscr{C} = \mathscr{R}$ and $\mathscr{S}$ such that $\mathscr{S}(c',c) = \mathscr{S}(c,c'') = 0.8$ and $\mathscr{S}(c',c'') = 0.6$ for some $c, c', c'' \in DC^2$. Let $\Pi = \{op_+(A,A,X), op_\times(2.0,A,Y), Z == c(X,Y)\} \subseteq \mathrm{Con}_{\mathscr{R}}$. Note that this choice of Π ensures $\Pi \models_{\mathscr{R}} X == Y$. Then $c'(Y,X) \approx_{0.7,\Pi} Z$ holds, because $\Pi \models_{\mathscr{R}} c'(Y,X) == c'(X,X)$, $\Pi \models_{\mathscr{R}} Z == c(X,X)$ and $\mathscr{S}(c'(X,X), c(X,X)) = 0.8 \geqslant 0.7$.

3 The SQCLP scheme

3.1 Programs, interpretations and models

The scheme SQCLP has instances SQCLP($\mathscr{S},\mathscr{D},\mathscr{C}$) where $\langle \mathscr{S},\mathscr{D},\mathscr{C}\rangle$ is an admissible triple. A SQCLP($\mathscr{S},\mathscr{D},\mathscr{C}$)-program is a set $\mathscr{P}$ of *qualified program rules* (also called *qualified clauses*) $C : A \overset{\alpha}{\leftarrow} B_1 \sharp w_1, \ldots, B_m \sharp w_m$, where A is a defined atom, $\alpha \in D \setminus \{\mathbf{b}\}$ is called the *attenuation factor* of the clause and each $B_j \sharp w_j$ ($1 \leqslant j \leqslant m$) is an atom B_j annotated with a so-called *threshold value* $w_j \in (D \setminus \{\mathbf{b}\}) \uplus \{?\}$. The intended meaning of C is as follows: if for all $1 \leqslant j \leqslant m$ one has $B_j \sharp e_j$ (meaning that B_j holds with qualification value e_j) for some $e_j \trianglerighteq^? w_j$, then $A \sharp d$ (meaning that A holds with

https://doi.org/10.1017/S1471068410000323 Published online by Cambridge University Press

qualification value d) can be inferred for any $d \in D \setminus \{\mathbf{b}\}$ such that $d \trianglelefteq \alpha \circ \bigsqcap_{j=1}^{m} e_j$. By convention, $e_j \trianglerighteq^{?} w_j$ means $e_j \trianglerighteq w_j$ if $w_j \neq ?$ and is identically true otherwise. In practice threshold values equal to '?' and attenuation values equal to $\mathbf{t}$ can be omitted.

As motivating example, consider a SQCLP($\mathscr{S}, \mathscr{U} \otimes \mathscr{W}, \mathscr{R}$)-program $\mathscr{P}$ including the clauses and equations for $\mathscr{S}$ displayed in Figure 1. From Subsection 2.2 recall that qualification values in $\mathscr{U} \otimes \mathscr{W}$ are pairs (d, e) (where d represents a certainty degree and e represents a proof cost), as well as the behavior of $\trianglelefteq$ and $\circ$ in $\mathscr{U} \otimes \mathscr{W}$. Consider the problem of proving `goodWork(king_liar)`$\sharp(d, e)$ from $\mathscr{P}$. This can be achieved for $d = 0.75 \times \min\{d_1, d_2\}$, $e = 3 + \max\{e_1, e_2\}$ by using R_1 instantiated by $\{\mathtt{X} \mapsto \mathtt{king\_liar}, \mathtt{Y} \mapsto \mathtt{shakespeare}\}$, and going on to prove `famousAuthor(shakespeare)`$\sharp(d_1, e_1)$ for some $d_1 \geqslant 0.5$, $e_1 \leqslant 100$ and `wrote(shakespeare,king_liar)`$\sharp(d_2, e_2)$ for some d_2, e_2. Thanks to R_2, R_3 and $\mathscr{S}$, these proofs succeed with $(d_1, e_1) = (0.9, 1)$ and $(d_2, e_2) = (0.8, 2)$. Therefore, the desired proof succeeds with certainty degree $d = 0.75 \times \min\{0.9, 0.8\} = 0.6$, and proof cost $e = 3 + \max\{1, 2\} = 5$.

```
R1 : goodWork(X) <-(0.75,3)- famousAuthor(Y)#(0.5,100), wrote(Y,X)#?
R2 : famousAuthor(shakespeare) <-(0.9,1)-
R3 : wrote(shakespeare,king_lear) <-(1,1)-

S(king_lear,king_liar) = (0.8,2)
```

Fig. 1. SQCLP($\mathscr{S}, \mathscr{U} \otimes \mathscr{W}, \mathscr{R}$) Program Fragment

The more technical SQCLP($\mathscr{S}, \mathscr{U}, \mathscr{R}$)-program $\mathscr{P}$ presented below will serve as a *running example* in the rest of the paper.

Example 3.1 (*Running example*)
Assume $c, c' \in DC^1$, $p, p', q \in DP^2$, $r \in DP^3$ and $\mathscr{S}$ such that $\mathscr{S}(c, c') = 0.9$ and $\mathscr{S}(p, p') = 0.8$. Let $\mathscr{P}$ consist of the following program rules:

$$R_1 : q(X, c(X)) \xleftarrow{1.0} \qquad R_3 : r(c(X), Y, Z) \xleftarrow{0.9} q(X, Y)\sharp 0.8, cp_{\geqslant}(X, 0.0)\sharp ?$$
$$R_2 : p(c(X), Y) \xleftarrow{0.9} q(X, Y)\sharp 0.8 \quad \square$$

The declarative semantics for SQCLP presented in the rest of this section is inspired by the $\mathscr{S}_2$ semantics for CLP given in Gabbrielli *et al.* (1995). We use *qualified constrained atoms* (or simply *qc-atoms*) of the form $A\sharp d \Leftarrow \Pi$, intended to assert that the validity of atom A with qualification degree $d \in D$ is entailed by the constraint set $\Pi \subseteq \mathrm{Con}_{\mathscr{C}}$. A qc-atom is called *defined*, *primitive* or *equational* according to the syntactic form of A; and it is called *observable* iff $d \in D \setminus \{\mathbf{b}\}$ and Π is satisfiable. In the sequel we restrict our attention to observable qc-atoms, viewing them as observations of computed answers for atomic goals. We use an *entailment relation* $\succcurlyeq_{\mathscr{D},\mathscr{C}}$ to capture some implications between qc-atoms whose validity depends neither on the proximity relation $\mathscr{S}$ nor on program clauses[3]. Formally, given $\varphi : A\sharp d \Leftarrow \Pi$ and $\varphi' : A'\sharp d' \Leftarrow \Pi'$, we say that φ $(\mathscr{D}, \mathscr{C})$-*entails* φ'

[3] In Caballero *et al.* (2008) we used a different entailment relation that depends on $\mathscr{S}$ and does not work properly if $\mathscr{S}$ is not transitive.

https://doi.org/10.1017/S1471068410000323 Published online by Cambridge University Press

(in symbols, $\varphi \succcurlyeq_{\mathscr{D},\mathscr{C}} \varphi'$) iff there is some substitution θ such that $A' = A\theta$, $d' \trianglelefteq d$ and $\Pi' \models_{\mathscr{C}} \Pi\theta$. The example below illustrates these notions:

Example 3.2 (*Observable qc-atoms and* $(\mathscr{D},\mathscr{C})$*-entailment*)
Building upon Example 3.1, let $\Pi = \{cp_{>}(X, 1.0),\ op_{+}(A, A, X),\ op_{\times}(2.0, A,\ Y)\}$ and $\Pi' = \{cp_{\geqslant}(A, 3.0),\ op_{\times}(2.0,\ A, X),\ op_{+}(A, A, Y)\}$. Then, the following are observable qc-atoms:

$$\varphi_1 : q(X, c'(Y))\sharp 0.9 \Leftarrow \Pi \qquad \varphi_3 : r(c'(Y), c(X), Z)\sharp 0.8 \Leftarrow \Pi$$
$$\varphi_2 : p'(c'(Y), c(X))\sharp 0.8 \Leftarrow \Pi \qquad \varphi'_3 : r(c'(Y), c(X), c(Z'))\sharp 0.7 \Leftarrow \Pi'$$

and $\varphi_3 \succcurlyeq_{\mathscr{U},\mathscr{R}} \varphi'_3$ holds, since $\theta = \{Z \mapsto c(Z')\}$ verifies $r(c'(Y), c(X),\ c(Z')) = r(c'(Y), c(X), Z)\theta$, $0.7 \leqslant 0.8$ and $\Pi' \models_{\mathscr{R}} \Pi\theta$. □

The intended meaning of $\succcurlyeq_{\mathscr{D},\mathscr{C}}$ motivates the first sentence in the next definition.

Definition 3.1 (*Interpretations*)
A *qualified constrained interpretation* (or qc-interpretation) is a set $\mathscr{I}$ of defined observable qc-atoms closed under $(\mathscr{D},\mathscr{C})$-entailment, i.e. $\varphi \in \mathscr{I}$ and $\varphi \succcurlyeq_{\mathscr{D},\mathscr{C}} \varphi'$ implies $\varphi' \in \mathscr{I}$. An observable qc-atom φ is called *valid* in the qc-interpretation $\mathscr{I}$ (in symbols, $\mathscr{I} \Vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi$) iff some of the following cases holds: (a) φ is a defined qc-atom and $\varphi \in \mathscr{I}$; or (b) φ is an equational qc-atom $(t == s)\sharp d \Leftarrow \Pi$ and $t \approx_{d,\Pi} s$; or (c) φ is a primitive qc-atom $\kappa\sharp d \Leftarrow \Pi$ and $\Pi \models_{\mathscr{C}} \kappa$. □

Note that a given interpretation $\mathscr{I}$ can include several observables $A\sharp d_i \Leftarrow \Pi$ for the same (possibly open) atom A, but is not required to include one "optimal" observable $A\sharp d \Leftarrow \Pi$ with d computed as the *lub* of all d_i. By contrast, the other related works discussed in the Introduction view program interpretations as mappings $\mathscr{I}$ from the ground Herbrand base into some set of lattice elements (the real interval $[0, 1]$ in many cases). In such interpretations, each ground atom A has attached one single lattice element $d = \mathscr{I}(A)$ intended as "the optimal qualification" for A. Our view of interpretations is closer to the expected operational behavior of goal solving systems and can be used to characterize the validity of solutions computed by such systems, as we will see in Subsection 3.4.

It can be proved that $\mathscr{I} \Vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi$ implies $\mathscr{I} \Vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi'$ for any φ' such that $\varphi \succcurlyeq_{\mathscr{D},\mathscr{C}} \varphi'$ (so-called *entailment property for interpretations*). The notions of model and semantic consequence are defined below.

Definition 3.2 (*Models and semantic consequence*)
Let a SQCLP$(\mathscr{S},\mathscr{D},\mathscr{C})$-program $\mathscr{P}$ and an observable qc-atom $\varphi : p'(\overline{t'}_n)\sharp d \Leftarrow \Pi$ be given. φ is an *immediate consequence* of a qc-interpretation $\mathscr{I}$ via a program rule $(R_l : p(\overline{t}_n) \overset{\alpha}{\leftarrow} B_1\sharp w_1, \ldots, B_m\sharp w_m) \in \mathscr{P}$ iff there exist a $\mathscr{C}$-substitution θ and a choice of qualification values $d_0, d_1, \ldots, d_n, e_1, \ldots, e_m \in D \setminus \{\mathbf{b}\}$ such that:

(a) $\mathscr{S}(p', p) = d_0$,
(b) $\mathscr{I} \Vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} (t'_i == t_i\theta)\sharp d_i \Leftarrow \Pi$ (i.e. $t'_i \approx_{d_i,\Pi} t_i\theta$) for $i = 1 \ldots n$,
(c) $\mathscr{I} \Vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} B_j\theta\sharp e_j \Leftarrow \Pi$ with $e_j \rhd^{?} w_j$ for $j = 1 \ldots m$,
(d) $d \trianglelefteq \bigsqcap_{i=0}^{n} d_i \sqcap \alpha \circ \bigsqcap_{j=1}^{m} e_j$.

Note that the qualification value d attached to φ is limited by two kinds of upper bounds: d_i $(0 \leqslant i \leqslant n)$, i.e. the $\mathscr{S}$-proximity between $p'(\overline{t'}_n)$ and the head of $R_l\theta$;

https://doi.org/10.1017/S1471068410000323 Published online by Cambridge University Press

and $\alpha \circ e_j$ $(1 \leqslant j \leqslant m)$, i.e. the qualification values of the atoms in the body of $R_l\theta$ attenuated w.r.t. R_l's attenuation factor α. Now we can define:

1. $\mathscr{I}$ is a *model* of $R_l \in \mathscr{P}$ (in symbols, $\mathscr{I} \models_{\mathscr{S},\mathscr{D},\mathscr{C}} R_l$) iff every defined observable qc-atom φ that is an immediate consequence of $\mathscr{I}$ via R_l verifies $\varphi \in \mathscr{I}$. And $\mathscr{I}$ is a *model* of $\mathscr{P}$ (in symbols, $\mathscr{I} \models_{\mathscr{S},\mathscr{D},\mathscr{C}} \mathscr{P}$) iff $\mathscr{I}$ is a model of each $R_l \in \mathscr{P}$.
2. φ is a *semantic consequence* of $\mathscr{P}$ (in symbols, $\mathscr{P} \models_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi$) iff $\mathscr{I} \models_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi$ for every qc-interpretation $\mathscr{I}$ such that $\mathscr{I} \models_{\mathscr{S},\mathscr{D},\mathscr{C}} \mathscr{P}$. □

The next example may serve as a concrete illustration:

Example 3.3 (*Models and semantic consequence*)
Recall the SQCLP($\mathscr{S},\mathscr{U},\mathscr{R}$)-program $\mathscr{P}$ from Example 3.1 and the qc-atoms φ_1 and φ_2 from Example 3.2. Assume an arbitrary model $\mathscr{I} \models_{\mathscr{S},\mathscr{U},\mathscr{R}} \mathscr{P}$. Then:
— (1) Note that the atom underlying φ_1 is $q(X, c'(Y))$, and the head atom of R_1 is $q(X, c(X))$. Since $\mathscr{S}(c, c') = 0.9$ and $\Pi \models_{\mathscr{C}} X == Y$, φ_1 can be obtained as an immediate consequence of $\mathscr{I}$ via R_1 using $\theta = \varepsilon$. Therefore, $\varphi_1 \in \mathscr{I}$ and $\mathscr{P} \models_{\mathscr{S},\mathscr{U},\mathscr{R}} \varphi_1$.
— (2) Consider $\theta = \{Y \mapsto c'(Y)\}$. Note that $p'(c'(Y), c(X))$ is the atom underlying φ_2, and the head atom of $R_2\theta$ is $p(c(X), c'(Y))$. Moreover, $\varphi_1 \in \mathscr{I}$ due to the previous item and the atom $q(X, c'(Y))$ underlying φ_1 is the same as the atom in the body of $R_2\theta$. These facts together with $\mathscr{S}(p, p') = 0.8$, $\mathscr{S}(c, c') = 0.9$ and $\Pi \models_{\mathscr{C}} X == Y$ allow to obtain φ_2 as an immediate consequence of $\mathscr{I}$ via R_2. Therefore, $\varphi_2 \in \mathscr{I}$ and $\mathscr{P} \models_{\mathscr{S},\mathscr{U},\mathscr{R}} \varphi_2$. □

3.2 A fixpoint semantics

As for other declarative languages, one can use immediate consequence operators to characterize the models and least models of a given SQCLP($\mathscr{S},\mathscr{D},\mathscr{C}$)-program $\mathscr{P}$. We start by considering the complete lattice $\mathrm{Int}_{\mathscr{D},\mathscr{C}}$ of all qc-interpretations partially ordered by set inclusion, with bottom element $\perp\!\!\!\perp = \emptyset$ and top element $\top\!\!\!\top = \{\varphi \mid \varphi$ is a defined observable qc-atom$\}$. For any subset $I \subseteq \mathrm{Int}_{\mathscr{D},\mathscr{C}}$ one gets the greatest lower bound $\bigsqcap I = \bigcap_{\mathscr{I} \in I} \mathscr{I}$ and the least upper bound $\bigsqcup I = \bigcup_{\mathscr{I} \in I} \mathscr{I}$. Next we define an *interpretation transformer* $\mathrm{T}_{\mathscr{P}} : \mathrm{Int}_{\mathscr{D},\mathscr{C}} \to \mathrm{Int}_{\mathscr{D},\mathscr{C}}$, intended to compute the immediate consequences obtained from a given qc-interpretation via the program rules belonging to $\mathscr{P}$, and defined as

$$\mathrm{T}_{\mathscr{P}}(\mathscr{I}) =_{\mathrm{def}} \{\varphi \mid \varphi \text{ is an immediate consequence of } \mathscr{I} \text{ via some } R_l \in \mathscr{P}\}$$

where immediate consequences are computed as explained in Definition 3.2. The following example illustrates the workings of $\mathrm{T}_{\mathscr{P}}$.

Example 3.4 (*Interpretation transformer in action*)
Recall again the SQCLP($\mathscr{S},\mathscr{U},\mathscr{R}$)-program $\mathscr{P}$ from Example 3.1 and the defined observable qc-atoms φ_1 and φ_2 from Example 3.2. Then: (1) The arguments given in Example 3.3(1) can be easily reused to show that φ_1 is an immediate consequence of $\perp\!\!\!\perp$ via R_1. Therefore, $\varphi_1 \in \mathrm{T}_{\mathscr{P}}(\perp\!\!\!\perp)$. (2) The arguments given in Example 3.3(2) can be easily reused to show that φ_2 is an immediate consequence of $\mathscr{I}$ via R_2, provided that $\varphi_1 \in \mathscr{I}$. Therefore, $\varphi_2 \in \mathrm{T}_{\mathscr{P}}(\mathrm{T}_{\mathscr{P}}(\perp\!\!\!\perp))$. □

The next proposition states the main properties of interpretation transformers.

https://doi.org/10.1017/S1471068410000323 Published online by Cambridge University Press

Proposition 3.1 (*Properties of interpretation transformers*)
For any SQCLP($\mathscr{S},\mathscr{D},\mathscr{C}$)-program $\mathscr{P}$, $\mathrm{T}_{\mathscr{P}}$ is a well defined mapping, i.e. for all $\mathscr{I} \in \mathrm{Int}_{\mathscr{D},\mathscr{C}}$ one has $\mathrm{T}_{\mathscr{P}}(\mathscr{I}) \in \mathrm{Int}_{\mathscr{D},\mathscr{C}}$. Moreover, $\mathrm{T}_{\mathscr{P}}$ is monotonic and continuous and its pre-fixpoints are the models of $\mathscr{P}$, i.e. for all $\mathscr{I} \in \mathrm{Int}_{\mathscr{D},\mathscr{C}}$ one has $\mathscr{I} \models_{\mathscr{S},\mathscr{D},\mathscr{C}} \mathscr{P} \iff \mathrm{T}_{\mathscr{P}}(\mathscr{I}) \subseteq \mathscr{I}$. □

As an immediate consequence one can prove the theorem below, that is the main result in this subsection.

Theorem 3.1 (*Fixpoint characterization of least program models*)
Every SQCLP($\mathscr{S},\mathscr{D},\mathscr{C}$)-program $\mathscr{P}$ has a *least model* $\mathscr{M}_{\mathscr{P}}$, smaller than any other model of $\mathscr{P}$ w.r.t. the set inclusion ordering of the interpretation lattice $\mathrm{Int}_{\mathscr{D},\mathscr{C}}$. Moreover, $\mathscr{M}_{\mathscr{P}}$ can be characterized as *least fixpoint* of $\mathrm{T}_{\mathscr{P}}$ as follows:

$$\mathscr{M}_{\mathscr{P}} = lfp(\mathrm{T}_{\mathscr{P}}) = \bigcup_{k\in\mathbb{N}} \mathrm{T}_{\mathscr{P}}\uparrow^k(\perp\!\!\!\perp) \ . \quad \square$$

3.3 An equivalent proof-theoretic semantics

In order to give a logical view of program semantics and an alternative characterization of least program models, we define the *Proximity-based Qualified Constrained Horn Logic* SQCHL($\mathscr{S},\mathscr{D},\mathscr{C}$) as a formal inference system consisting of the three inference rules displayed in Figure 2.

SQDA $$\frac{(\ (t'_i == t_i\theta)\sharp d_i \Leftarrow \Pi\ )_{i=1\ldots n} \quad (\ B_j\theta\sharp e_j \Leftarrow \Pi\ )_{j=1\ldots m}}{p'(\overline{t'}_n)\sharp d \Leftarrow \Pi}$$

if $(p(\bar{t}_n) \overset{\alpha}{\leftarrow} B_1\sharp w_1,\ldots,B_m\sharp w_m) \in \mathscr{P}$, θ subst., $\mathscr{S}(p',p) = d_0 \neq \mathbf{b}$, $e_j \trianglerighteq^? w_j$ $(1 \leqslant j \leqslant m)$ and $d \trianglelefteq \bigsqcap_{i=0}^{n} d_i \sqcap \alpha \circ \bigsqcap_{j=1}^{m} e_j$.

SQEA $$\frac{}{(t == s)\sharp d \Leftarrow \Pi}$$ if $t \approx_{d,\Pi} s$. **SQPA** $$\frac{}{\kappa\sharp d \Leftarrow \Pi}$$ if $\Pi \models_{\mathscr{C}} \kappa$.

Fig. 2. Proximity-based Qualified Constrained Horn Logic

Rule **SQDA** formalizes an extension of the classical *Modus Ponens* inference allowing to infer a defined qc-atom $p'(\overline{t'}_n)\sharp d \Leftarrow \Pi$ by means of an instantiated clause with head $p(\bar{t}_n)\theta$ and body atoms $B_j\theta\sharp w_j$. The n premises $(t'_i == t_i\theta)\sharp d_i \Leftarrow \Pi$ combined with the side condition $\mathscr{S}(p',p) = d_0 \neq \mathbf{b}$ ensure the "equality" between $p'(\overline{t'}_n)$ and $p(\bar{t}_n)\theta$ modulo $\mathscr{S}$; the m premises $B_j\theta\sharp e_j \Leftarrow \Pi$ require to prove the body atoms; and the side conditions $e_j \trianglerighteq^? w_j$ and $d \trianglelefteq \bigsqcap_{i=0}^{n} d_i \sqcap \alpha \circ \bigsqcap_{j=1}^{m} e_j$ check the threshold conditions of the body atoms and impose the proper relationships between the qualification value d attached to the conclusion and the qualification values d_i and e_j attached to the premises. Rule **SQEA** is designed to work with constraint-based term proximity in the sense of Definition 2.3, inferring $(t == s)\sharp d \Leftarrow \Pi$ just in the case that $t \approx_{d,\Pi} s$ holds. Rule **SQPA** infers primitive qc-atoms $\kappa\sharp d \Leftarrow \Pi$ for an arbitrary $d \in D \setminus \{\mathbf{b}\}$, provided that $\Pi \models_{\mathscr{C}} \kappa$ holds.

We will write $\mathscr{P} \vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi$ to indicate that φ can be deduced from $\mathscr{P}$ in SQCHL($\mathscr{S}$, $\mathscr{D},\mathscr{C}$), and $\mathscr{P} \vdash^k_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi$ in the case that the deduction can be performed with exactly

https://doi.org/10.1017/S1471068410000323 Published online by Cambridge University Press

k **SQDA** inference steps. As usual in formal inference systems, SQCHL($\mathscr{S},\mathscr{D},\mathscr{C}$) proofs can be represented as *proof trees* whose nodes correspond to qc-atoms, each node being inferred from its children by means of some SQCHL($\mathscr{S},\mathscr{D},\mathscr{C}$) inference step. The next example shows a simple SQCHL($\mathscr{S},\mathscr{U},\mathscr{R}$) proof tree.

Example 3.5 (*SQCHL*($\mathscr{S},\mathscr{D},\mathscr{C}$) *proof tree*)
Recall the proximity relation $\mathscr{S}$ and the program $\mathscr{P}$ from our running example 3.1 and the observable qc-statement $\varphi_1 = q(X, c'(Y)) \sharp 0.9 \Leftarrow \Pi$ already known from Example 3.2. A SQCHL($\mathscr{S},\mathscr{U},\mathscr{R}$) proof tree witnessing $\mathscr{P} \vdash^1_{\mathscr{S},\mathscr{U},\mathscr{R}} \varphi_1$ can be displayed as follows:

$$\dfrac{\dfrac{}{(X == X) \sharp 1.0 \Leftarrow \Pi}\ \textbf{SQEA(2)} \qquad \dfrac{}{(c'(Y) == c(X)) \sharp 0.9 \Leftarrow \Pi}\ \textbf{SQEA(3)}}{q(X, c'(Y)) \sharp 0.9 \Leftarrow \Pi}\ \textbf{SQDA(1)}$$

Where: step (1) uses $R_1 = q(X,\ c(X)) \overset{1.0}{\longleftarrow}$ instantiated by the empty substitution (note that $0.9 \leqslant \min\{1.0, 0.9\}$); step (2) uses $X \approx_{1.0,\Pi} X$, trivially true; and step (3) uses $c'(Y) \approx_{0.9,\Pi} c(X)$, true due to $\mathscr{S}$(c,c') = 0.9 and $\Pi \models_{\mathscr{R}} X == Y$. □

It can be proved that $\mathscr{P} \vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi$ implies $\mathscr{P} \vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi'$ for any φ' such that $\varphi \succcurlyeq_{\mathscr{D},\mathscr{C}} \varphi'$ (so-called *entailment property for programs*). Moreover, if φ is either equational or primitive, then $\mathscr{P} \vdash^0_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi \iff \mathscr{P} \vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi \iff \mathscr{I} \Vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi$ for any program $\mathscr{P}$ and any qc-interpretation $\mathscr{I}$. The following theorem is the main result in this subsection.

Theorem 3.2 (*Logical characterization of least program models*)
For any SQCLP($\mathscr{S},\mathscr{D},\mathscr{C}$)-program $\mathscr{P}$, its least model can be characterized as:

$$\mathscr{M}_{\mathscr{P}} = \{\varphi \mid \varphi \text{ is an observable defined qc-atom and } \mathscr{P} \vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi\}. \quad \square$$

As an easy consequence of the previous theorem we can prove:

Corollary 3.1 (*SQCHL*($\mathscr{S},\mathscr{D},\mathscr{C}$) *is sound and complete*)
For any SQCLP($\mathscr{S},\mathscr{D},\mathscr{C}$)-program $\mathscr{P}$ and any observable qc-atom φ one has:

1. $\mathscr{P} \vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi \iff \mathscr{P} \models_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi \iff \mathscr{M}_{\mathscr{P}} \Vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi$.
2. $\mathscr{P} \vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi \Longrightarrow \mathscr{P} \models_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi$ (*soundness*).
3. $\mathscr{P} \models_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi \Longrightarrow \mathscr{P} \vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} \varphi$ (*completeness*). □

3.4 Goals and solutions

In order to build goals for SQCLP($\mathscr{S},\mathscr{D},\mathscr{C}$)-programs, we assume a countably infinite set $\mathscr{W}ar$ of so-called *qualification variables* W. Goals for a given program $\mathscr{P}$ have the form:

$$G \ :\ A_1 \sharp W_1,\ \ldots,\ A_m \sharp W_m \ [\!]\ W_1 \trianglerighteq^? \beta_1,\ \ldots,\ W_m \trianglerighteq^? \beta_m$$

abbreviated as $(A_i \sharp W_i, W_i \trianglerighteq^? \beta_i)_{i=1\ldots m}$ with annotated atoms $A_i \sharp W_i$ (where the qualification variables $W_i \in \mathscr{W}ar$ are pairwise different) and threshold conditions $W_i \trianglerighteq^? \beta_i$ with $\beta_i \in (D \setminus \{\mathbf{b}\}) \uplus \{?\}$. The notations ? and $\trianglerighteq^?$ have been explained in Subsection 3.1.

The proof-theoretical semantics developed in Subsection 3.3 allows to characterize *goal solutions* in a natural and declarative way by means of the following definition:

https://doi.org/10.1017/S1471068410000323 Published online by Cambridge University Press

the set of solutions of a goal G w.r.t. program $\mathscr{P}$ is noted $\mathrm{Sol}_{\mathscr{P}}(G)$ and consists of all triples $\langle \sigma, \mu, \Pi \rangle$ such that σ is a $\mathscr{C}$-substitution (not required to be ground), $\mu : \{W_1, \ldots, W_m\} \to D \setminus \{\mathbf{b}\}$, Π is a satisfiable and finite set of atomic $\mathscr{C}$-constraints and the following two conditions hold for all $i = 1 \ldots m$: $W_i\mu = d_i \trianglerighteq^? \beta_i$ and $\mathscr{P} \vdash_{\mathscr{S},\mathscr{D},\mathscr{C}} A_i\sigma \sharp W_i\mu \Leftarrow \Pi$. Although operational semantics is not investigated in this paper, *computed answers* obtained by means of a correct goal solving system for SQCLP($\mathscr{S}, \mathscr{D}, \mathscr{C}$) are expected to be valid solutions in this sense.

For instance, G : `goodWork(X)#W [] W |> (0.55,30)` is a goal for the program fragment $\mathscr{P}$ shown in Figure 1, and the arguments given near the beginning of Subsection 3.1 can be formalized to prove that $\langle \{\mathtt{X} \mapsto \mathtt{king\_liar}\}, \{\mathtt{W} \mapsto \mathtt{(0.6,5)}\}, \emptyset \rangle \in \mathrm{Sol}_{\mathscr{P}}(G)$.

As an additional example involving constraints, recall the SQCLP($\mathscr{S}, \mathscr{U}, \mathscr{R}$) program $\mathscr{P}$ presented in Example 3.1 and consider the goal $G : q(X, Z)\sharp W \,[]\, W \geqslant 0.8$ for $\mathscr{P}$. Then $\langle \sigma, \mu, \Pi \rangle \in \mathrm{Sol}_{\mathscr{P}}(G)$, where $\sigma = \{Z \mapsto c'(Y)\}$, $\mu = \{W \mapsto 0.9\}$ and $\Pi = \{cp_{>}(X, 1.0),\ op_{+}(A, A, X),\ op_{\times}(2.0, A, Y)\}$. Note that $W\mu = 0.9 \geqslant 0.8$ and $\mathscr{P} \vdash_{\mathscr{S},\mathscr{U},\mathscr{R}} q(X, Z)\sigma \sharp 0.9 \Leftarrow \Pi$ is known from Example 3.5.

4 Conclusions

We have extended the classical CLP scheme to a new scheme SQCLP whose instances SQCLP($\mathscr{S}, \mathscr{D}, \mathscr{C}$) are parameterized by a proximity relation $\mathscr{S}$, a qualification domain $\mathscr{D}$ and a constraint domain $\mathscr{C}$. In addition to the known features of CLP programming, the new scheme offers extra facilities for dealing with expert knowledge representation and flexible query answering. Inspired by the observable CLP semantics in Gabbrielli *et al.* (1995), we have presented a declarative semantics for SQCLP that provides fixpoint and proof-theoretical characterizations of least program models as well as an implementation-independent notion of goal solutions.

SQCLP is a quite general scheme. Different partial instantiations of its three parameters lead to more particular schemes, most of which can be placed in close correspondence to previous proposals. The items below present seven particularizations, along with some comments which make use of the following terminology: a SCQLP program is called *threshold-free* in case that all its clauses use only '?' as threshold value; *attenuation-free* in case that all its clauses use only **t** as attenuation value; and *constraint-free* in case that no constraints occur in clause bodies.

1. By definition, QCLP has instances QCLP($\mathscr{D}, \mathscr{C}$) $=_{\mathrm{def}}$ SQCLP($\mathscr{S}_{\mathrm{id}}, \mathscr{D}, \mathscr{C}$) where $\mathscr{S}_{\mathrm{id}}$ is the *identity* proximity relation. The *quantitative* CLP scheme proposed in Riezler (1998) can be understood as a further particularization of QCLP that works with threshold-free QCLP($\mathscr{U}, \mathscr{C}$) programs, where $\mathscr{U}$ is the qualification domain of uncertainty values (see Subsection 2.2).
2. By definition, SQLP has instances SQLP($\mathscr{S}, \mathscr{D}$) $=_{\mathrm{def}}$ SQCLP($\mathscr{S}, \mathscr{D}, \mathscr{R}$) where $\mathscr{R}$ is the real constraint domain (see Subsection 2.1). The scheme with the same name originally proposed in Caballero *et al.* (2008) can be understood as a restricted form of the present formulation; it worked with threshold-free and constraint-free SQLP($\mathscr{S}, \mathscr{D}$) programs and it restricted the choice of the $\mathscr{S}$ parameter to transitive proximity (i.e. similarity) relations.

https://doi.org/10.1017/S1471068410000323 Published online by Cambridge University Press

3. By definition, SCLP[4] has instances SCLP($\mathscr{S},\mathscr{C}$) $=_{\text{def}}$ SQCLP($\mathscr{S},\mathscr{B},\mathscr{C}$) where $\mathscr{B}$ is the qualification domain of classical boolean values (see Subsection 2.2). Due to the fixed parameter choice $\mathscr{D} = \mathscr{B}$, both attenuation values and threshold values become useless, and each choice of $\mathscr{S}$ must necessarily represent a crisp reflexive and symmetric relation. Therefore, this new scheme is not so interesting from the viewpoint of uncertain and qualified reasoning.
4. By definition, QLP has instances QLP($\mathscr{D}$) $=_{\text{def}}$ SQCLP($\mathscr{S}_{\text{id}},\mathscr{D},\mathscr{R}$). The original scheme with the same name proposed in Rodríguez-Artalejo and Romero-Díaz (2008) can be understood as a restricted form of the present formulation; it worked with threshold-free and constraint-free QLP($\mathscr{D}$) programs.
5. By definition, SLP has instances SLP($\mathscr{S}$) $=_{\text{def}}$ SQCLP($\mathscr{S},\mathscr{U},\mathscr{R}$). The pure fragment of Bousi~Prolog (Julián-Iranzo and Rubio-Manzano 2009) can be understood as a restricted form of SLP in the present formulation; it works with threshold-free, attenuation-free and constraint-free SLP($\mathscr{S}$) programs. Moreover, restricting the choice of $\mathscr{S}$ to similarity relations leads to SLP in the sense of Sessa (2002) and related papers.
6. The CLP scheme can be defined by instances CLP($\mathscr{C}$) $=_{\text{def}}$ SQCLP($\mathscr{S}_{\text{id}},\mathscr{B},\mathscr{C}$). Both attenuation values and threshold values are useless in CLP programs, due to the fixed parameter choice $\mathscr{D} = \mathscr{B}$.
7. Finally, the pure LP paradigm can be defined as LP $=_{\text{def}}$ SQCLP($\mathscr{S}_{\text{id}},\mathscr{B},\mathscr{H}$) where $\mathscr{H}$ is the *Herbrand* constraint domain. Again, attenuation values and threshold values are useless in LP due to the fixed parameter choice $\mathscr{D} = \mathscr{B}$.

In all the previous items, the schemes obtained by partial instantiation inherit the declarative semantics from SQCLP, using sets of observables of the form $A \sharp d \Leftarrow \Pi$ as interpretations. Similar semantic approaches were used in our previous papers (Rodríguez-Artalejo and Romero-Díaz 2008; Caballero *et al.* 2008), except that Π and equations were absent due to the lack of CLP features. The other related works discussed in the Introduction view program interpretations as mappings $\mathscr{I}$ from the ground Herbrand base into some set of lattice elements (the real interval $[0,1]$ in many cases), as already discussed in the explanations following Definition 3.1.

As seen in Subsection 3.4, SQCLP's semantics enables a declarative characterization of valid goal solutions. This fact is relevant for modeling the expected behavior of goal solving devices and reasoning about their correctness. Moreover, the relations $\approx_{\lambda,\Pi}$ introduced for the first time in the present paper (see Definition 2.3) allow to specify the semantic role of $\mathscr{S}$ in a constraint-based framework, with less technical overhead than in previous related approaches.

A related work not mentioned in items above is semiring-based CLP (Bistarelli *et al.* 2001), a scheme with instances SCLP(S) parameterized by a semiring $\mathrm{S} = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ whose elements are used to represent consistency levels in soft constraint solving. The semirings used in this approach can be equipped with a lattice structure whose *lub* operation is always $+$, but whose *glb* operation may be different from $\times$. On the other hand, our qualification domains are defined as lattices with an additional attenuation operation $\circ$. It turns out that the kind of semirings used in SCLP(S) correspond to qualification domains only in some cases. Moreover, $\times$ is

[4] Not to be confused with SCLP in the sense of (Bistarelli *et al.* 2001), discussed below.

https://doi.org/10.1017/S1471068410000323 Published online by Cambridge University Press

used in SCLP(S) to interpret logical conjunction in clause bodies and goals, while the *glb* operation is used in instances SQCLP($\mathscr{S},\mathscr{D},\mathscr{C}$) for the same purpose. For this reason, even if $\mathscr{D}$ is "equivalent" to S, SQCLP($\mathscr{S},\mathscr{D},\mathscr{C}$) cannot be naturally used to express SCLP(S) in the case that $\times$ is not the *glb*. Assuming that $\mathscr{D}$ is "equivalent" to S and that $\times$ behaves as the *glb* in S, program clauses in SCLP(S) can be viewed as a particular case of program clauses in SQCLP($\mathscr{S},\mathscr{D},\mathscr{C}$) which use an attenuation factor different from **t** only for facts. Other relevant differences between SQCLP($\mathscr{S},\mathscr{D},\mathscr{C}$) and SCLP(S) can be explained by comparing the parameters. As said before $\mathscr{D}$ may be "equivalent" to S in some cases, but $\mathscr{S}$ is absent and $\mathscr{C}$ is not made explicit in SCLP(S). Seemingly, the intended use of SCLP(S) is related to finite domain constraints and no parametrically given constraint domain is provided.

In the future we plan to implement some SQCLP instances by extending the semantically correct program transformation techniques from Caballero *et al.* (2008), and to investigate applications which can profit from flexible query answering in the line of Campi *et al.* (2009) and other related papers. Other interesting lines of future work include: a) extension of the qualified SLD resolution presented in Rodríguez-Artalejo and Romero-Díaz (2008) to a SQCLP goal solving procedure able to work with constraints and proximity relations; and b) extension of the QCFLP scheme in Caballero *et al.* (2009) to work with proximity relations and higher-order functions.

Acknowledgements

The authors are thankful to the anonymous referees for constructive remarks and suggestions which helped to improve the presentation. They are also thankful to Rafael Caballero for useful discussions on the paper's topics and to Jesús Almendros for pointing to bibliographic references in the area of flexible query answering.

References

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

Bistarelli, S., Montanari, U., and Rossi, F. 2001. Semiring-based constraint logic programming: Syntax and semantics. *ACM Transactions on Programming Languages and Systems 3,* 1 (January), 1–29.

Caballero, R., Rodríguez-Artalejo, M., and Romero-Díaz, C. A. 2008. 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*. ACM, Valencia, Spain, 185–194.

Caballero, R., Rodríguez-Artalejo, M., and Romero-Díaz, C. A. 2009. Qualified computations in functional logic programming. In *Logic Programming (ICLP'09)*, P. Hill and D. Warren, Eds. LNCS, vol. 5649. Springer, Berlin, 449–463.

Campi, A., Damiani, E., Guinea, S., Marrara, S., Pasi, G., and Spoletini, P. 2009. A fuzzy extension of the XPath query language. *Journal of Intelligent Information Systems 33,* 3 (December), 285–305.

Dubois, D. and Prade, H. 1980. *Fuzzy Sets and Systems: Theory and Applications.* Academic Press, New York.

Freuder, E. C. and Wallace, R. J. 1992. Partial constraint satisfaction. *Artificial Intelligence 58,* 1–3, 21–70.

Gabbrielli, M., Dore, G. M., and Levi, G. 1995. Observable semantics for constraint logic programs. *Journal of Logic and Computation 5,* 2, 133–171.

Georget, Y. and Codognet, P. 1998. Compiling semiring-based constraints with CLP(FD,S). In *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming*. LNCS, vol. 1520. Springer, 205–219.

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

Hanus, M. 2006. Curry: An integrated functional logic language, version 0.8.2. M. Hanus (Ed.), Available at `http://www.informatik.uni-kiel.de/~curry/report.html`.

Höhfeld, M. and Smolka, G. 1988. *Definite Relations over Constraint Languages*. Technical Report LILOG Report 53, IBM Deutschland.

Jaffar, J. and Lassez, J. L. 1987. Constraint logic programming. In *Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of Programming Languages (POPL'87)*. ACM New York, 111–119.

Jaffar, J., Michaylov, S., Stuckey, P. J., and Yap, R. H. C. 1992. The CLP(R) language and system. *ACM Transactions on Programming Languages and Systems 14(3)*, 339–395.

Julián-Iranzo, P. and Rubio-Manzano, C. 2009. A declarative semantics for Bousi∼Prolog. In *PPDP'09: Proceedings of the 11th ACM SIGPLAN conference on Principles and Practice of Declarative Programming*. ACM, Valencia, Spain, 149–160.

Kifer, M. and Subrahmanian, V. S. 1992. Theory of generalized annotated logic programs and their applications. *Journal of Logic Programming 12,* 3–4, 335–367.

Krajči, S., Lencses, R., and Vojtáš, P. 2004. A comparison of fuzzy and annonated logic programming. *Fuzzy Sets and Systems 144*, 173–192.

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

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

Ng, R. T. and Subrahmanian, V. S. 1992. Probabilistic logic programming. *Information and Computation 101,* 2, 150–201.

Riezler, S. 1998. *Probabilistic Constraint Logic Programming*. PhD thesis, Neuphilologischen Fakultät del Universität Tübingen.

Rodríguez-Artalejo, M. and Romero-Díaz, C. A. 2008. Quantitative logic programming revisited. In *Functional and Logic Programming (FLOPS'08)*, J. Garrigue and M. Hermenegildo, Eds. LNCS, vol. 4989. Springer, 272–288.

Rodríguez-Artalejo, M. and Romero-Díaz, C. A. 2010. *Fixpoint and proof-Theoretic Semantics for CLP with Qualification and Proximity*. Technical Report SIC-1-10, Universidad Complutense, Departamento de Sistemas Informáticos y Computación, Madrid, Spain. URL: `federwin.sip.ucm.es/sic/investigacion/publicaciones/informes-tecnicos`.

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

Shenoi, S. and Melton, A. 1999. Proximity relations in the fuzzy relational database model. *Fuzzy Sets and Systems 100,* supl., 51–62.

van Emden, M. H. 1986. Quantitative deduction and its fixpoint theory. *Journal of Logic Programming 3,* 1, 37–53.

Vojtáš, P. 2001. Fuzzy logic programming. *Fuzzy Sets and Systems 124*, 361–370.

Zadeh, L. A. 1971. Similarity relations and fuzzy orderings. *Information Sciences 3,* 2, 177–200.