International Journal of Approximate Reasoning 50 (2009) 837–853

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

journal homepage: www.elsevier.com/locate/ijar

Description logic programs under probabilistic uncertainty and fuzzy vagueness [☆]

Thomas Lukasiewicz [a,*], Umberto Straccia [b]

[a] Computing Laboratory, University of Oxford, Wolfson Building, Parks Road, Oxford OX1 3QD, UK
[b] ISTI-CNR, Via G. Moruzzi 1, 56124 Pisa, Italy

A R T I C L E   I N F O

Article history:
Received 15 April 2008
Received in revised form 22 February 2009
Accepted 4 March 2009
Available online 14 March 2009

Keywords:
Probabilistic logic programs
Fuzzy description logics
Fuzzy logic programs
Probabilistic uncertainty
Fuzzy vagueness
Semantic Web
Algorithms
Data tractability

A B S T R A C T

This paper is directed towards an infrastructure for handling both uncertainty and vagueness in the Rules, Logic, and Proof layers of the Semantic Web. More concretely, we present probabilistic fuzzy description logic programs, which combine fuzzy description logics, fuzzy logic programs (with stratified default-negation), and probabilistic uncertainty in a uniform framework for the Semantic Web. We define important concepts dealing with both probabilistic uncertainty and fuzzy vagueness, such as the expected truth value of a crisp sentence and the probability of a vague sentence. Furthermore, we describe a shopping agent example, which gives evidence of the usefulness of probabilistic fuzzy description logic programs in realistic Web applications. We also provide algorithms for query processing in probabilistic fuzzy description logic programs, and we delineate a special case where query processing can be done in polynomial time in the data complexity.

1. Introduction

The *Semantic Web* [1,8] aims at an extension of the current World Wide Web by standards and technologies that help machines to understand the information on the Web so that they can support richer discovery, data integration, navigation, and automation of tasks. The main ideas behind it are to add a machine-readable meaning to Web pages, to use ontologies for a precise definition of shared terms in Web resources, to use KR technology for automated reasoning from Web resources, and to apply cooperative agent technology for processing the information of the Web.

The Semantic Web consists of several hierarchical layers, where the *Ontology layer*, in form of the *OWL Web Ontology Language* [31], is currently the highest layer of sufficient maturity. OWL consists of three increasingly expressive sublanguages, namely, *OWL Lite*, *OWL DL*, and *OWL Full*. OWL Lite and OWL DL are essentially very expressive description logics with an RDF syntax. As shown in [13], ontology entailment in OWL Lite (resp., OWL DL) reduces to knowledge base (un)satisfiability in the description logic $\mathcal{SHIF}(\mathbf{D})$ (resp., $\mathcal{SHOIN}(\mathbf{D})$). On top of the Ontology layer, sophisticated representation and reasoning capabilities for the *Rules*, *Logic*, and *Proof layers* of the Semantic Web are developed next.

In particular, a key requirement of the layered architecture of the Semantic Web is to integrate the Rules and the Ontology layer. Here, it is crucial to allow for building rules on top of ontologies, that is, for rule-based systems that use vocabulary from ontology knowledge bases. Another type of combination is to build ontologies on top of rules, where ontological

[☆] This paper is a significantly extended and revised version of a paper that appeared in: *Proceedings ECSQARU-2007*, pp. 187–198. *LNCS* 4724, Springer, 2007.

[*] Corresponding author. Address: Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, 1040 Wien, Austria.
E-mail addresses: thomas.lukasiewicz@comlab.ox.ac.uk (T. Lukasiewicz), straccia@isti.cnr.it (U. Straccia).

0888-613X/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ijar.2009.03.004

definitions are supplemented by rules or imported from rules. Both types of integration have been realized in recent hybrid integrations of rules and ontologies under the loose coupling, called (*loosely coupled*) *description logic programs* (or simply *dl-programs*), which are of the form $KB = (L, P)$, where L is a description logic knowledge base, and P is a finite set of rules involving queries to L (see especially [6] and the references therein).

Other research efforts are directed towards *handling uncertainty and vagueness in the Semantic Web*, which are motivated by important Web and Semantic Web applications. In particular, formalisms for handling uncertainty are used in data integration, ontology mapping, and information retrieval, while dealing with vagueness is motivated by multimedia information processing/retrieval and natural language interfaces to the Web. There are several extensions of description logics and Web ontology languages by probabilistic uncertainty and by fuzzy vagueness. Similarly, there are also extensions of hybrid integrations of rules and ontologies by probabilistic uncertainty and by fuzzy vagueness.

Clearly, since uncertainty and vagueness are semantically quite different, it is important to have a unifying formalism for the Semantic Web, which allows for dealing with both uncertainty and vagueness. But although there has been some important work in the fuzzy logic community in this direction [9], to date there are no Semantic Web formalisms that allow for handling both uncertainty and vagueness.

In this paper, we try to fill this gap by presenting a novel approach to dl-programs, where probabilistic rules are defined on top of fuzzy rules, which are in turn defined on top of fuzzy description logics. This allows for handling both probabilistic uncertainty and fuzzy vagueness. The main contributions can be briefly summarized as follows:

- We present probabilistic fuzzy dl-programs, which combine (i) fuzzy description logics, (ii) stratified fuzzy logic programs, and (iii) stratified probabilistic logic programs in a uniform framework for the Semantic Web.[1]
- Such programs allow for handling both probabilistic uncertainty (especially for probabilistic ontology mapping and data integration) and fuzzy vagueness (especially for dealing with vague concepts). We define important concepts dealing with both probabilistic uncertainty and fuzzy vagueness, such as the expected truth value of a crisp sentence and the probability of a vague sentence.
- We describe a shopping agent example, which gives evidence of the usefulness of probabilistic fuzzy dl-programs in realistic Web applications.
- We also give algorithms for query processing in probabilistic fuzzy dl-programs, and we delineate a special case where query processing in probabilistic fuzzy dl-programs is data tractable (under suitable assumptions about the underlying fuzzy description logic), which is an important feature for the Web.

The rest of this paper is organized as follows. Section 2 gives the motivating shopping agent example. In Section 3, we recall combination strategies and fuzzy description logics. Section 4 defines fuzzy dl-programs on top of fuzzy description logics. In Section 5, we then define probabilistic fuzzy dl-programs. Section 6 provides algorithms for query processing in probabilistic fuzzy dl-programs, including data tractability results. Section 7 summarizes our main results and gives an outlook on future research.

2. Motivating example

We now describe a shopping agent example, where we encounter both probabilistic uncertainty (in resource selection, ontology mapping/query transformation, and data integration) and fuzzy vagueness (in query matching with vague concepts).

We first recall the quite general scenario of distributed information retrieval (DIR) on the Semantic Web [29] (see also to some extent [10]). The main task in it is, given a query, to access and to retrieve from distributed information resources the relevant information to a query in an effective way. Our shopping agent example will be an instance of DIR on the Semantic Web. In order to effectively cope with very large amounts of knowledge, the task of DIR in the Semantic Web may be defined in terms of three different sub-tasks. Let us assume that an agent A has to satisfy an information need Q_A expressed in a query language $\mathcal{Q}_A$, whose basic terms belong to an ontology O_A, defined using the ontology language $\mathcal{O}_A$. Let us assume also that there are a large number of ontology-based Web resources $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_n\}$ accessible to A, where each Web resource $\mathcal{S}_i$ provides access to its Web pages by having its own ontology O_i, ontology language $\mathcal{O}_i$ and query language $\mathcal{Q}_i$ (see Fig. 1). Then, the agent should perform the following three steps to satisfy its information need:

(1) **Resource selection**: The agent has to *select* a subset of some *relevant* resources $\mathcal{S}' \subseteq \mathcal{S}$, since it is not reasonable to assume that it will access and query all the resources;
(2) **Query reformulation**: For every selected resource $\mathcal{S}_i \in \mathcal{S}'$ the agent has to *re-formulate* its information need Q_A into the query language $\mathcal{L}_i$ provided by the resource;
(3) **Data fusion and rank aggregation**: The results from the selected resources have to finally be merged together.

[1] Note that the ingredients in (i) are fuzzy generalizations of the description logics behind OWL Lite and OWL DL, while the ingredients in (ii) resp. (iii) are fuzzy resp. probabilistic generalizations of stratified dl-programs following [6], which are a semantically and computationally well-behaved fragment of one of the most spread approaches to dl-programs.

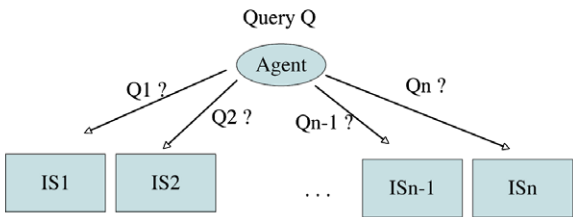

Fig. 1. Distributed information retrieval.

That is, an agent must know *where* to search, *how* to query, and *how* to combine the information and ranked lists provided back from querying different and heterogeneous resources. This is an ineffective manual task for which accurate automated tools are desired. The tasks of automated resource selection and the one of query reformulation seem to be the more problematic ones, while the data fusion and rank aggregation issue may be solved by applying directly existing techniques (see, e.g., [24]).

In DIR, both the automated resource selection and the query-reformulation tasks are fully automatic and do not require human intervention. In order to make resource selection effective and automatic, in DIR, an agent has to compute an approximation of the content of each information resource. Based on this approximation, the agent is then able to select resources and perform query reformulation effectively. The approximation is computed by relying on the so-called *query-based resource sampling methodology* (see, e.g., [5]). This method consists of computing automatically an approximation of the content of a resource, relying on a sampling technique. Roughly, it consists of a series of quasi-random queries submitted to the information resource. In the context of textual IR, it has been shown that the retrieval of a few documents is a sufficient representation of the content of information resource [5]. In automated resource selection, this approximation is then used to decide whether a resource may contain relevant information with respect to the agents' information need [4]. For ontology-based information resources, such an approximation may contain the ontology that the information resource relies on and some annotated documents (called instances) retrieved using quasi-random queries. For the query reformulation task, the agent relies on so-called transformation rules, which specify how to translate concepts and terms of the agent's vocabulary into the vocabulary of the information resource. Once the set of rules is given, the query transformation is relatively easy. What is difficult is to learn these rules automatically. In the context of the Semantic Web, these rules are essentially rules that map an entity (concept or property) in the agent's ontology into one or several entities of the information resource's ontology. Therefore, the major difficulty is in learning these *ontology mappings* automatically. Again, to do this, we may rely on the approximation computed so far through query-based sampling. The ontology of the agent, the ontology of the information resource and some annotated documents will allow the agent to learn these mappings automatically. This task is called *ontology alignment* in the Semantic Web [7,21].

Now, we turn to our shopping agent example. As we will see, it is a special case of DIR on the Semantic Web.

Example 2.1 (*Shopping agent*). Suppose a person would like to buy "a sports car that costs at most about 22 000 € and that has a power of around 150 HP". In today's Web, the buyer has to *manually* (i) search for car selling sites, e.g., using Google, (ii) select the most promising sites (e.g., http://www.autos.com), (iii) browse through them, query them to see the cars they sell, and match the cars with the requirements, (iv) select the offers in each Web site that match the requirements, and (v) eventually merge all the best offers from each site and select the best ones. Obviously, the whole process is rather *tedious* and *time consuming*, since e.g. (i) the buyer has to visit many sites, (ii) the browsing in each site is very time consuming, (iii) finding the right information in a site (which has to match the requirements) is not simple, and (iv) the way of browsing and querying may differ from site to site.

A *shopping agent* may now support the buyer as follows, *automatizing* the whole selection process once it receives the request/query q from the buyer:

- *Probabilistic resource selection*. The agent selects some sites/resources S that it considers as promising for the buyer's request. The agent has to select a subset of some *relevant* resources, since it is not reasonable to assume that it will access and query all the resources known to him. The relevance of a resource $\mathcal{S}$ to a query is usually (automatically) estimated as the probability $Pr(R|q,\mathcal{S})$ (the probability that the information resource $\mathcal{S}$ is relevant (R) to the information need q, see e.g. [4,11]). It is not difficult to see that such probabilities can be represented by probabilistic rules.
- *Probabilistic ontology mapping/query reformulation*. For the top-k selected sites, the agent has to re-formulate the buyer's query using the terminology/ontology of the specific car selling site. For this task, the agent relies on ontology mapping rules, which say how to translate a concept or property of the agent's ontology into the ontology of the information resource. To relate a concept C_B of the buyer's ontology to a concept C_S of the seller's ontology, one often automatically estimates the probability $P(C_B|C_S)$ that an instance of C_S is also an instance of C_B, which can then be represented as a probabilistic rule [28,29,21].[2]

[2] Note that there are fuzzy logic based approaches to ontology mapping as well, see e.g. [7].

- *Vague query matching*. Once the agent has translated the buyer's request for the specific site's terminology, the agent submits the query. But the buyer's request often contains many so-called *vague/fuzzy* concepts such as "the price is around 22 000 € or less", rather than strict conditions, and thus a car may *match* the buyer's condition to a *degree*. As a consequence, a site/resource/Web service may return a ranked list of cars, where the ranks depend on the degrees to which the sold items match the buyer's requests q.
- *Probabilistic rank aggregation*. Eventually, the agent has to combine the ranked lists (see e.g. [24]) by considering the involved matching (or truth) degrees (vagueness) and probability degrees (uncertainty) and show the top-n items to the buyer.

Informally, our language will allow to represent probabilities in rules to support the representation of ontology mapping and resource selection rules, and fuzziness in the description logic component to support vague query matching.

3. Preliminaries

In this section, we briefly review combination strategies and fuzzy description logics; see especially the survey [20] for further details and background.

3.1. Combination strategies

Rather than being restricted to a binary truth value among **false** and **true**, *vague propositions* may also have a truth value strictly between **false** and **true**. In the sequel, we use the unit interval $[0,1]$ as the set of all possible truth values, where 0 and 1 represent the ordinary binary truth values **false** and **true**, respectively. To combine and modify the truth values in $[0,1]$, we assume *combination strategies*, namely, *conjunction*, *disjunction*, *implication*, and *negation strategies*, denoted $\otimes, \oplus, \rhd$, and $\ominus$, respectively, which are functions $\otimes, \oplus, \rhd : [0,1] \times [0,1] \rightarrow [0,1]$ and $\ominus : [0,1] \rightarrow [0,1]$ that generalize the ordinary Boolean operators $\wedge, \vee, \rightarrow$, and $\neg$, respectively, to the set of truth values $[0,1]$. As usual, we assume that combination strategies have some natural algebraic properties. Conjunction and disjunction strategies are also called *triangular norms* (or *t-norms*) and *triangular co-norms* (or *s-norms*) [12], respectively. Combination strategies in some fuzzy logics are shown in Table 1.

3.2. Fuzzy description logics

Intuitively, description logics model a domain of interest in terms of concepts and roles, which represent classes of individuals and binary relations between classes of individuals, respectively. A knowledge base encodes in particular subset relationships between concepts, subset relationships between roles, the membership of individuals to concepts, and the membership of pairs of individuals to roles. In fuzzy description logics, these relationships and memberships then have a degree of truth in $[0,1]$.

We assume fuzzy generalizations of the crisp description logics $\mathcal{SHIF}(\mathbf{D})$ and $\mathcal{SHOIN}(\mathbf{D})$, which stand behind OWL Lite and OWL DL, respectively. We now describe the syntax and the semantics of fuzzy $\mathcal{SHIF}(\mathbf{D})$ and fuzzy $\mathcal{SHOIN}(\mathbf{D})$ (see especially [26]). For further details and background, see [20].

Syntax. We first define the syntax of fuzzy $\mathcal{SHOIN}(\mathbf{D})$. The elementary ingredients are similar to the ones of crisp $\mathcal{SHOIN}(\mathbf{D})$, except that we now also have fuzzy datatypes and fuzzy modifiers. We assume a set of *data values*, a set of *elementary datatypes*, and a set of *datatype predicates* (each with a predefined arity $n \geqslant 1$). A *datatype* is an elementary datatype or a finite set of data values. A *fuzzy datatype theory* $\mathbf{D} = (\Delta^{\mathbf{D}}, \cdot^{\mathbf{D}})$ consists of a datatype domain $\Delta^{\mathbf{D}}$ and a mapping $\cdot^{\mathbf{D}}$ that assigns to each data value an element of $\Delta^{\mathbf{D}}$, to each elementary datatype a subset of $\Delta^{\mathbf{D}}$, and to each datatype predicate of arity n a fuzzy relation over $\Delta^{\mathbf{D}}$ of arity n (that is, a mapping $(\Delta^{\mathbf{D}})^n \rightarrow [0,1]$). We extend $\cdot^{\mathbf{D}}$ to all datatypes by $\{v_1, \ldots, v_n\}^{\mathbf{D}} = \{v_1^{\mathbf{D}}, \ldots, v_n^{\mathbf{D}}\}$. Non-crisp predicates are usually defined by functions for specifying fuzzy set membership degrees, such as the trapezoidal, triangular, left-shoulder, and right-shoulder functions (see Fig. 2).

Let us consider pairwise disjoint alphabets of *atomic concepts*, *abstract roles*, *datatype roles*, *individuals*, and *fuzzy modifiers*, respectively.

Table 1
Combination strategies in some fuzzy logics.

	Łukasiewicz logic	Gödel logic	Product logic	"Zadeh logic"
$a \otimes b$	$\max(a+b-1, 0)$	$\min(a,b)$	$a \cdot b$	$\min(a,b)$
$a \oplus b$	$\min(a+b, 1)$	$\max(a,b)$	$a+b-a \cdot b$	$\max(a,b)$
$a \rhd b$	$\min(1-a+b, 1)$	$\begin{cases} 1 & \text{if } a \leqslant b \\ b & \text{otherwise} \end{cases}$	$\min(1, b/a)$	$\max(1-a, b)$
$\ominus a$	$1-a$	$\begin{cases} 1 & \text{if } a = 0 \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} 1 & \text{if } a = 0 \\ 0 & \text{otherwise} \end{cases}$	$1-a$

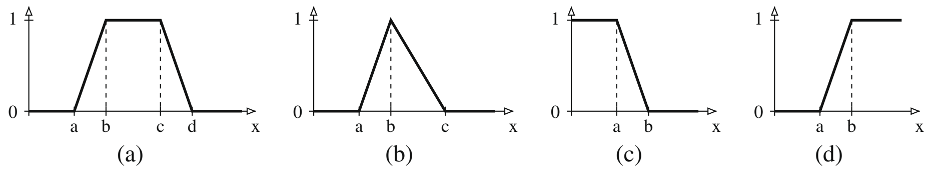

Fig. 2. (a) Trapezoidal function $trz(x;a,b,c,d)$, (b) triangular function $tri(x;a,b,c)$, (c) left-shoulder function $ls(x;a,b)$, and (d) right-shoulder function $rs(x;a,b)$.

Roles and concepts in fuzzy $\mathcal{SHOIN}(\mathbf{D})$ are defined in nearly the same way as concepts in $\mathcal{SHOIN}(\mathbf{D})$, except that we now also allow fuzzy modifiers as unary operators on concepts. A *role R* is either an abstract role, a datatype role, or the inverse of an abstract role (denoted R^-). We define *concepts* inductively as follows. Each atomic concept A is a concept, $\bot$ and $\top$ are concepts, and if $a_1,\ldots,a_n$ are individuals, then $\{a_1,\ldots,a_n\}$ is a concept (called *oneOf*). If C, C_1, C_2 are concepts, R, S are abstract roles, and m is a modifier, then $(C_1 \sqcap C_2), (C_1 \sqcup C_2), \neg C$, and $m(C)$ are concepts (called *conjunction*, *disjunction*, *negation*, and *fuzzy modification*, respectively), as well as $\exists R.C, \forall R.C, \geqslant nS$, and $\leqslant nS$ (called *existential*, *value*, *atleast*, and *atmost restriction*, respectively) for an integer $n \geqslant 0$. If D is a datatype and $T, T_1,\ldots,T_n$ are datatype roles, then $\exists T_1,\ldots,T_n.D, \forall T_1,\ldots,T_n.D, \geqslant nT$, and $\leqslant nT$ are concepts (called *datatype existential*, *value*, *atleast*, and *atmost restriction*, respectively) for an integer $n \geqslant 0$. We eliminate parentheses as usual.

Example 3.1. Examples of concepts expressions are shown below.

$$Cars \sqcup Trucks \sqcup Vans \sqcup SUVs, \tag{1}$$
$$PassengerCars \sqcap LuxuryCars, \tag{2}$$
$$Vehicles \sqcap \exists hasHP.{>_{200}}, \tag{3}$$
$$Cars \sqcap \forall hasOwner.Italian, \tag{4}$$
$$Cars \sqcap \exists hasInvoice.ls(x;22000,25000). \tag{5}$$

Concept (1) denotes the union of the sets of cars, trucks, vans, and SUVs, while concept (2) denotes the intersection of the sets of passenger cars and luxury cars, i.e., all luxury passenger cars. Concept (3) denotes the set of vehicles with more than 200 HP. Here, *hasHP* is a datatype role, and $>_{200}$ is a datatype predicate denoting $\{n \mid n > 200\}$. Concept (4) denotes the set of cars with only Italian owners, while concept (5) denotes the set of cars costing "at most about 22 000 €". Here, *hasInvoice* is a datatype role, and $ls(x;22000,25000)$ is a fuzzy datatype representing "at most about 22 000 €".

A *crisp axiom* has one of the following forms: (1) $C \sqsubseteq D$ (called *concept inclusion axiom*), where C and D are concepts[3]; (2) $R \sqsubseteq S$ (called *role inclusion axiom*), where either both R, S are datatype roles or both R, S are abstract roles; (3) Trans(R) (called *transitivity axiom*), where R is an abstract role; (4) $C(a)$ (called *concept assertion axiom*), where C is a concept, and a is an individual; (5) $R(a,b)$ (resp., $U(a,v)$) (called *role assertion axiom*), where R is an abstract role (resp., U is a datatype role), and a, b are individuals (resp., a is an individual, and v is a data value); and (6) $a = b$ (resp., $a \neq b$) (*equality* (resp., *inequality*) *axiom*), where a, b are individuals.[4]

Example 3.2. Examples of crisp axioms are shown below.

$$PassengerCars \sqcup LuxuryCars \sqsubseteq Cars, \tag{6}$$
$$\mathrm{Trans}(hasPart), \tag{7}$$
$$(PassengerCars \sqcap \exists hasHP.{=_{75}})(fiat500), \tag{8}$$
$$hasOwner(fiat500, rossi), \tag{9}$$
$$hasInvoice(fiat500, 9500). \tag{10}$$

Axiom (6) is a crisp concept inclusion axiom stating that passenger and luxury cars are cars; (7) is a transitivity axiom stating that the role *hasPart* is transitive; (8) is a concept assertion axiom stating that *fiat500* is a passenger car with 75 HP ($=_{75}$ is a datatype predicate denoting $\{75\}$); (9) is a role assertion axiom stating that *fiat500* has the owner *rossi*; and (10) is a role assertion axiom stating that *fiat500* costs 9500 €.

We define *fuzzy axioms* as follows. A *fuzzy concept inclusion* (resp., *fuzzy role inclusion*, *fuzzy concept assertion*, *fuzzy role assertion*) *axiom* is of the form $\alpha\ \theta\ n$, where α is a concept inclusion (resp., role inclusion, concept assertion, role assertion) axiom, $\theta \in \{\leqslant, =, \geqslant\}$, and $n \in [0,1]$. Informally, $\alpha \leqslant n$ (resp., $\alpha = n, \alpha \geqslant n$) encodes that the truth value of α is at most (resp., equal to, at least) n. We often use α to abbreviate $\alpha = 1$.

[3] Note that concept inclusion axioms $C \sqsubseteq D$ involve fully general concepts C and D.

[4] Note that the equality (resp., inequality) in equality (resp., inequality) axioms is crisp.

Example 3.3. Examples of fuzzy axioms are shown below.

$$\mathit{Cars} \sqcap \exists \mathit{hasSpeed}.\mathit{HighSpeed} \sqsubseteq \mathit{SportCars} \geqslant 0.8, \tag{11}$$

$$\mathit{SportCars}(\mathit{audiTT}) \geqslant 0.9. \tag{12}$$

Axiom (11) is a fuzzy concept inclusion axiom stating that a car whose speed is high is to some extent a sports car; and (8) is a fuzzy concept assertion axiom stating that *audiTT* is to some extent a sports car.

A *fuzzy* (*description logic*) *knowledge base* L is a finite set of fuzzy axioms, transitivity axioms, and equality and inequality axioms. For decidability, number restrictions in L are restricted to simple abstract roles. Notice that L may contain fuzzy concept inclusion axioms (between general concepts).

Fuzzy $\mathcal{SHIF}(\mathbf{D})$ has the same syntax as fuzzy $\mathcal{SHOIN}(\mathbf{D})$, but without the oneOf constructor and with the atleast and atmost constructors limited to 0 and 1.

Example 3.4 (*Shopping agent cont'd*). A fuzzy description logic knowledge base L encoding the site in Example 2.1 may contain in particular the following axioms:

$$\mathit{Cars} \sqcup \mathit{Trucks} \sqcup \mathit{Vans} \sqcup \mathit{SUVs} \sqsubseteq \mathit{Vehicles}, \tag{13}$$

$$\mathit{PassengerCars} \sqcup \mathit{LuxuryCars} \sqsubseteq \mathit{Cars}, \tag{14}$$

$$\mathit{CompactCars} \sqcup \mathit{MidSizeCars} \sqcup \mathit{SportsCars} \sqsubseteq \mathit{PassengerCars}, \tag{15}$$

$$\begin{aligned}\mathit{Cars} \sqsubseteq\ & (\exists \mathit{hasReview}.\mathbf{Integer}) \sqcap (\exists \mathit{hasInvoice}.\mathbf{Integer}) \\ & \sqcap (\exists \mathit{hasHP}.\mathbf{Integer}) \sqcap (\exists \mathit{hasResellValue}.\mathbf{Integer}) \\ & \sqcap (\exists \mathit{hasSafetyFeatures}.\mathbf{Integer}) \sqcap \cdots,\end{aligned} \tag{16}$$

$$(\mathit{SportsCar} \sqcap (\exists \mathit{hasInvoice}.\{18883\}) \sqcap (\exists \mathit{hasHP}.\{166\}) \sqcap \cdots)(\mathit{mazdaMX5Miata}), \tag{17}$$

$$(\mathit{SportsCar} \sqcap (\exists \mathit{hasInvoice}.\{20341\}) \sqcap (\exists \mathit{hasHP}.\{200\}) \sqcap \cdots)(\mathit{volkswagenGTI}), \tag{18}$$

$$(\mathit{SportsCar} \sqcap (\exists \mathit{hasInvoice}.\{24029\}) \sqcap (\exists \mathit{hasHP}.\{162\}) \sqcap \cdots)(\mathit{mitsubishiES}). \tag{19}$$

Here, axioms (13)–(15) describe the concept taxonomy of the Web site, while axiom (16) describes the datatype attributes of the cars sold in the site. For example, every passenger or luxury car is also a car, and every car has a resell value. Axioms (17)–(19) describe the properties of some sold cars. For example, the *mazdaMX5Miata* is a sports car, costing 18 883 €. Note that **Integer** is the datatype of all integers. We may now encode "costs at most about 22 000 €" and "has a power of around 150 HP" in the buyer's request through the following concepts C resp. D:

$$C = \exists \mathit{hasInvoice}.\mathit{LeqAbout}22000 \quad \text{and} \quad D = \exists \mathit{hasHP}.\mathit{Around}150\mathit{HP},$$

where $\mathit{LeqAbout}22000(x) = \mathit{ls}(x; 22000, 25000)$ and $\mathit{Around}150\mathit{HP}(x) = \mathit{tri}(x; 125, 150, 175)$ (see Fig. 2), respectively. The latter two equations define the fuzzy concepts "at most about 22 000 €" and "around 150 HP", respectively. The former is modeled as a left-shoulder function stating that if the price is less than 22 000 €, then the degree of truth (degree of buyer's satisfaction) is 1, else the degree of truth is linearly decreasing to 0 (reached at the cost of 25 000 €). In fact, we are modeling a case were the buyer would like to pay less than 22 000 €, though may still accept a higher price (up to 25 000 €) to a lesser degree. Similarly, the latter models the fuzzy concept "around 150 HP" as a triangular function with vertice in 150 HP.

Semantics. We now define the semantics of fuzzy $\mathcal{SHIF}(\mathbf{D})$ and fuzzy $\mathcal{SHOIN}(\mathbf{D})$. The main idea behind it is that concepts and roles are interpreted as fuzzy subsets of an interpretation's domain. Therefore, rather than being satisfied (true) or unsatisfied (false) in an interpretation, axioms are associated with a degree of truth in $[0, 1]$. In the following, let $\otimes, \oplus, \rhd$, and $\ominus$ be arbitrary but fixed conjunction, disjunction, implication, and negation strategies (see Table 1), respectively.

A *fuzzy interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ relative to a fuzzy datatype theory $\mathbf{D} = (\Delta^{\mathbf{D}}, \cdot^{\mathbf{D}})$ consists of a nonempty set $\Delta^{\mathcal{I}}$ (called the *domain*), disjoint from $\Delta^{\mathbf{D}}$, and of a *fuzzy interpretation function* $\cdot^{\mathcal{I}}$ that coincides with $\cdot^{\mathbf{D}}$ on every data value, datatype, and fuzzy datatype predicate, and it assigns

- to each individual a an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$;
- to each atomic concept A a function $A^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0, 1]$;
- to each abstract role R a function $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0, 1]$;
- to each datatype role T a function $T^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathbf{D}} \to [0, 1]$;
- to each fuzzy modifier m the modifier function $m^{\mathcal{I}} = f_{\mathrm{m}} : [0, 1] \to [0, 1]$.

The mapping $\cdot^{\mathcal{I}}$ is extended to all roles and concepts as follows (where $x, y \in \Delta^{\mathcal{I}}$):

$$(R^-)^{\mathcal{I}}(x, y) = R^{\mathcal{I}}(y, x);$$

$$\top^{\mathcal{I}}(x) = 1;$$

$$\bot^{\mathcal{I}}(x) = 0;$$

$$\{a_1, \ldots, a_n\}^{\mathcal{I}}(x) = \begin{cases} 1 & \text{if } x \in \{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}; \\ 0 & \text{otherwise;} \end{cases}$$

$$(C_1 \sqcap C_2)^{\mathcal{I}}(x) = C_1^{\mathcal{I}}(x) \otimes C_2^{\mathcal{I}}(x);$$
$$(C_1 \sqcup C_2)^{\mathcal{I}}(x) = C_1^{\mathcal{I}}(x) \oplus C_2^{\mathcal{I}}(x);$$
$$(\neg C)^{\mathcal{I}}(x) = \ominus C^{\mathcal{I}}(x);$$
$$(m(C))^{\mathcal{I}}(x) = m^{\mathcal{I}}(C^{\mathcal{I}}(x));$$
$$(\exists R.C)^{\mathcal{I}}(x) = \sup_{y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \otimes C^{\mathcal{I}}(y);$$
$$(\forall R.C)^{\mathcal{I}}(x) = \inf_{y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \rhd C^{\mathcal{I}}(y);$$
$$(\geqslant n\ R)^{\mathcal{I}}(x) = \sup_{y_1,\ldots,y_n \in \Delta^{\mathcal{I}}, |\{y_1,\ldots,y_n\}|=n} \bigotimes_{i=1}^{n} R^{\mathcal{I}}(x,y_i);$$
$$(\leqslant n\ R)^{\mathcal{I}}(x) = \inf_{y_1,\ldots,y_{n+1} \in \Delta^{\mathcal{I}}, |\{y_1,\ldots,y_{n+1}\}|=n+1} \left(\bigotimes_{i=1}^{n+1} R^{\mathcal{I}}(x,y_i) \right) \rhd 0;$$
$$(\exists T_1,\ldots,T_n.D)^{\mathcal{I}}(x) = \sup_{y_1,\ldots,y_n \in \Delta^{\mathbf{D}}} \left(\bigotimes_{i=1}^{n} T_i^{\mathcal{I}}(x,y_i) \right) \otimes D^{\mathbf{D}}(y_1,\ldots,y_n);$$
$$(\forall T_1,\ldots,T_n.D)^{\mathcal{I}}(x) = \inf_{y_1,\ldots,y_n \in \Delta^{\mathbf{D}}} \left(\bigotimes_{i=1}^{n} T_i^{\mathcal{I}}(x,y_i) \right) \rhd D^{\mathbf{D}}(y_1,\ldots,y_n).$$

The mapping $\cdot^{\mathcal{I}}$ is extended to concept inclusion, role inclusion, concept assertion, and role assertion axioms as follows (where a, b are individuals, and v is a data value):

$$(C \sqsubseteq D)^{\mathcal{I}} = \inf_{x \in \Delta^{\mathcal{I}}} C^{\mathcal{I}}(x) \rhd D^{\mathcal{I}}(x);$$
$$(R \sqsubseteq S)^{\mathcal{I}} = \inf_{x,y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \rhd S^{\mathcal{I}}(x,y);$$
$$(T \sqsubseteq U)^{\mathcal{I}} = \inf_{(x,y) \in \Delta^{\mathcal{I}} \times \Delta^{\mathbf{D}}} T^{\mathcal{I}}(x,y) \rhd U^{\mathcal{I}}(x,y);$$
$$(a : C)^{\mathcal{I}} = C^{\mathcal{I}}(a^{\mathcal{I}});$$
$$((a,b) : R)^{\mathcal{I}} = R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}});$$
$$((a,v) : T)^{\mathcal{I}} = T^{\mathcal{I}}(a^{\mathcal{I}}, v^{\mathbf{D}}).$$

The notion of a fuzzy interpretation $\mathcal{I}$ *satisfying* a transitivity, equality, inequality, or fuzzy axiom E, or $\mathcal{I}$ being a *model* of E, denoted $\mathcal{I} \models E$, is defined as follows: (i) $\mathcal{I} \models \mathrm{Trans}(R)$ iff $R^{\mathcal{I}}(x,y) \geqslant \sup_{z \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,z) \otimes R^{\mathcal{I}}(z,y)$ for all $x, y \in \Delta^{\mathcal{I}}$; (ii) $\mathcal{I} \models a = b$ iff $a^{\mathcal{I}} = b^{\mathcal{I}}$, and $\mathcal{I} \models a \neq b$ iff $a^{\mathcal{I}} \neq b^{\mathcal{I}}$; and (iii) $\mathcal{I} \models \alpha\ \theta\ n$ iff $\alpha^{\mathcal{I}}\ \theta\ n$. A concept C is *satisfiable* iff there exists an interpretation $\mathcal{I}$ and an individual $x \in \Delta^{\mathcal{I}}$ such that $C^{\mathcal{I}}(x) > 0$. We say $\mathcal{I}$ *satisfies* a set of fuzzy axioms $\mathcal{E}$, or $\mathcal{I}$ is a *model* of $\mathcal{E}$, denoted $\mathcal{I} \models \mathcal{E}$, iff $\mathcal{I}$ satisfies every $E \in \mathcal{E}$. We say $\mathcal{I}$ *satisfies* a fuzzy description logic knowledge base $KB = (\mathcal{T}, \mathcal{R}, \mathcal{A})$, or $\mathcal{I}$ is a *model* of KB, denoted $\mathcal{I} \models KB$, iff $\mathcal{I}$ is a model of $\mathcal{T} \cup \mathcal{R} \cup \mathcal{A}$. We say KB is *satisfiable* iff there exists a model of KB. A fuzzy axiom E is a *logical consequence* of KB, denoted $KB \models E$, iff every model of KB satisfies E.

Example 3.5 (*Shopping agent cont'd*). The following fuzzy axioms are logical consequences of the fuzzy description logic knowledge base L in Example 3.4 (using Gödel t-norm and s-norm, and $x \rhd y = \max(1-x, y)$):

$$C(\mathit{mazdaMX5Miata}) = 1.0, \quad D(\mathit{mazdaMX5Miata}) = 0.36,$$
$$C(\mathit{volkswagenGTI}) = 1.0, \quad D(\mathit{volkswagenGTI}) = 0.0,$$
$$C(\mathit{mitsubishiES}) = 0.32, \quad D(\mathit{mitsubishiES}) = 0.52.$$

For example, $KB \models C(\mathit{mitsubishiES}) = 0.32$ can be verified as follows. By axiom (19), *mitsubishiES* is a sports car, whose cost is 24029 €, and thus $KB \models \mathit{hasInvoice}(\mathit{mitsubishiES}, 24029) = 1.0$. Then, from the definition $\mathit{LeqAbout22000}(x) = \mathit{ls}(x; 22000, 25000)$, from $\mathit{ls}(24029; 22000, 25000) = 0.32$, and from $x \otimes y = \min(x,y)$, it follows that $KB \models \exists \mathit{hasInvoice}.\mathit{LeqAbout22000}(\mathit{mitsubishiES}) = 0.32$.

4. Fuzzy description logic programs

In this section, we define *fuzzy description logic programs* (or *fuzzy dl-programs*), which are a combination of normal fuzzy programs under their answer set semantics with fuzzy description logic knowledge bases under their fuzzy first-order semantics (which are loosely coupled via a fuzzy generalization of the dl-query interfacing technique of [6]).

The fuzzy dl-programs here are similar to the ones in [14], except that they are based on fuzzy description logics as in [26], and that we consider only stratified fuzzy dl-programs here. Their canonical model associates with every ground atom a truth value, and so defines a ranking on the Herbrand base. We first introduce the syntax and models of (normal) fuzzy dl-programs, and we then define the semantics of positive and stratified fuzzy dl-programs in terms of a least model and an iterative least model semantics, respectively.

4.1. Syntax of fuzzy dl-programs

A normal fuzzy program is a finite collection of normal fuzzy rules, which are similar to ordinary normal rules, except that (i) they have a lower bound for their truth value, and (ii) they refer to fuzzy interpretations rather than binary interpretations, and thus every of their logical operators (that is, "$\leftarrow$", "$\wedge$", and "*not*") is associated with a combination strategy (that is, "$\leftarrow$" and "$\wedge$" are associated with a conjunction strategy $\otimes$, while "*not*" is associated with a negation strategy $\ominus$) to specify how the logical operators combine truth values. Informally, an example of a crisp rule according to [6] is

$$NiceSportsCar(x) \leftarrow madeIn(x,y), Italy(y), DL[SportsCar](x).$$

This rule states that a sports car that is made in Italy is nice. In this rule, the construct $DL[SportsCar](x)$ acts as a query to the description logic theorem prover, asking that x is provably a sports car from the description logic component of the knowledge base. Instead, in a fuzzy variant of the rule above, e.g.,

$$NiceSportsCar(x) \leftarrow_{\otimes}^{0.8} madeIn(x,y) \wedge_{\otimes} Italy(y) \wedge_{\otimes} DL[SportsCar](x),$$

we allow to use fuzzy conjunctions as combination strategies. Informally, the reading of this rule is as follows. For any given x, the degree of truth of the head of the rule, i.e., $NiceSportCar(x)$, is at least the degree of the evaluation of the body of the rule (right-hand side of $\leftarrow_{\otimes}$). The degree of the body is the conjunction of its components together with 0.8 (hence, 0.8 acts as a kind of threshold of the rule). The degree of truth n of $DL[SportsCar](x)$ is determined by a query to the fuzzy description logic theorem prover, requiring that $SportsCar(x) \geqslant n$ is entailed by the fuzzy description logic component.

Formally, we assume a function-free first-order vocabulary Φ with finite nonempty sets of constant and predicate symbols, and a set of variables. We use Φ_c to denote the set of all constant symbols in Φ. A *term* is a constant symbol from Φ or a variable. If p is a predicate symbol of arity $k \geqslant 0$ from Φ, and $t_1, \ldots, t_k$ are terms, then $p(t_1, \ldots, t_k)$ is an *atom*. A *literal* is an atom a or a default-negated atom *not* a. A *normal fuzzy rule* r has the form[5]

$$a \leftarrow_{\otimes_0}^{v} b_1 \wedge_{\otimes_1} b_2 \wedge_{\otimes_2} \cdots \wedge_{\otimes_{k-1}} b_k \wedge_{\otimes_k} not_{\ominus_{k+1}} b_{k+1} \wedge_{\otimes_{k+1}} \cdots \wedge_{\otimes_{m-1}} not_{\ominus_m} b_m, \tag{20}$$

where $m \geqslant k \geqslant 0, a, b_1, b_2, \ldots, b_k, b_{k+1}, \ldots, b_m$ are atoms, $\otimes_0, \ldots, \otimes_{m-1}$ are conjunction strategies, $\ominus_{k+1}, \ldots, \ominus_m$ are negation strategies, and $v \in [0,1]$. We call a the *head* of r, denoted $H(r)$, while the conjunction $b_1 \wedge_{\otimes_1} \cdots \wedge_{\otimes_{m-1}} not_{\ominus_m} b_m$ is the *body* of r. We define $B(r) = B^+(r) \cup B^-(r)$, where $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$. We call the normal fuzzy rule r a *fuzzy fact* iff $m = 0$. Note that a fuzzy fact $a \leftarrow_{\otimes}^{v}$ will establish that a is true to degree at least v. A *normal fuzzy program P* is a finite set of normal fuzzy rules.

We next allow queries to the description logic theorem prover to occur in rules. A (normal) fuzzy dl-program consists of a fuzzy description logic knowledge base L and a generalized normal fuzzy program P, which may contain queries to L in rule bodies. In such a query (called *dl-query*), it is asked for the tight truth value with which a certain concept or role assertion follows from L.

Formally, as in Section 3, we assume pairwise disjoint sets of atomic concepts, abstract roles, datatype roles, individuals, and fuzzy modifiers, respectively, and data values. We also assume Φ and Φ_c as above, where (i) Φ_c is a subset of individuals and data values (since the constants in Φ_c may occur in concept and role assertions of dl-queries), and (ii) Φ does not contain any atomic concept or role symbols (and thus dl-queries are the only interface between P and L). A *dl-query* $Q(\mathbf{t})$ is either (a) of the form $C(t)$, where C is a concept, and t is a term, or (b) of the form $R(t_1, t_2)$, where R is a role, and t_1 and t_2 are terms. A *dl-atom* has the form

$$DL[S_1 \uplus p_1, \ldots, S_m \uplus p_m; Q](\mathbf{t}),$$

where each S_i is an atomic concept or a role, p_i is a unary resp. binary predicate symbol, $Q(\mathbf{t})$ is a dl-query, and $m \geqslant 0$. We call $p_1, \ldots, p_m$ its *input predicate symbols*. The informal meaning of a dl-atom $DL[S_1 \uplus p_1, \ldots, S_m \uplus p_m; Q](t)$ is that we ask about the truth degree of $Q(t)$ with respect to the description logic component L in which every $S_i(e)$ is at least the truth value of $p_i(e)$, where $\mathbf{e}$ is a constant (resp., pair of constants) from Φ when S_i is a concept (resp., role) (and thus p_i is a unary (resp., binary) predicate symbol). Essentially, $S_i \uplus p_i$ is used to pass additional conditions to the fuzzy description logic theorem prover in order to compute the degree of truth of $Q(t)$ with respect to L. So, for example,

$$DL[builtIn \uplus madeIn; SportsCar](x)$$

is a query to the description logic component L asking about the truth of $SportsCar(x)$, where L is augmented with the additional constraints $builtIn(e_1, e_2) \geqslant madeIn(e_1, e_2)$ (where e_1 and e_2 are constants from Φ). The statement $builtIn \uplus madeIn$ thus allows to pass the information about *madeIn* derived from the rule component to the role *builtIn* dealt within the description logic component.

A *fuzzy dl-rule r* is of the form (20), where any b_i in the body of r may be a dl-atom. A fuzzy dl-rule r with empty body is a *fuzzy fact*. A (*normal*) *fuzzy dl-program KB* $= (L, P)$ consists of a satisfiable fuzzy description logic knowledge base L and a finite

[5] Note that "$\leftarrow_{\otimes_0}$", "$\wedge_{\otimes_i}$", and "$not_{\ominus_j}$" denote the annotations of the logical operators "$\leftarrow$", "$\wedge$", and "*not*" with the combination strategies $\otimes_0, \otimes_i$, and $\ominus_j$, respectively.

set of fuzzy dl-rules P. *Substitutions*, *ground substitutions*, *ground terms*, *ground atoms*, etc., are defined as usual. We denote by $ground(P)$ the set of all ground instances of fuzzy dl-rules in P relative to Φ.

Example 4.1 (*Shopping agent cont'd*). A fuzzy dl-program $KB = (L, P)$ is given by the fuzzy description logic knowledge base L in Example 3.4, and the set of fuzzy dl-rules P, which contains only the following fuzzy dl-rule encoding the buyer's request, where $\otimes$ is, e.g., given by $x \otimes y = \min(x, y)$ for all $x, y \in [0,1]$:

$$query(x) \leftarrow^{1}_{\otimes} DL[SportsCar](x) \wedge_{\otimes} DL[\exists hasInvoice.LeqAbout22000](x) \wedge_{\otimes} DL[\exists hasHP.Around150HP](x).$$

Essentially, we are asking about sports cars with a price of about 22 000 € and a power of around 150 HP. The degree of truth of $query(x)$ will be the conjunction of the degrees of truth of the three dl-atoms in the rule body. Note that the specific combination strategies to be used in a fuzzy dl-program depend on the concrete application at hand.

4.2. Models of fuzzy dl-programs

We now define (Herbrand) models of (normal) fuzzy dl-programs. We first define fuzzy (Herbrand) interpretations, the semantics of dl-queries, and the truth of fuzzy dl-rules and of fuzzy dl-programs in fuzzy interpretations. The latter is done by defining the truth value of ground dl-atoms in fuzzy interpretations. In the sequel, let $KB = (L, P)$ be a (normal) fuzzy dl-program.

We denote by HB (resp., HU) the Herbrand base (resp., universe) over Φ. In the sequel, we assume that HB is nonempty. A *fuzzy interpretation* I is a mapping $I : HB \to [0,1]$. We denote by $I_\top$ the fuzzy interpretation I such that $I(a) = 1$ for all $a \in HB$, and by $I_\bot$ the fuzzy interpretation I such that $I(a) = 0$ for all $a \in HB$. For fuzzy interpretations I and J, we write $I \preceq J$ iff $I(a) \leqslant J(a)$ for all $a \in HB$, and we define the *meet* (resp., *join*) of I and J, denoted $I \wedge J$ (resp., $I \vee J$), by $(I \wedge J)(a) = \min(I(a), J(a))$ (resp., $(I \vee J)(a) = \max(I(a), J(a))$) for all $a \in HB$. The truth value of $a \in HB$ in I under L, denoted $I_L(a)$, is defined as $I(a)$. The truth value of a ground dl-atom $a = DL[S_1 \uplus p_1, \ldots, S_m \uplus p_m; Q](c)$ in I under L, denoted $I_L(a)$, is defined as

$$I_L(a) = \sup\left\{ v \middle| L \cup \bigcup_{i=1}^{m} A_i(I) \models Q(c) \geqslant v \right\}, \quad \text{where}$$
$$A_i(I) = \{S_i(e) \geqslant I(p_i(e)) \mid I(p_i(e)) > 0,\ p_i(e) \in HB\}.$$

We shortly explain how $I_L(a)$ is determined. Essentially, we ask about the degree of truth v of $Q(c)$ with respect to the description logic component L, which has been augmented with the additional information $A_i(I)$. This latter consists of a set of fuzzy assertion axioms of the form $S_i(e) \geqslant I(p_i(e))$ (the truth of $S_i(e)$ is at least the value of $I(p_i(e))$) allowing to transfer the information from the rule component P to the description logic component L (recall that p_i is a predicate of the rule language, while S_i is a concept or role). Now, we say I is a *model* of a ground fuzzy dl-rule r of form (20) under L, denoted $I \models_L r$, iff

$$I_L(a) \geqslant \begin{cases} I_L(b_1) \otimes_1 I_L(b_2) \otimes_2 \cdots \otimes_{k-1} I_L(b_k) \otimes_k & \text{if } m \geqslant 1; \\ \quad \ominus_{k+1} I_L(b_{k+1}) \otimes_{k+1} \cdots \otimes_{m-1} \ominus_m I_L(b_m) \otimes_0 v & \\ v & \text{otherwise.} \end{cases}$$

Here, we assume that $\otimes_1, \ldots, \otimes_{m-1}, \otimes_0$ are evaluated from left to right. Note that for a fuzzy fact $a \leftarrow^{v}_{\otimes}$, we have $I_L(a) \geqslant v$ as anticipated, while if $m \geqslant 1$ then the truth value v acts as a threshold. We say I is a *model* of $KB = (L, P)$, denoted $I \models KB$, iff $I \models_L r$ for all $r \in ground(P)$.

4.3. Positive fuzzy dl-programs

Positive fuzzy dl-programs are fuzzy dl-programs without default negation. Like ordinary positive programs and positive dl-programs, they are always satisfiable and have a unique least model, which defines their canonical semantics.

Formally, a fuzzy dl-program $KB = (L, P)$ is *positive* iff P is "*not*"-free. For ordinary positive programs KB, as well as positive dl-programs (without classical negation) KB, the meet of a set of models of KB is also a model of KB, i.e., if I, J are models of KB, so is $I \wedge J$. A similar result holds for positive fuzzy dl-programs KB. Hence, every positive fuzzy dl-program KB has as its *canonical model* a unique least model, denoted M_{KB}.

Example 4.2 (*Shopping agent cont'd*). The fuzzy dl-program $KB = (L, P)$ of Example 4.1 is positive, and its minimal model M_{KB} is given as follows:

$$M_{KB}(query(mazdaMX5Miata)) = 0.36, \quad M_{KB}(query(mitsubishiES)) = 0.32,$$

and all other ground instances of $query(x)$ have the truth value 0 under M_{KB}.

4.4. Stratified fuzzy dl-programs

We next consider default-negation. The semantics assumes that fuzzy dl-programs are *stratified*. Stratified fuzzy dl-programs are composed of hierarchic layers of positive fuzzy dl-programs that are linked via default-negation.

For any fuzzy dl-program $KB = (L, P)$, let DL_P denote the set of all ground dl-atoms that occur in $ground(P)$. An *input atom* of $a \in DL_P$ is a ground atom with an input predicate of a and constant symbols in Φ. The notion of a stratification for fuzzy dl-programs defines an ordered partition of the set of all ground atoms and ground dl-atoms as follows. A *stratification of* $KB = (L, P)$ (*with respect to* DL_P) is a mapping $\lambda : HB \cup DL_P \to \{0, 1, \ldots, k\}$ such that (i) to (iii) hold:

(i) $\lambda(H(r)) \geqslant \lambda(a)$ for each $r \in ground(P)$ and $a \in B^+(r)$;
(ii) $\lambda(H(r)) > \lambda(a)$ for each $r \in ground(P)$ and $a \in B^-(r)$); and
(iii) $\lambda(a) \geqslant \lambda(a')$ for each input atom a' of each $a \in DL_P$,

where $k \geqslant 0$ is the *length* of λ. A fuzzy dl-program $KB = (L, P)$ is *stratified* iff it has a stratification λ of some length $k \geqslant 0$. For $i \in \{0, \ldots, k\}$, we define $KB_i = (L, P_i)$ where $P_i = \{r \in ground(P) \mid \lambda(H(r)) = i\}$, and we define HB_i (resp., $HB_i^\star$) as the set of all $a \in HB$ such that $\lambda(a) = i$ (resp., $\lambda(a) \leqslant i$).

Example 4.3. Consider $KB = (\emptyset, P)$, where the rule component P is

$$\begin{array}{lll} a \leftarrow_{\otimes}^{0.8} & c \leftarrow_{\otimes}^{0.6} a & d \leftarrow_{\otimes}^{0.9} not_{\ominus} c \\ b \leftarrow_{\otimes}^{0.7} & c \leftarrow_{\otimes}^{0.9} not_{\ominus} b & d \leftarrow_{\otimes}^{0.9} not_{\ominus} a. \end{array}$$

where $\otimes$ is the Gödel t-norm and $\ominus$ is the Łukasiewicz negation. Then, a stratification is $\lambda(a) = \lambda(b) = 0, \lambda(c) = 1$, and $\lambda(d) = 2$. Furthermore, we have that

$$\begin{array}{l} P_0 = \{a \leftarrow_{\otimes}^{0.8}, b \leftarrow_{\otimes}^{0.7}\}, \quad P_1 = \{c \leftarrow_{\otimes}^{0.6} a, c \leftarrow_{\otimes}^{0.9} not_{\ominus} b\}, \quad P_2 = \{d \leftarrow_{\otimes}^{0.9} not_{\ominus} c, d \leftarrow_{\otimes}^{0.9} not_{\ominus} a\}, \\ HB_0 = \{a, b\}, \quad HB_1 = \{c\}, \quad HB_2 = \{d\}, \\ HB_0^\star = \{a, b\}, \quad HB_1^\star = \{a, b, c\}, \quad HB_2^\star = \{a, b, c, d\}. \end{array}$$

Like for ordinary stratified programs, as well as stratified dl-programs, a minimal model can be defined by a finite number of iterative least models, which naturally describes as the *canonical model* M_{KB} the semantics of stratified fuzzy dl-programs KB. We define its iterative least models M_i, $i \in \{0, \ldots, k\}$, by:

(i) M_0 is the least model of KB_0;
(ii) if $i > 0$, then M_i is the least model of KB_i such that $M_i | HB_{P_{i-1}}^\star = M_{i-1} | HB_{P_{i-1}}^\star$, where $M_i | HB_{P_{i-1}}^\star$ and $M_{i-1} | HB_{P_{i-1}}^\star$ denote the restrictions of the mappings M_i and M_{i-1} to $HB_{P_{i-1}}^\star$, respectively.

Then, M_{KB} denotes M_k. Note that M_{KB} is well-defined, since it does not depend on a particular stratification λ. Moreover, M_{KB} is in fact a minimal model of KB.

Example 4.4 (*Example 4.3 cont'd*). We have that

$$\begin{array}{l} M_0(a) = 0.8, \quad M_0(b) = 0.7, \quad M_0(x) = 0 \quad \text{for } x \in \{c, d\}, \\ M_1(x) = M_0(x) \quad \text{for } x \in \{a, b\}, \\ M_1(c) = \max(\min(0.8, 0.6), \min(1 - 0.7, 0.9)) = 0.6, \\ M_2(x) = M_1(x) \quad \text{for } x \in \{a, b, c\}, \\ M_2(d) = \max(\min(1 - 0.6, 0.9), \min(1 - 0.8, 0.9)) = 0.4, \\ M_{KB}(a) = 0.8, \quad M_{KB}(b) = 0.7, \quad M_{KB}(c) = 0.6, \quad M_{KB}(d) = 0.4. \end{array}$$

5. Probabilistic fuzzy description logic programs

In this section, we introduce probabilistic fuzzy dl-programs as a combination of stratified fuzzy dl-programs with Poole's independent choice logic (ICL) [22]. This then allows us to express probabilistic rules. The ICL is based on ordinary acyclic logic programs P under different "atomic choices", where each atomic choice along with P produces a first-order model, and one then obtains a probability distribution on the set of first-order models by placing a probability distribution on the different atomic choices. Here, differently from the ICL, we use stratified fuzzy dl-programs rather than ordinary acyclic logic programs, and we thus define a probability distribution on a set of fuzzy interpretations. In other words, we define a probability distribution on a set of rankings on the Herbrand base. We first define the syntax and then the semantics of probabilistic fuzzy dl-programs.

5.1. Syntax

We now define the syntax of probabilistic fuzzy dl-programs and of probabilistic queries for them. We first introduce fuzzy formulas, query constraints, and probabilistic formulas, and we define choice spaces and probabilities on choice spaces.

We define *fuzzy formulas* by induction as follows. The propositional constants *false* and *true*, denoted $\bot$ and $\top$, respectively, and all atoms $p(t_1, \ldots, t_k)$ are fuzzy formulas. If ϕ and ψ are fuzzy formulas, and $\otimes, \oplus, \rhd$, and $\ominus$ are conjunction, disjunction, implication, and negation strategies, respectively, then $(\phi \wedge_\otimes \psi), (\phi \vee_\oplus \psi)$, $(\phi \Rightarrow_\rhd \psi)$, and $\neg_\ominus \phi$ are also fuzzy formulas. For example, $\mathit{SportyCar}(x) \wedge_\otimes \mathit{hasPrice}(x, y_1)$ is a fuzzy formula.

A *query constraint* has either the form $(\phi\ \theta\ r)[l, u]$ or the form $(\mathbf{E}[\phi])[l, u]$, where $\theta \in \{\geqslant, >, <, \leqslant\}, r, l, u \in [0, 1]$, and ϕ is a fuzzy formula. Informally, the former asks whether the probability that the truth value v of ϕ satisfies $v\ \theta\ r$ lies in the interval $[l, u]$, while the latter asks whether the expected truth value of ϕ lies in the interval $[l, u]$. For example,

$$(\mathit{SportyCar}(x) \wedge_\otimes \mathit{hasPrice}(x, y_1) \geqslant 0.7)[0.3, 0.4]$$

asks whether the probability that the truth value v of $\mathit{SportyCar}(x) \wedge_\otimes \mathit{hasPrice}(x, y_1)$ satisfies $v \geqslant 0.7$ lies in the interval [0.3,0.4]. On the other hand,

$$(\mathbf{E}[\mathit{SportyCar}(x) \wedge_\otimes \mathit{hasPower}(x, y_2)])[0.5, 0.7]$$

asks whether the expected truth value of $\mathit{SportyCar}(x) \wedge_\otimes \mathit{hasPower}(x, y_2)$ lies in the interval [0.5,0.7]. Note that query constraints are used in queries only.

We define *probabilistic formulas* inductively as follows. Each query constraint is a probabilistic formula. If F and G are probabilistic formulas, then also $\neg F$ and $(F \wedge G)$. We use $(F \vee G)$ and $(F \Rightarrow G)$ to abbreviate $\neg(\neg F \wedge \neg G)$ and $\neg(F \wedge \neg G)$, respectively, and eliminate parentheses as usual. For example,

$$(\mathit{SportyCar}(x) \wedge_\otimes \mathit{hasPrice}(x, y_1) \geqslant 0.7)[0.3, 0.4] \vee (\mathbf{E}[\mathit{SportyCar}(x) \wedge_\otimes \mathit{hasPower}(x, y_2)])[0.5, 0.7]$$

is a probabilistic formula.

A *choice space* C is a set of pairwise disjoint and nonempty sets $A \subseteq HB$. Any $A \in C$ is an *alternative* of C and any $a \in A$ an *atomic choice* of C. Intuitively, every alternative $A \in C$ represents a random variable and every atomic choice $a \in A$ one of its possible values. A *total choice* of C is a set $B \subseteq HB$ such that $|B \cap A| = 1$ for all $A \in C$. Intuitively, every total choice B of C represents an assignment of values to all the random variables. A *probability* μ on a choice space C is a probability function on the set of all total choices of C. Intuitively, every μ is a probability distribution over the set of all variable assignments. Since C and all its alternatives are finite, μ can be defined by (i) a mapping $\mu : \bigcup C \to [0, 1]$ such that $\sum_{a \in A} \mu(a) = 1$ for all $A \in C$, and (ii) $\mu(B) = \prod_{b \in B} \mu(b)$ for all total choices B of C. Intuitively, (i) defines a probability over the values of each random variable of C, and (ii) assumes independence between the random variables.

Example 5.1 (*Shopping agent cont'd*). Suppose that the concepts "SportyCar", "hasPrice", and "hasPower" belong to the buyer's ontology language, while "SportsCar", "hasInvoice", and "hasHP" belong to the seller's ontology language. Assume that an automatic ontology alignment tool between the buyer's and the seller's terminology determines that "SportyCar" and "SportsCar" have the same meaning with probability 0.91, "hasPrice" and "hasInvoice" have the same meaning with probability 0.78, and that "hasPower" and "hasHP" have the same meaning with probability 0.83. We will consider then a choice space C with three alternatives $C_1 = \{sc_{pos}, sc_{neg}\}, C_2 = \{hi_{pos}, hi_{neg}\}$, and $C_3 = \{hhp_{pos}, hhp_{neg}\}$, along with the probability distributions

$$\begin{aligned}
&\mu(sc_{pos}) = 0.91, \quad \mu(sc_{neg}) = 0.09, \\
&\mu(hi_{pos}) = 0.78, \quad \mu(hi_{neg}) = 0.22, \\
&\mu(hhp_{pos}) = 0.83, \quad \mu(hhp_{neg}) = 0.17.
\end{aligned}$$

For example, $\mu(sc_{pos}) = 0.91$ encodes that with probability 0.91 "SportyCar" and "SportsCar" have the same meaning, while $\mu(sc_{neg}) = 0.09$ encodes that with probability 0.09 "SportyCar" and "SportsCar" have *not* the same meaning. Now, there are eight possible total choices B_i, which are shown below together with their probability $\mu(B) = \prod_{b \in B} \mu(b)$ (two decimal rounding). Note that $\sum_{B_i} \mu(B_i) = 1$.

B	Total choice	$\mu(B)$
B_1	sc_{pos}, hi_{pos}, hpp_{pos}	0.59
B_2	sc_{pos}, hi_{pos}, hpp_{neg}	0.12
B_3	sc_{pos}, hi_{neg}, hpp_{pos}	0.17
B_4	sc_{pos}, hi_{neg}, hpp_{neg}	0.03
B_5	sc_{neg}, hi_{pos}, hpp_{pos}	0.06
B_6	sc_{neg}, hi_{pos}, hpp_{neg}	0.01
B_7	sc_{neg}, hi_{neg}, hpp_{pos}	0.02
B_8	sc_{neg}, hi_{neg}, hpp_{neg}	0.00

A *probabilistic fuzzy dl-program* $KB = (L, P, C, \mu)$ consists of

(1) a fuzzy dl-program (L, P) such that $(L, P \cup \{p \leftarrow \mid p \in B\})$ is stratified for every total choice B of C;
(2) a choice space C such that no atomic choice in C coincides with the head of any fuzzy dl-rule in $ground(P)$;
(3) a probability μ on C.

Note that, since the total choices of C select subsets of P, and μ is a probability distribution on the total choices of C, every probabilistic fuzzy dl-program compactly represents a probability distribution on a finite set of stratified fuzzy dl-programs.

A *probabilistic query* to KB has the form $\exists F$, or $\exists(\alpha\ \theta\ r)[L, U]$, or $\exists(\mathbf{E}[\alpha])[L, U]$, where F is a probabilistic formula, α is a fuzzy formula, $r \in [0, 1]$, and L, U are variables.

Example 5.2 (*Shopping agent cont'd*). Assume that the buyer is looking for sports cars with price around 22 000 € or less and with around 150 HP. Assume also that the terminology between buyer and seller are different, as illustrated in Example 5.1. Then, we may consider the probabilistic fuzzy dl-program $KB = (L, P, C, \mu)$, where L is given by Example 3.4, and P is the following set of fuzzy dl-rules P, which model the query reformulation/retrieval steps using ontology mapping rules:

$$query(x) \leftarrow^{1}_{\otimes} SportyCar(x) \wedge_{\otimes} hasPrice(x, y_1) \wedge_{\otimes} hasPower(x, y_2) \wedge_{\otimes} DL[LeqAbout22000](y_1) \wedge_{\otimes} DL[Around150HP](y_2), \tag{21}$$

$$SportyCar(x) \leftarrow^{0.9}_{\otimes} DL[SportsCar](x) \wedge_{\otimes} sc_{pos}, \tag{22}$$

$$hasPrice(x, y) \leftarrow^{0.8}_{\otimes} DL[hasInvoice](x, y) \wedge_{\otimes} hi_{pos}, \tag{23}$$

$$hasPower(x, y) \leftarrow^{0.8}_{\otimes} DL[hasHP](x, y) \wedge_{\otimes} hhp_{pos}, \tag{24}$$

where $\otimes$ is given by $x \otimes y = \min(x, y)$. The choice space C and the probability distribution is as in Example 5.1. Intuitively, rule (21) is the buyer's request, but in a "different" terminology than the one of the car selling site. Rules (22)–(24) are the automatically built ontology alignment mapping rules. For example, rule (22) states that the predicate "SportyCar" of the buyer's terminology refers to the concept "SportsCar" of the selected site with probability 0.91 ($\mu(sc_{pos}) = 0.91$).

5.2. Semantics

We now define the semantics of probabilistic fuzzy dl-programs, and the notions of correct and tight answers for probabilistic queries addressed to such programs. We first define the semantics of fuzzy formulas in fuzzy interpretations and of probabilistic formulas in probability distributions over fuzzy interpretations. Intuitively, each fuzzy interpretation I maps a ground formula ϕ to $[0,1]$, i.e., $I(\phi) \in [0, 1]$. Furthermore, each fuzzy interpretation I has a probability $Pr(I)$ to be the actual one. Now, the probability of a constraint, e.g., $\phi \geqslant 0.8$ will be

$$Pr(\phi \geqslant 0.8) = \sum_{I \in \mathcal{I}, I(\phi) \geqslant 0.8} Pr(I),$$

where $\mathcal{I}$ denotes the set of all fuzzy interpretations. That is, we sum up the probabilities of the fuzzy interpretations that satisfy the condition that the degree of truth of ϕ under I is at least 0.8. On the other hand, as for any fuzzy interpretation $I, I(\phi) \in [0, 1]$, and I has probability $Pr(I)$ to occur, the expected truth value of ϕ is

$$\sum_{I \in \mathcal{I}} Pr(I) \cdot I(\phi).$$

As a consequence, a probabilistic query of, e.g., the form $(\phi \geqslant 0.8)[0.4, 0.6]$ will be satisfied if $Pr(\phi \geqslant 0.8)$ lies in $[0.4, 0.6]$, while a probabilistic query of, e.g., the form $(\mathbf{E}[\phi])[0.6, 0.7]$ will be satisfied if the expected truth of ϕ lies in [0.6,0.7].

Formally, we proceed as follows. A *world* I is a fuzzy interpretation over HB. We denote by $\mathcal{I}$ the set of all worlds over Φ. The *truth value* of ground fuzzy formulas ϕ in I, denoted $I(\phi)$, is inductively defined by (1) $I(\bot) = 0$ and $I(\top) = 1$, (2) $I(\phi \wedge_{\otimes} \psi) = I(\phi) \otimes I(\psi)$, (3) $I(\phi \vee_{\oplus} \psi) = I(\phi) \oplus I(\psi)$, (4) $I(\phi \Rightarrow_{\triangleright} \psi) = I(\phi) \triangleright I(\psi)$, and (5) $I(\neg_{\ominus} \phi) = \ominus I(\phi)$.

A *probabilistic interpretation* Pr is a probability function on $\mathcal{I}$ (that is, a mapping $Pr : \mathcal{I} \to [0, 1]$ such that (i) the set of all $I \in \mathcal{I}$ with $Pr(I) > 0$ is denumerable, and (ii) all $Pr(I)$ with $I \in \mathcal{I}$ sum up to 1).

The *probability* of $\phi\,\theta\, r$ in Pr, denoted $Pr(\phi\ \theta\ r)$, is

$$Pr(\phi\,\theta\, r) = \sum_{I \in \mathcal{I},\ I(\phi)\,\theta\, r} Pr(I).$$

The *expected truth value* of ϕ under Pr, denoted $\mathbf{E}_{Pr}[\phi]$, is

$$\mathbf{E}_{Pr}[\phi] = \sum_{I \in \mathcal{I}} Pr(I) \cdot I(\phi).$$

Notice that in the probability of $\phi\ \theta\ r$ in Pr (resp., in the expected truth value of ϕ under Pr), we implicitly (resp., explicitly) combine probabilities and truth values. Intuitively, $\mathbf{E}_{Pr}[\phi]$ may also be interpreted as the probability of ϕ seen as fuzzy event.

The notion of *satisfiability* of ground probabilistic formulas F in Pr, denoted $Pr \models F$, is inductively defined by

(1) $Pr \models (\phi\ \theta\ r)[l,u]$ iff $Pr(\phi\ \theta\ r) \in [l,u]$;
(2) $Pr \models (\mathbf{E}[\phi])[l,u]$ iff $\mathbf{E}_{Pr}[\phi] \in [l,u]$;
(3) $Pr \models \neg F$ iff not $Pr \models F$; and
(4) $Pr \models (F \wedge G)$ iff $Pr \models F$ and $Pr \models G$.

A probabilistic interpretation Pr is a *model* of a (possibly non-ground) probabilistic formula F iff $Pr \models F'$ for every ground instance F' of F.

We say Pr is the *canonical model* of a probabilistic fuzzy dl-program $KB = (L,P,C,\mu)$ iff every world $I \in \mathcal{I}$ with $Pr(I) > 0$ is the canonical model of $(L, P \cup \{p \leftarrow \mid p \in B\})$ for some total choice B of C such that $Pr(I) = \mu(B)$. We may represent a canonical model Pr as a set of pairs, $Pr = \{(I, Pr(I)) \mid I \in \mathcal{I}\}$. Notice that every such KB has a unique canonical model Pr.

Example 5.3 (*Shopping agent cont'd*). Consider the knowledge base $KB = (L,P)$ in Example 5.2. We have seen that there are eight total choices B_i for it (see Example 5.1). For each B_i, we consider $KB_i = (L, P \cup \{p \leftarrow \mid p \in B_i\})$ and its unique canonical model I_i. Now, let the set of worlds be $\mathcal{I} = \{I_1, \ldots, I_8\}$, and let $Pr(I_i) = \mu(B_i)$. Then, Pr is the canonical model of KB. The table below is an excerpt of the computation above, where the columns 3–8 report the degrees of truth of the fuzzy formulas $a_1 = SportyCar(mitsubishiES), a_2 = hasPrice(mitsubishiES, 24029), a_3 = hasPower(mitsubishiES, 162), a_4 = DL[LeqAbout22000](24029), a_5 = DL[Around150HP](162)$, and $a_6 = query(mitsubishiES)$ in the canonical model I_i.

Canonical model	$Pr(I)$	a_1	a_2	a_3	a_4	a_5	a_6
I_1	0.59	0.9	0.8	0.8	0.32	0.52	0.32
I_2	0.12	0.9	0.8	0.0	0.32	0.52	0.0
I_3	0.17	0.9	0.0	0.8	0.32	0.52	0.0
I_4	0.03	0.9	0.0	0.0	0.32	0.52	0.0
I_5	0.06	0.0	0.8	0.8	0.32	0.52	0.0
I_6	0.01	0.0	0.8	0.0	0.32	0.52	0.0
I_7	0.02	0.0	0.0	0.8	0.32	0.52	0.0
I_8	0.00	0.0	0.0	0.0	0.32	0.52	0.0

Note that, for example, since $I_2(hpp_{neg}) = 1$, we have $I_2(hpp_{pos}) = 0$, and we thus obtain $I_2(hasPo(MES, 24029)) = 0.0$ (see rule (24)).

We say F is a *consequence* of KB, denoted $KB \,\|\!\!\sim F$, iff the canonical model of KB is also a model of F.

A ground query constraint $(\phi\ \theta\ r)[l,u]$ (resp., $(\mathbf{E}[\phi])[l,u]$) is a *tight consequence* of KB, denoted $KB \,\|\!\!\sim_{tight} (\phi\,\theta\,r)[l,u]$ (resp., $KB \,\|\!\!\sim_{tight} (\mathbf{E}[\phi])[l,u]$), iff l (resp., u) is the infimum (resp., supremum) of $Pr(\phi\ \theta\ r)$ (resp., $\mathbf{E}_{Pr}[\phi]$) subject to the canonical model Pr of KB. A *correct answer* to a probabilistic query $\exists F$ is a substitution δ such that $F\delta$ is a consequence of KB. A *tight answer* to $\exists(\phi\ \theta\ r)[L,U]$ (resp., $\exists(\mathbf{E}[\phi])[L,U]$) is a substitution δ of variables L and U such that $(\phi'\ \theta\ r)[L,U]\delta$ (resp., $(\mathbf{E}[\phi'])[L,U]\delta$) is a tight consequence of KB, for all ground instances ϕ' of ϕ.

Example 5.4 (*Shopping agent cont'd*). The following are tight consequences of the probabilistic fuzzy dl-program $KB = (L,P,C,\mu)$ in Example 5.2:

$$(\mathbf{E}[query(mazdaMX5Miata)])[0.21, 0.21], \quad (\mathbf{E}[query(mitsubishiES)])[0.19, 0.19].$$

Note that, e.g., (see the data from Example 5.3)

$$\mathbf{E}_{Pr}[query(mitsubishiES)] = \sum_{I \in \mathcal{I}} Pr(I) \cdot I(query(mitsubishiES)) = Pr(I_1) \cdot I_1(query(mitsubishiES)) = 0.59 \cdot 0.32 = 0.19.$$

So, the agent ranks the *mazdaMX5Miata* first with expected truth degree 0.21 and the *mitsubishiES* second with expected truth degree 0.19.

6. Query processing in probabilistic fuzzy dl-programs

The canonical model of an ordinary positive resp. stratified program KB, as well as of a positive resp. stratified dl-program KB has a well-known fixpoint characterization in terms of an immediate consequence operator T_{KB}, which generalizes to positive resp. stratified fuzzy dl-programs. This can be exploited for a bottom-up computation of the canonical model of a positive resp. stratified fuzzy dl-program, and for query processing in probabilistic fuzzy dl-programs. Furthermore, it can also be used to delineate a special case where query processing in probabilistic fuzzy dl-programs can be done in polynomial time in the data complexity.

6.1. Positive fuzzy dl-programs

For a fuzzy dl-program $KB = (L, P)$, we define the operator T_{KB} on the set of fuzzy interpretations as follows. For every fuzzy interpretation I and ground atom $a \in HB$, let $T_{KB}(I)(a)$ be defined as follows (where $\max \emptyset = 0$):

$$T_{KB}(I)(a) = \max\{v \mid a \leftarrow^{n}_{\otimes} \varphi \in ground(P), v = I(\varphi) \otimes n\}.$$

Example 6.1 (*Shopping agent cont'd*). Consider the fuzzy interpretation I_1 from Example 5.3. Then, $T_{KB}(I_1)(query(mitsubishiES)) = 0.32$.

The following lemma shows that, if KB is positive, then the operator T_{KB} is monotonic. This result follows immediately from the fact that every dl-atom and every conjunction strategy in $ground(P)$ is monotonic.

Lemma 6.2. *Let $KB = (L, P)$ be a positive fuzzy dl-program. Then, T_{KB} is monotonic, that is, $I \preceq I'$ implies $T_{KB}(I) \preceq T_{KB}(I')$.*

Since every monotonic operator has a least fixpoint, also T_{KB} has one, denoted $lfp(T_{KB})$, which coincides with M_{KB}.

Example 6.3 (*Shopping agent cont'd*). Consider the fuzzy interpretations I_i and the knowledge bases KB_i from Example 5.3. Then, $T_{KB_i}(I_i)(a) = I_i(a)$ for all $a \in HB$, i.e., $T_{KB_i}(I_i) = I_i$ and, thus, I_i is a fixpoint of T_{KB_i}, which coincides with the canonical/least model M_{KB_i} of KB_i.

Furthermore, $lfp(T_{KB})$ can be computed by a finite fixpoint iteration, if KB is *closed* under a finite set of truth values $TV \subseteq [0, 1]$ (with $|TV| \geqslant 2$), which means that (i) each datatype predicate in KB is interpreted by a mapping to TV, (ii) each fuzzy modifier m in KB is interpreted by a mapping $f_m : TV \to TV$, (iii) each truth value in KB is from TV, and (iv) each combination strategy in KB is closed under TV (note that the ones of Łukasiewicz, Gödel, and Zadeh Logic are closed under every $TV_n = \{0, \frac{1}{n}, \ldots, \frac{n}{n}\}$ with $n > 0$). To this end, for every fuzzy interpretation I, we define

$$T^{i}_{KB}(I) = \begin{cases} I & \text{if } i = 0, \\ T_{KB}(T^{i-1}_{KB}(I)) & \text{if } i > 0. \end{cases}$$

The following theorem follows easily from the monotonicity of T_{KB} and the "closure" condition on KB.

Theorem 6.4. *Let $KB = (L, P)$ be a positive fuzzy dl-program. Then, $lfp(T_{KB}) = M_{KB}$. Furthermore, if KB is closed under a finite set of truth values $TV \subseteq [0, 1]$ (with $|TV| \geqslant 2$), then $lfp(T_{KB}) = \bigvee_{i=0}^{n} T^{i}_{KB}(I_{\bot}) = T^{n}_{KB}(I_{\bot})$, for some $n \geqslant 0$.*

6.2. Stratified fuzzy dl-programs

So far, we have considered positive fuzzy dl-programs. We next describe a fixpoint iteration for stratified fuzzy dl-programs. Using Theorem 6.4, we can characterize the canonical model M_{KB} of a stratified fuzzy dl-program KB by a sequence of fixpoint iterations along a stratification of KB. Let the operator $\widehat{T}^{i}_{KB}$ on fuzzy interpretations I be defined by

$$\widehat{T}^{i}_{KB}(I) = T^{i}_{KB}(I) \vee I$$

for all $i \geqslant 0$.

The following theorem follows easily from the fact that $\widehat{T}^{i}_{KB}$ is inflationary and monotonic, and the "closure" condition on KB.

Theorem 6.5. *Let $KB = (L, P)$ be a fuzzy dl-program with stratification λ of length $k \geqslant 0$. Suppose that KB is closed under some finite $TV \subseteq [0, 1]$ (with $|TV| \geqslant 2$). Define fuzzy interpretations $M_i, i \in \{-1, 0, \ldots, k\}$, by $M_{-1} = I_{\bot}$, and*

$$M_i = \widehat{T}^{n_i}_{KB_i}(M_{i-1})$$

for each $i \geqslant 0$, where $n_i \geqslant 0$ such that

$$\widehat{T}^{n_i}_{KB_i}(M_{i-1}) = \widehat{T}^{n_i+1}_{KB_i}(M_{i-1}).$$

Then, $M_k = M_{KB}$.

Informally, to build the canonical model, we start with $I_{\bot}$. Then, we consider the first stratum KB_0 and compute the least fixpoint M_0 of $\widehat{T}_{KB_0}$. Then, we consider the second stratum KB_1 and interpretation M_0, and compute the fixpoint M_1 obtained by reiterating $\widehat{T}_{KB_1}$ starting with M_0, and so on until considering the last stratum KB_k and interpretation M_{k-1}.

```
Algorithm canonical_model

Input: probabilistic fuzzy dl-program $KB = (L, P, C, \mu)$.
Output: canonical model $Pr$ of $KB$ (encoded as $\{(I, Pr(I)) \mid I \in \mathcal{I},\ Pr(I) > 0\}$).

1.  for every total choice $B$ of $C$ do begin
2.    compute the canonical model $I$ of the
        stratified fuzzy dl-program $(L, P \cup \{p \leftarrow \mid p \in B\})$;
3.    $Pr(I) := \mu(B)$
4.  end;
5.  return $Pr$.
```

Fig. 3. Algorithm canonical_model.

```
Algorithm tight_answer

Input: probabilistic fuzzy dl-program $KB = (L, P, C, \mu)$ and
         probabilistic query $Q = \exists(\phi\, \theta\, r)[L, U]$ (resp., $Q = \exists(\mathbf{E}[\phi])[L, U]$).
Output: tight answer $\delta = \{L/l, U/u\}$ for $Q$ to $KB$.

1.  $Pr :=$ canonical_model$(KB)$;
2.  $l := 1$;
3.  $u := 0$;
4.  for every ground instance $\phi'$ of $\phi$ do begin
5.    $l := \min(l, Pr(\phi'\, \theta\, r))$; (resp., $l := \min(l, \mathbf{E}[\phi'])$;)
6.    $u := \max(u, Pr(\phi'\, \theta\, r))$ (resp., $u := \max(u, \mathbf{E}[\phi'])$)
7.  end;
8.  return $\delta = \{L/l, U/u\}$.
```

Fig. 4. Algorithm tight_answer.

Example 6.6. Consider Examples 4.3 and 4.4. By definition, $KB_i = (L, P_i)$, where P_i is as from Example 4.3. Now, it can be verified that the iteration as from Theorem 6.5 determines the models as described in Example 4.4. Specifically, $\widehat{T}^1_{KB_0}(I_\perp) = M_0$, $\widehat{T}^1_{KB_1}(M_0) = M_1$, and $\widehat{T}^1_{KB_2}(M_1) = M_2 = M_{KB}$.

6.3. Probabilistic fuzzy dl-programs

Algorithm canonical_model in Fig. 3 computes the canonical model *Pr* of a given probabilistic fuzzy dl-program *KB*. It is essentially based on a reduction to computing the canonical model of stratified fuzzy dl-programs (see line 2), which can be done using the above finite sequence of finite fixpoint iterations.

Example 6.7 (*Shopping agent cont'd*). Consider Example 5.3. It is easy to show that indeed, the canonical models I_i can be computed by a fixpoint iteration of $\widehat{T}_{KB_i}$.

Algorithm tight_answer in Fig. 4 computes the tight answer $\theta = \{L/l, U/u\}$ for a given probabilistic query $Q = \exists(\phi\ \theta\ r)[L, U]$ (resp., $Q = \exists(\mathbf{E}[\phi])[L, U]$) to a given probabilistic fuzzy dl-program *KB*. It first computes the canonical model of *KB* in line 1 and then the tight answer $\theta = \{L/l, U/u\}$ in lines 2–8.

Example 6.8 (*Shopping agent cont'd*). Consider again Example 5.4 and the probabilistic query $(\mathbf{E}[query(mitsubishiES)])[L, U]$. Since $query(mitsubishiES)$ is ground, steps 4–8 of the algorithm in Fig. 4 are executed just once. Since $\mathbf{E}_{Pr}[query(mitsubishiES)] = 0.19$, we get immediately the tight answer $\delta = \{L/0.19, U/0.19\}$.

6.4. Tractable special case

As shown in [14], given a stratified fuzzy dl-program $KB = (L, B)$, computing the truth value of a ground atom $a \in HB$ in the canonical model of *KB* (and thus computing the canonical model of *KB* as a whole) can be done in polynomial time in the data complexity, when (i) *L* is defined in *fuzzy DL-Lite* [25], and (ii) *KB* is closed under $TV_n = \{0, \frac{1}{n}, \ldots, \frac{n}{n}\}$ for some $n > 0$. Hence, given a probabilistic fuzzy dl-program $KB = (L, P, C, \mu)$ with (i) and (ii), and a probabilistic query $Q = \exists(\phi\ \theta\ r)[L, U]$ (resp., $Q = \exists(\mathbf{E}[\phi])[L, U]$) with ground α, computing the tight answer for *Q* to *KB* can be also done in polynomial time in the data complexity. Here, in the *data complexity*, we keep nearly all of *KB* (including the predicate symbols in Φ and also the number of truth values $n + 1$) fixed, except for Φ_c, the fuzzy concept and role assertion axioms in *L*, and the fuzzy facts in *P*.

7. Conclusion

We have presented probabilistic fuzzy dl-programs for the Semantic Web, which allow for handling both probabilistic uncertainty (especially for probabilistic ontology mapping and probabilistic data integration) and fuzzy vagueness (especially for dealing with vague concepts) in a uniform framework. We have defined important concepts related to both probabilistic uncertainty and fuzzy vagueness. Furthermore, we have described a shopping agent example, which gives evidence of the usefulness of probabilistic fuzzy dl-programs in realistic Web applications. We have also provided algorithms for query processing in such programs, which can be done in polynomial time in the data complexity under suitable assumptions.

To our knowledge, to date there are no other Semantic Web formalisms that allow for handling both uncertainty and vagueness in a uniform framework. Most closely related to the presented approach are hybrid integrations of rules and ontologies that allow for handling probabilistic uncertainty and fuzzy vagueness separately. The earliest work on probabilistic dl-programs [15] is based on loosely coupled dl-programs under the answer set and the well-founded semantics. Recent extensions include a tractable variant [17] and a tightly coupled version [2,3], especially for representing and reasoning with ontology mappings. Another approach is the one introduced in [16]: differently from the one presented here, it is a purely probabilistic approach, and it is based on probabilistic default reasoning as underlying reasoning model. A related (less expressive approach is [23], which is based on Bayesian logic programs. As for fuzzy vagueness, [14] proposes loosely coupled fuzzy dl-programs under the answer set semantics, while [27,18] present tightly coupled fuzzy dl-programs, and [19] presents a *top-k* retrieval technique in this context. For a more detailed overview, see [30].

An interesting topic for future research is to generalize probabilistic fuzzy dl-programs by non-stratified default-negations in rule bodies, which may be done via a well-founded semantics for fuzzy dl-programs. Other possible extensions are classical negations in rules, and disjunctions in rule heads. A further interesting issue is to explore how to update probabilistic fuzzy dl-programs.

Acknowledgement

This work has been partially supported by the German Research Foundation (DFG) under the Heisenberg Programme. We thank the reviewers of this paper and its ECSQARU-2007 preliminary version for their useful and constructive comments, which have helped to improve this work.

References

[1] T. Berners-Lee, Weaving the Web, Harper, San Francisco, 1999.
[2] A. Calì, T. Lukasiewicz, Tightly integrated probabilistic description logic programs for the Semantic Web, in: Proceedings ICLP-2007, LNCS, vol. 4670, Springer, 2007, pp. 428–429.
[3] A. Calì, T. Lukasiewicz, L. Predoiu, H. Stuckenschmidt, Tightly integrated probabilistic description logic programs for representing ontology mappings, in: Proceedings FoIKS-2008, LNCS, vol. 4932, Springer, 2008, pp. 178–198.
[4] J. Callan, Distributed information retrieval, in: W.B. Croft (Ed.), Advances in Information Retrieval, Kluwer, 2000, pp. 127–150.
[5] J. Callan, M. Connell, Query-based sampling of text databases, ACM Trans. Inform. Syst. 19 (2) (2001) 97–130.
[6] T. Eiter, T. Lukasiewicz, R. Schindlauer, H. Tompits, Combining answer set programming with description logics for the Semantic Web, in: Proceedings KR-2004, AAAI Press, 2004, pp. 141–151 (Extended version: Artif. Intell. 172 (12/13) (2008) 1495–1539).
[7] J. Euzenat, P. Shvaiko, Ontology Matching, Springer, 2007.
[8] D. Fensel, W. Wahlster, H. Lieberman, J. Hendler (Eds.), Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential, MIT Press, 2002.
[9] T. Flaminio, L. Godo, A logic for reasoning about the probability of fuzzy events, Fuzzy Sets Syst. 158 (6) (2007) 625–638.
[10] G. Fenza, V. Loia, S. Senatore, A hybrid approach to Semantic Web services matchmaking, Int. J. Approx. Reason. 48 (3) (2008) 808–828.
[11] N. Fuhr, A decision-theoretic approach to database selection in networked IR, ACM Trans. Inform. Syst. 3 (17) (1999) 229–249.
[12] P. Hájek, Metamathematics of Fuzzy Logic, Kluwer, 1998.
[13] I. Horrocks, P.F. Patel-Schneider, Reducing OWL entailment to description logic satisfiability, in: Proceedings ISWC-2003, LNCS, vol. 2870, Springer, 2003, pp. 17–29.
[14] T. Lukasiewicz, Fuzzy description logic programs under the answer set semantics for the Semantic Web, in: Proceedings RuleML-2006, IEEE Computer Society, 2006, pp. 89–96 (Extended version: Fundam. Inform. 82 (3) (2008) 289–310).
[15] T. Lukasiewicz, Probabilistic description logic programs, Int. J. Approx. Reason. 45 (2) (2007) 288–307.
[16] T. Lukasiewicz, Probabilistic description logic programs under inheritance with overriding for the Semantic Web, Int. J. Approx. Reason. 49 (1) (2008) 18–34.
[17] T. Lukasiewicz, Tractable probabilistic description logic programs, in: Proceedings SUM-2007, LNCS, vol. 4772, Springer, 2007, pp. 143–156.
[18] T. Lukasiewicz, U. Straccia, Tightly integrated fuzzy description logic programs under the answer set semantics for the Semantic Web, in: Proceedings RR-2007, LNCS, vol. 4524, Springer, 2007, pp. 289–298.
[19] T. Lukasiewicz, U. Straccia, Top-*k* retrieval in description logic programs under vagueness for the Semantic Web, in: Proceedings SUM-2007, LNCS, vol. 4772, Springer, 2007, pp. 16–30.
[20] T. Lukasiewicz, U. Straccia, Managing uncertainty and vagueness in description logics for the Semantic Web, J. Web Sem. 6 (4) (2008) 291–308.
[21] H. Nottelmann, U. Straccia, Information retrieval and machine learning for probabilistic schema matching, Inform. Process. Manage. 43 (3) (2007) 552–576.
[22] D. Poole, The independent choice logic for modelling multiple agents under uncertainty, Artif. Intell. 94 (1/2) (1997) 7–56.
[23] L. Predoiu, H. Stuckenschmidt, A probabilistic framework for information integration and retrieval on the Semantic Web, in: Proceedings InterDB-2007 Workshop on Database Interoperability, 2007.
[24] M.E. Renda, U. Straccia, Web metasearch: rank vs. score-based rank aggregation methods, in: Proceedings SAC, ACM Press, 2003, pp. 841–846.

[25] U. Straccia, Answering vague queries in fuzzy DL-Lite, in: Proceedings IPMU-2006, 2006, pp. 2238–2245.
[26] U. Straccia, A fuzzy description logic for the Semantic Web, in: E. Sanchez (Ed.), Fuzzy Logic and the Semantic Web, Capturing Intelligence, Elsevier, 2006, pp. 73–90 (Chapter 4).
[27] U. Straccia, Fuzzy description logic programs, in: Proceedings IPMU-2006, 2006, pp. 1818–1825.
[28] U. Straccia, R. Troncy, oMAP: Combining classifiers for aligning automatically OWL ontologies, in: Proceedings WISE-2005, LNCS, vol. 3806, Springer, 2005, pp. 133–147.
[29] U. Straccia, R. Troncy, Towards distributed information retrieval in the Semantic Web, in: Proceedings ESWC-2006, LNCS, vol. 4011, Springer, 2006, pp. 378–392.
[30] U. Straccia, Managing uncertainty and vagueness in description logics, logic programs and description logic programs, in: Reasoning Web, 4th International Summer School, Tutorial Lectures, LNCS, vol. 5224, Springer, 2008, pp. 54–103.
[31] W3C. OWL Web ontology language overview, 2004. W3C Recommendation (10 February 2004). Available at: <http://www.w3.org/TR/2004/REC-owl-features-20040210/>.