Ranking Services Using Fuzzy HEX Programs*

Stijn Heymans[1,2] and Ioan Toma[3]

[1] Knowledge-Based Systems Group, Institute of Information Systems, Vienna University of Technology, Austria
heymans@kr.tuwien.ac.at
[2] Computational Web Intelligence, Department of Applied Mathematics and Computer Science, Ghent University, Belgium
[3] Semantic Technology Institute (STI) Innsbruck, University of Innsbruck, Austria
ioan.toma@sti-innsbruck.at

Abstract. The need to reason with knowledge expressed in both Logic Programming (LP) and Description Logics (DLs) paradigms on the Semantic Web lead to several integrating formalisms, e.g., Description Logic programs (*dl-programs*) allow a logic program to retrieve results from and feed results to a DL knowledge base. Two functional extensions of dl-programs are HEX programs and fuzzy dl-programs. The former abstract away from DLs, allowing for general external queries, the latter deal with the uncertain, vague, and inconsistent nature of knowledge on the Web by means of fuzzy logic mechanisms. In this paper, we generalize both HEX programs and fuzzy dl-programs to *fuzzy* HEX *programs*: a LP-based paradigm, supporting both fuzziness as well as reasoning with external sources. We define basic syntax and semantics and analyze the framework semantically, e.g., by investigating the complexity. Additionally, we provide a translation from fuzzy HEX programs to HEX programs, enabling an implementation via the dlvhex reasoner. Finally, we illustrate the use of fuzzy HEX programs for ranking services by using them to model non-functional properties of services and user preferences.

1 Introduction

Logic Programming (LP) [2] and Description Logics (DLs) [1] are two of the main underlying knowledge representation and reasoning paradigms of the Semantic Web, a machine-understandable instead of just machine-readable Web [4]. Logic Programming underlies, for example, several variants of the Web Service Modeling Language WSML [6] and Description Logics form the basis of the ontology language OWL-DL [3].

As the Semantic Web is about understanding knowledge and automatizing inferences from this knowledge, it is not surprising that there is a lot of interest

* Stijn Heymans is partially supported by the Austrian Science Fund (FWF) under project P20305-N18 and the Fund for Scientific Research Flanders (FWO Vlaanderen) under project 3G010107. Ioan Toma is supported by the EU FP7 IST project 27867, *SOA4ALL - Service Oriented Architectures For All*.

D. Calvanese and G. Lausen (Eds.): RR 2008, LNCS 5341, pp. 181–196, 2008.

in the integration of these paradigms (see, e.g., [5] for an overview). One of these integrating approaches are *Description Logic Programs*, dubbed *dl-programs* [11], that take a LP view on a DL knowledge base: logic programs are able to query DL knowledge bases via *dl-atoms*. Moreover, dl-atoms can stream knowledge from the logic program to the DL knowledge base, where it can be used to make additional DL inferences (which can then in turn be used in the LP deduction process). In effect, there is a bi-directional stream of information between the logic program and the DL knowledge base.

In [10], dl-programs were generalized to HEX *programs*. HEX programs combine higher-order reasoning - naively put, they allow for variables to appear in the predicate position, enabling thus meta-reasoning over concepts - and external atoms. The latter generalize dl-atoms as they do not just access DL knowledge bases but are associated with any external function - one can use them, e.g., to query RDF repositories or SQL databases.

Another extension to dl-programs was inspired by the uncertainty, vagueness, and inconsistency of the (Semantic) Web. As anyone can produce knowledge on the Web, it is impossible to ensure that all knowledge on the Web is logically true. Moreover, often there is a need (as there is in real-life) to express vague concepts, such as *very*, *beautiful*, or *old/young*; a need that is not met by traditional two-valued logics like LP or DLs. And finally, the Web is inconsistent: source A might have another (contradicting) opinion than source B on a topic. Together with the need for integrating approaches, this lead to so-called *fuzzy dl-programs* [14,12]. Fuzzy dl-programs extend dl-programs by allowing to query fuzzy DLs [17] and by using fuzzy dl-rules on the LP side.

Intuitively, fuzzy dl-rules use *combination strategies* instead of the usual conjunction, disjunction, and negation in normal LP rules. Those combination strategies do nothing else than computing a resulting truth value based on two (or one, in the case of the negation strategy) input truth values, where truth values, in contrast with two-valued logics, range over the interval $[0, 1]$. For example, in normal LP, a rule $\mathit{fail} \leftarrow \mathit{not\ study}, \mathit{smart}$ where *study* is false (or 0) and *smart* is true, results in a truth value of 1 for *fail*. A fuzzy variant could be $\mathit{fail} \leftarrow_{\otimes_G} \mathit{not}_{\ominus_L}\ \mathit{study} \otimes_G \mathit{smart} \geq 0.5$ where $\otimes_G$ is the Gödel conjunction (which takes the minimum of two values) and $\ominus_L$ is the Lukasiewizc negation (which takes the complement of a value w.r.t. 1). If *study* has a fuzzy value of 0.4 and *smart* of 0.9, we would have that $\mathit{not}_{\ominus_L}\ \mathit{study}$ has a value of 0.6. The value 0.5 indicates to what extent the rule should be satisfied. Using $\otimes_G$ we would have that the value of the body (the part to the right of $\leftarrow$) has to be $0.5 \otimes_G 0.6 \otimes_G 0.9 = 0.5$ and that the value of *fail* should be at least this value (0.5) in order to make the fuzzy rule satisfied. Note that the value 0.5 that indicates to which degree a rule should be satisfied is used to calculate the value of the body.

In this paper, we generalize both extensions - from dl-programs to HEX programs and from dl-programs to fuzzy dl-programs - to *fuzzy* HEX *programs*. Fuzzy HEX programs thus support higher-order reasoning, reasoning with external sources like DL knowledge bases or in general with external functions (e.g., *sum*, *max*), and fuzzy reasoning. We establish the basic syntax and semantics

of such programs and show that the complexity of disjunction-free fuzzy HEX programs, under equal conditions for the external predicates and appropriately behaving fuzzy combination strategies, is the same as for disjunction-free HEX programs, namely NEXPTIME-complete.

We furthermore establish a translation from fuzzy HEX programs to HEX programs, basically writing the combination strategies as external predicates that can be computed by external functions. This enables reasoning with fuzzy HEX programs using the DLVHEX [9] reasoner for HEX programs.

To show the applicability of fuzzy HEX programs, we use them to describe non-functional properties of Web services, enabling better ranking of services. A *service* is a provision of value to a client [15], e.g., the delivery of a package with some specified constraints. *Service ranking* is then the process which generates an ordered list of services out of the candidate services set according to user's preferences. As ranking criteria, specified by the user, various aspects of a service description can be used. We differentiate between (1) *functional*, (2) *behavioral*, and (3) *non-functional*. The *functional* description contains the formal specification of what exactly the service can do. The *behavioral* description is about how the functionality of the service can be achieved in terms of interaction with the service as well as in terms of functionality required from other services. Finally, the *non-functional description* captures constraints over the previous two [7]. For example, in case of a shipping service, invoking its functionality (shipping a package) might be constrained by paying a certain amount (*price* as non-functional property).

As part of our previous work [18], we have proposed an approach for the service ranking problem based on the evaluation of non-functional properties such as price, response time, liability, etc. Rules encoding conditions and constraints over multiple non-functional properties are used to model both users and service provides perspectives. Although this modeling approach is useful for modeling some of the multitude of non-functional properties such as liability/contractual obligations, for properties, like price or delivery time, a more natural choice to express requests and preferences requires a formalism for handling vagueness and imprecision, e.g., imagine a provider advertising "the cheapest service". Another issue is that service descriptions, besides fuzzy information, often need to refer to external libraries and data sources. Fuzzy HEX programs address exactly these requirements.

The remainder of the paper starts with an introduction to HEX programs in Section 2. Section 3 defines fuzzy HEX programs with its basic properties. In Section 4, we show that fuzzy HEX programs generalize both HEX programs and fuzzy dl-programs, and Section 5 gives complexity results for fuzzy HEX programs as well as a translation from fuzzy HEX programs to HEX programs, allowing an implementation using DLVHEX. Section 6 contains an application of fuzzy HEX programs to the problem of ranking services and we give directions for further research in Section 7. Proofs of the key results can be found at `http://www.kr.tuwien.ac.at/staff/heymans/fuzzy-hex-proofs.pdf`.

2 Preliminaries: HEX Programs

We introduce HEX programs as in [10]. Assume the existence of 3 mutually disjoint sets $\mathcal{C}, \mathcal{X}$, and $\mathcal{G}$, consisting of constants, variables, and external predicates respectively. A *term* is either a *constant* $a \in \mathcal{C}$ or a *variable* $X \in \mathcal{X}$, denoted with symbols starting with lower-case or upper-case letters respectively. A *higher-order atom* is of the form $t_0(t_1, \ldots, t_n)$ for terms t_i, $0 \leq i \leq n$. If t_0 is a constant, we call $t_0(t_1, \ldots, t_n)$ an *ordinary atom*. An *external predicate* from $\mathcal{G}$ starts with the symbol #, e.g., $\#g$ or $\#sqrt$, where each external predicate has an associated *input* and *output arity*. An *external atom* is of the form $\#g[t_1, \ldots, t_n](s_1, \ldots, s_m)$ where $t_1, \ldots, t_n$ is the *input list* of terms for the input arity n of $\#g$ and $s_1, \ldots, s_m$ is the *output list* of terms for the output arity m of $\#g$.

A *rule* r is of the form:

$$a_1 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_n, not\ b_{n+1}, \ldots, not\ b_m \tag{1}$$

where $a_1, \ldots, a_k$ are higher-order atoms, and $b_1, \ldots, b_m$ are higher-order or external atoms. The *head* of r is $\text{head}(r) = \{a_1, \ldots, a_k\}$, and the *body* of r is $\text{body}(r) = \text{body}^+(r) \cup \text{body}^-(r)$ with $\text{body}^+(r) = \{b_1, \ldots, b_n\}$ and $\text{body}^-(r) = \{b_{n+1}, \ldots, b_m\}$. A rule is ordinary if it only contains ordinary atoms. If $k = 1$ we call the rule *disjunction-free*. A *(disjunction-free)* HEX *program* is a finite set P of (disjunction-free) rules.

An atom (higher-order or external), rule, or program, is *ground* if no variables appear in it. A *grounding* of a program P is a ground program $gr(P)$ that contains all possible ground rules resulting from replacing the variables in those rules with all possible constants from $\mathcal{C}$. The *Herbrand Base* of P, denoted $\mathcal{B}_P$, is the set of all possible ground versions of atoms (higher-order or external) occuring in P using constants of $\mathcal{C}$. Note that the Herbrand Base only contains ordinary atoms and external atoms. If $\mathcal{C}$, $\mathcal{X}$, or $\mathcal{G}$ are not explicitly given, we assume they are implicitly given by the program P under consideration.

An *interpretation* I of a program P is a set $I \subseteq \mathcal{B}_P$ of ordinary atoms (i.e., no external atoms). We say that I is a *model* of a ground ordinary atom a, denoted $I \models a$, if $a \in I$.

We associate with every external predicate symbol $\#g \in \mathcal{G}$, an $(n+m+1)$-ary function $f_{\#g}$, that assigns a tuple $(I, y_1, \ldots, y_n, x_1, \ldots, x_m)$ to 0 or 1, with n the input arity of $\#g$ and m the output arity of $\#g$, I an interpretation, and y_i, x_j constants. I is then a *model* of a ground external atom $a = \#g[y_1, \ldots, y_n](x_1, \ldots, x_m)$, denoted $I \models a$, if and only if $f_{\#g}(I, y_1, \ldots, y_n, x_1, \ldots, x_m) = 1$. For a ground atom (possibly external) a, we have $I \models not\ a$ iff $I \not\models a$. This definition extends for sets containing ground ordinary and ground external atoms as usual.

We say that a ground rule r is *satisfied* by I, denoted $I \models r$, if, whenever $I \models \text{body}^+(r) \cup not\ \text{body}^-(r)$, we have that there is a a_i, $1 \leq i \leq k$, such that $I \models a_i$. For a HEX program P, I is a *model* of P, denoted $I \models P$, iff $I \models r$ for each $r \in gr(P)$. We define the *FLP-reduct* P^I of a program w.r.t an interpretation I as all rules $r \in gr(P)$ such that $I \models \text{body}^+(r) \cup not\ \text{body}^-(r)$. An interpretation I of P is an *answer set* of P iff I is a minimal model of P^I.

3 Fuzzy HEX Programs

We use the definition of fuzzy *dl-rules* from [14,12] as an inspiration to extend HEX programs to fuzzy HEX programs, and we start by identifying different *combination strategies*:

- The *negation* strategy $\ominus : [0,1] \to [0,1]$, where we call $\ominus v$, $v \in [0,1]$, the *negation* of v. The negation strategy has to be *antitonic*, i.e., if $v_1 \leq v_2$, then $\ominus v_1 \geq \ominus v_2$. Furthermore, we have that $\ominus 1 = 0$ and $\ominus 0 = 1$. Particular negation strategies ([17]) are, for example, the *Lukasiewizc negation* $\ominus_L$, defined by $\ominus_L x = 1 - x$, or the *Gödel negation* $\ominus_G$, defined by $\ominus 0 = 1$ and $\ominus x = 0$ if $x > 0$.
- The *conjunction* strategy $\otimes : [0,1] \times [0,1] \to [0,1]$, where we call $v_1 \otimes v_2$, $v_1, v_2 \in [0,1]$, the conjunction of v_1 and v_2. The conjunction strategy has to be *commutative*, *associative*, and *monotone* (if $v_1 \leq v_1'$ and $v_2 \leq v_2'$, then $v_1 \otimes v_2 \leq v_1' \otimes v_2'$). Furthermore, we need to have that $v \otimes 1 = v$ and $v \otimes 0 = 0$. Particular conjunction strategies (also called *t-norms* [17]) are, for example, the *Lukasiewizc conjunction* $\otimes_L$, defined by $x \otimes_L y = \max(x + y - 1, 0)$, the *Gödel conjunction* $\otimes_G$, defined by $x \otimes_G y = \min(x, y)$, and the *product conjunction* $\otimes_P$, defined by $x \otimes_P y = x.y$.
- The *disjunction* strategy $\oplus : [0,1] \times [0,1] \to [0,1]$, where we call $v_1 \oplus v_2$, $v_1, v_2 \in [0,1]$, the disjunction of v_1 and v_2. The disjunction strategy has to be *commutative*, *associative*, and *monotone* (if $v_1 \leq v_1'$ and $v_2 \leq v_2'$, then $v_1 \oplus v_2 \leq v_1' \oplus v_2'$). Furthermore, we need to have that $v \oplus 1 = 1$ and $v \oplus 0 = v$. Particular disjunction strategies (also called *s-norms* [17]) are, for example, the *Lukasiewizc disjunction* $\oplus_L$, defined by $x \oplus_L y = \min(x + y, 1)$, the *Gödel disjunction* $\oplus_G$, defined by $x \oplus_G y = \max(x, y)$, and the *product disjunction* $\oplus_P$, defined by $x \oplus_P y = x + y - x.y$.

Definition 1. *A* fuzzy rule *r is of the form*

$$a_1 \oplus_1 \ldots \oplus_{k-1} a_k \leftarrow_{\otimes_0} b_1 \otimes_1 \ldots \otimes_{n-1} b_n \otimes_n not_{\ominus_{n+1}} b_{n+1} \otimes_{n+1} \ldots \otimes_{m-1} not_{\ominus_m} b_m \geq v \quad (2)$$

where $a_1, \ldots, a_k$ are higher-order atoms, $b_1, \ldots, b_m$ are higher-order, external atoms, or elements from $[0,1]$, and $v \in [0,1]$. The head *and* body *of r is defined as before. A* (disjunction-free) fuzzy HEX program *is a finite set P of (disjunction-free) fuzzy rules.*

Note that a fuzzy rule can contain different negation strategies; the order of evaluation of such strategies will be *left-to-right*.

Ground atoms, rules, programs, as well as a *grounding* are defined similarly as for HEX programs.

A *fuzzy interpretation* of a fuzzy HEX program is a mapping $I : O_P \subseteq \mathcal{B}_P \to [0,1]$ where O_P are the ordinary atoms in $\mathcal{B}_P$. Define $I \subseteq J$ for fuzzy interpretations I and J of P, if $I(a) \leq J(a)$ for each $a \in O_P$. We call I *minimal* if there is no interpretation $J \neq I$ such that $J \subseteq I$. The *fuzzy value* v_I of a ground ordinary atom a w.r.t. an interpretation I is $v_I(a) = I(a)$.

We associate with every external predicate symbol $\#g \in \mathcal{G}$, an $(n+m+1)$-ary function $f_{\#g}$, that assigns a tuple $(I, y_1, \ldots, y_n, x_1, \ldots, x_m)$ to $[0,1]$, with n the input arity of $\#g$ and m the output arity of $\#g$, I a fuzzy interpretation, and y_i, x_j constants. The *fuzzy value* v_I of a ground external atom $a = \#g[y_1, \ldots, y_n](x_1, \ldots, x_m)$ w.r.t. an interpretation I is $v_I(a) = f_{\#g}(I, y_1, \ldots, y_n, x_1, \ldots, x_m)$. We complete the definition of v_I by defining it for values v from $[0,1]$ as $v_I(v) = v$.

A fuzzy interpretation I *satisfies* a ground fuzzy rule (2) iff

$$v_I(a_1) \oplus_1 \ldots \oplus_{k-1} v_I(a_k) \geq v \otimes_0 v_I(b_1) \otimes_1 \ldots \otimes_{n-1} v_I(b_n) \otimes_n \ominus_{n+1} v_I(b_{n+1}) \otimes_{n+1} \ldots \otimes_{m-1} \ominus_m v_I(b_m) \ . \quad (3)$$

A fuzzy interpretation I is a *fuzzy model*[1] of a fuzzy HEX program P if it satisfies every rule in $gr(P)$.

The *FLP-reduct* P^I of a fuzzy HEX program w.r.t a fuzzy interpretation I are all rules $r \in gr(P)$ of the form (2) where

$$v \otimes_0 v_I(b_1) \otimes_1 \ldots \otimes_{n-1} v_I(b_n) \otimes_n \ominus_{n+1} v_I(b_{n+1}) \otimes_{n+1} \ldots \otimes_{m-1} \ominus_m v_I(b_m) > 0 \ .$$

We can then define fuzzy answer sets as follows:

Definition 2. *Let P be a fuzzy* HEX *program. A fuzzy interpretation I of P is a* fuzzy answer set *of P iff I is a minimal fuzzy model of P^I.*

Example 1. Take P with rules $a \leftarrow_{\otimes_P} not_{\ominus_L}\ b \geq 1$ and $b \leftarrow_{\otimes_P} not_{\ominus_L}\ a \geq 1$. One can check that a fuzzy interpretation I_1 with $I_1(a) = 0.8$ and $I_1(b) = 0$ is not a model of P and thus not a fuzzy answer set. On the other hand, I_2 with $I_2(a) = 0.6$ and $I_2(b) = 0.4$ is a fuzzy answer set.

Example 2. Take the program P with rule $a \leftarrow_{\otimes_P} not_{\ominus_L}\ a \geq 1$. Although the normal program $a \leftarrow not\ a$ has no answer sets, the fuzzy version of this program has a fuzzy answer set I where $I(a) = \frac{1}{2}$, i.e., if one is equally unsure about a as about *not* a, the contradicting rule is no longer relevant.

A *positive* fuzzy HEX program is a program without negation strategies.

We have that for positive programs the FLP-reduct has no influence on the fuzzy answer sets:

Proposition 1. *Let P be a positive fuzzy* HEX *program. Then, M is a fuzzy answer set of P iff M is a minimal fuzzy model of P.*

Proposition 1 does not necessarily hold if P is not positive as one can see from the fuzzy program $a \leftarrow_{\otimes_P} not_{\ominus}\ a \geq 1$ where we define $\ominus$ as follows: $\ominus x = 1$ for $x < 0.1$ and $x = 0$ for $x \geq 0.1$.

[1] We will omit the modifier *fuzzy* if it is clear from the context.

4 Fuzzy HEX Programs Generalize HEX Programs and Fuzzy dl-Programs

To show that HEX programs are properly embedded in fuzzy HEX programs we introduce a *crisp conjunction* $x \otimes_c y = 1$ if $x = 1 \wedge y = 1$ and 0 else, a *crisp disjunction* $x \oplus_c y = 1$ if $x = 1 \vee y = 1$ and 0 else, and a *crisp negation* $\ominus_c x = 0$ if $x = 1$ and 1 else.

Proposition 2. *The crisp conjunction (disjunction, negation) is a well-defined conjunction (disjunction, negation) strategy.*

For a HEX program P we define its fuzzy version P^f as follows:

Definition 3. *Let P be a HEX program. Then, P^f consists of the rules*

$$a_1 \oplus_c \ldots \oplus_c a_k \leftarrow_{\otimes_c} b_1^f \otimes_c \ldots \otimes_c b_n^f \otimes_c not_{\ominus_c}\, b_{n+1}^f \otimes_c \ldots \otimes_c not_{\ominus_c}\, b_m^f \geq 1 \quad (4)$$

for every rule of the form (1) in P, where b_i^f, $1 \leq i \leq m$, is defined such that $b_i^f = b_i$ when b_i is not external, and, if $b_i = \#g[t_1, \ldots, t_n](s_1, \ldots, s_m)$ then $b_i^f = \#g^f[t_1, \ldots, t_n](s_1, \ldots, s_m)$ where $\#g^f$ is associated with the external function $f_{\#g^f}$ that assigns, for any fuzzy interpretation I, the tuple $(I, y_1, \ldots, y_n, x_1, \ldots, x_m)$ to the value $f_{\#g}(I', y_1, \ldots, y_n, x_1, \ldots, x_m)$ where $a \in I'$ iff $I(a) = 1$, $a \in O_P$.

Proposition 3. *Let P be a HEX program. Then, M is an answer set of P iff M^f is a fuzzy answer set of P^f where $M^f : O_P \to [0, 1]$ is such that $M^f(a) = 1$ if $a \in M$ and $M^f(a) = 0$ otherwise.*

Proposition 3 shows that fuzzy HEX programs are layered on HEX programs.

Description Logic Programs (*dl-programs* for short) [11] is a formalism that allows to combine DL knowledge bases with logic programs. Roughly, in a dl-program the logic program can query the DL knowledge base, while possibly feeding deductions from the logic program as input to it. As dl-programs can be embedded in HEX programs [10], and the fuzzy rules we consider are syntactically and semantically similar in spirit as the fuzzy rules used in [12], it comes as no surprise that the so-called *fuzzy dl-programs* from [12] can be embedded in fuzzy HEX programs.

We briefly introduce fuzzy dl-programs and refer the reader for more details to [12]. A *fuzzy dl-program* (L, P) consists of a *fuzzy description logic knowledge base* L and a finite set of ground fuzzy rules P. We again refer to [12] for more details on fuzzy DLs, and retain from [12] that L comes associated with a *models* operator $\models$ such that one can express statements $L \models C(t) \geq v$ for a concept C and a term t and statements $L \models R(t_1, t_2) \geq v$ for a role R and terms t_1 and t_2; v is a value from some $[0, 1]$[2]. Intuitively, one can deduce statements from L

[2] Note that [12] restricts itself to a set $TV_n = \{0, \frac{1}{n}, \ldots, 1\}$. We will later restrict ourselves also to this set instead of considering $[0, 1]$, but for showing that fuzzy dl-programs are embedded in fuzzy HEX programs we can safely take the more general interval $[0, 1]$.

that indicate to what fuzzy degree v, the term t belongs to the concept C (or (t_1, t_2) belongs to R).

Fuzzy dl-rules in P are of the form (2) with the following modifications:

- No non-ordinary higher-order or external atoms appear in P,
- Atoms may also be *dl-atoms* $DL[S_1 \cup p_1, \ldots, S_n \cup p_n; Q](d)$, where S_i are concepts or roles, p_i are unary or binary predicates (unary if S_i is a concept, binary if S_i is a role), Q and d are either a concept and a term or a role and a pair of terms. .

For a fuzzy interpretation I of P, the value $v_I(a)$ of a ground dl-atom $a = DL[S_1 \cup p_1, \ldots, S_n \cup p_n; Q](d)$ w.r.t. L is defined as the maximum value $v \in [0, 1]$ such that $L \cup \bigcup_{i=1}^{m} A_i(I) \models Q(d) \geq v$ with $A_i(I) = \{S_i(e_i) \geq I(p_i(e_i)) \mid I(p_i(e_i)) > 0\}$ where e_i is a constant or a pair of constants depending on the arity of p_i. Intuitively, we query the DL knowledge base L where the fuzzy degrees of the concepts/roles S_i are augmented with what we know from P (i.e., via the p_i predicates) to find out what the fuzzy degree v is of membership of d in Q.

A fuzzy answer set of such a fuzzy dl-program is then defined analogous to our fuzzy answer sets where dl-atoms a have the value $v_I(a)$ w.r.t. L as defined above.

Similar as in [10], we can replace dl-atoms $a = DL[S_1 \cup p_1, \ldots, S_n \cup p_n; Q](d)$ by external atoms $\#a_L[\,](d)$ such that the associated external function $f_{\#a_L}(I, d) = v$ iff $L \cup \bigcup_{i=1}^{m} A_i(I) \models Q(d) \geq v$. For a fuzzy dl-program (L, P), let $P^{\#}$ be the program obtained from P by replacing all dl-atoms $a = DL[S_1 \cup p_1, \ldots, S_n \cup p_n; Q](d)$ by their external version $\#a_L[\,](d)$.

Proposition 4. *Let (L, P) be a fuzzy dl-program. Then, M is a fuzzy answer set of (L, P) iff M is a fuzzy answer set of $P^{\#}$.*

5 Complexity and Reasoning

We restrict ourselves in the following to the fixed set $TV_n = \{0, \frac{1}{n}, \frac{2}{n}, \ldots, 1\}$ instead of the interval $[0, 1]$, and we assume, similar as in [13], that the combination strategies are *closed* in TV_n. Note that the Lukasiewizc and Gödel combination strategies are all closed, but, for example, the production conjunction is not: on TV_3 we have that $\frac{1}{3} \otimes_P \frac{2}{3} = \frac{2}{9} \notin TV_3$. Additionally, we assume that external functions $f_{\#g}$, associated with external predicates $\#g$, are defined as functions $f_{\#g} : TV_n \to TV_n$.

For combination strategies $\otimes$ and $\ominus$, we assume the existence of external atoms $\#\otimes[X, Y](Z)$ and $\#\ominus[X](Z)$, with associated external functions to $\{0, 1\}$ defined as follows for a fuzzy interpretation I: $f_{\#\otimes}(I, X, Y, Z) = 1$ iff $X \otimes Y = Z$ and $f_{\#\ominus}(I, X, Z) = 1$ iff $\ominus X = Z$.

Additionally, we define a $\#max[X](Y)$ atom such that $f_{\#max}(I, X, Y) = 1$ if $Y = \max\{v \mid X(v) \in I\}$, i.e., Y is the maximum value that the argument of an X-atom can take.

We transform a disjunction-free fuzzy HEX program P in a HEX program P^h:

Definition 4. *Let P be a disjunction-free fuzzy* HEX *program. We take $\mathcal{C} = TV_n$. Then P^h consists of rules*

$$\sigma_a(x) \leftarrow \tag{5}$$

for each non-external atom $a \in gr(P)$[3] where $x = a$ if $a \in TV_n$ and $x = 0$ otherwise, rules

$$\sigma_a(X) \leftarrow \#g^h[y_1, \ldots, y_n](x_1, \ldots, x_m, X) \tag{6}$$

for each external atom $a = \#g[y_1, \ldots, y_n](x_1, \ldots, x_m) \in gr(P)$, where

$$f_{\#g^h}(I^h, y_1, \ldots, y_n, x_1, \ldots, x_m, x) = 1 \text{ iff } f_{\#g}(I, y_1, \ldots, y_n, x_1, \ldots, x_m) = x \ ,$$

for any interpretation I^h of P^h and I its fuzzy variant *defined such that, for a non-external atom a, $I(a) = \max\{y \mid \sigma_a(y) \in I^h\}$, and for each rule with non-empty body of the form (2) in $gr(P)$, rules*

$$\begin{aligned} \sigma_a(U_m) \ \leftarrow \\ & \sigma_{b_1}(X_1), \#max[\sigma_{b_1}](X_1), \ldots, \\ & \sigma_{b_m}(Y_m), \#max[\sigma_{b_m}](Y_m), \#\ominus_m[Y_m](X_m), \#\otimes_0[U_{m-1}, v](U_m), \\ & \#\otimes_1[X_1, X_2](U_1), \#\otimes_2[U_1, X_3](U_2), \ldots, \#\otimes_{n-1}[U_{m-2}, X_m](U_{m-1}) \end{aligned} \tag{7}$$

for rules with empty body, we introduce $\sigma_a(U_m) \leftarrow \#\otimes_0[1, v](U_m)$, and finally rules

$$\sigma_a(x - \frac{1}{n}) \leftarrow \sigma_a(x) \tag{8}$$

for all non-external atoms $a \in gr(P)$ and for all $x \neq 0 \in TV_n$.

Intuitively, the rules (5) make sure that the initial fuzzy value of a non-external atom a is equal to its fuzzy value if $a \in TV_n$ or 0 otherwise, where the value of an atom a is encoded using the binary predicate σ_a. Note that atoms from P are treated as constants in P^h (that are, however, not used to ground the transformed program, see the defintion of $\mathcal{C}$ for P^h). Similarly, the rules in (6) ensure that the values of the external atoms are correctly set. The rules in (7) compute the value of the head atom a based on the maximum values of its body atom, i.e., we assume implicitly that the actual fuzzy value of an atom is the maximum value that is present in the interpretation for that atom (using again the σ-encoding for values). We use the external atoms that correspond to the combination strategies to compute the value of the body and impose that the value of the body is equal to the value of the head, namely U_m. Note that for satisfaction of fuzzy rules the value of the head just needs to be greater than or equal the value of the body; we impose equality to ensure minimality of the fuzzy interpretation. The case where the value of an a is actually bigger than

[3] Grounding w.r.t. the original $\mathcal{C}$ of P, i.e., not w.r.t. the new $\mathcal{C} = TV_n$.

U_m is covered by the rules in (8) that also introduce any lower values for a value x.

Note that this is a different reduction than the one in [13] from fuzzy dl-programs to dl-programs, where additionally to the closedness restrictions, all combination strategies have to be the ones from Zadeh's logic (i.e., $\otimes = \min, \oplus = \max$, and $\ominus = \textit{complement}$). Our reduction is more general in this sense as it only requires closedness. However, the reduction in [13] allows for disjunctive program, which we do not handle.

We can compute fuzzy answer sets of a fuzzy HEX program by computing the answer sets of the corresponding HEX program:

Proposition 5. *Let P be a disjunction-free fuzzy* HEX *program with closed combination strategies. Then, M is a fuzzy answer set of P iff $M^h = \{\sigma(a,x) \mid M(a) = y, 0 \leq x \leq y\}$ is an answer set of P^h. Vice versa, M^h is an answer set of P^h iff M is a fuzzy answer set of P where M is defined such that $M(a) = \max\{y \mid \sigma_a(y) \in M^h\}$.*

Using the DLVHEX [9] reasoner for reasoning with HEX programs, this proposition thus gives us a method to reason with fuzzy HEX programs as well, in particular by translating them first to HEX programs. Some provisos we have to make in this respect are that the original external functions have to be computable, as well as the external functions associated with the combination strategies. Moreover, the sets of constants, variables, and external predicates under consideration should be finite in order to ensure a finite P^h (note that P itself is by definition finite).

Using the complexity results for HEX programs in [10] and the reduction of HEX programs to fuzzy HEX programs in Proposition 3 we get the following hardness results for different classes of fuzzy HEX programs.

Proposition 6. *Deciding whether a fuzzy* HEX *program without external atoms has a fuzzy answer set is $\mathsf{NEXPTIME}^{\mathsf{NP}}$-hard and $\mathsf{NEXPTIME}$-hard if the program is disjunction-free.*

For programs with external atoms $\#g$, we can deduce the same hardness results if the corresponding function $f_{\#g}$ is decidable in exponential time in $|\mathcal{C}|$.

Proposition 7. *Deciding whether a fuzzy* HEX *program, where for every $\#g \in \mathcal{G}$ the function $f_{\#g}$ is decidable in exponential time in $|\mathcal{C}|$, has a fuzzy answer set is $\mathsf{NEXPTIME}^{\mathsf{NP}}$-hard and $\mathsf{NEXPTIME}$-hard if the program is disjunction-free.*

From the complexity perspective, we even introduce in the absence of external atoms in a fuzzy HEX program P external atoms in the translation P^h to compute the combination strategies as well as the maximum value of an atom. The $\#max$ external function is not introducing extra complexity as the maximum can be calculated in linear time in the size of $gr(P)$. However, the combination strategies can add extra complexity, or lead to undecidability of checking whether there exists a fuzzy answer set in case they are undecidable - note, however, they do not depend on the program at hand. We restrict ourselves thus to *combination-computable* combination strategies:

Definition 5. *A combination strategy $\otimes$ ($\oplus$,$\ominus$) on TV_n is combination-computable if it is closed and its corresponding external function $f_{\#\otimes}$ ($f_{\#\oplus}$, $f_{\#\ominus}$) is decidable in polynomial time. We call a fuzzy* HEX *program combination-computable if its combination strategies are combination-computable.*

Note that for all the combination strategies we treated in this paper the corresponding functions are decidable in polynomial time, assuming the numbers are encoded in unary format.

Proposition 8. *Deciding whether a combination-computable disjunction-free fuzzy* HEX *program without external atoms has a fuzzy answer set is in* NEXPTIME.

Proof. The size of the program P^h is linear in the size of $gr(P)$, such that, since the size of $gr(P)$ is in general exponential in the size of P and $\mathcal{C}$, the size of P^h is exponential in the size of P and $\mathcal{C}$. Since we assume that TV_n is fixed, we get that the size of $gr(P^h)$ is polynomial in the size of P^h, and thus, the size of $gr(P^h)$ is exponential in the size of P and $\mathcal{C}$.

Using Proposition 5, checking whether there is a fuzzy answer set of a fuzzy HEX program P, amounts to checking whether there is an answer set of $gr(P^h)$, the latter can be done by a non-deterministic Turing machine in time that is polynomial in the size of $gr(P^h)$ (see, e.g., [10] and [8]). Since the size of $gr(P^h)$ is exponential in the size of P and $\mathcal{C}$, we have that checking whether there is an answer set of $gr(P^h)$ can be done by a non-deterministic Turing machine in time that is exponential in the size of P and $\mathcal{C}$, i.e., in NEXPTIME. □

Using Propositions 6 and 8, we have the following:

Corollary 1. *Deciding whether a combination-computable disjunction-free fuzzy* HEX *program without external atoms has a fuzzy answer set, is* NEXPTIME*-complete.*

External atoms in the fuzzy HEX program can introduce complexity, or even undecidability. For fuzzy HEX programs where each external predicate $\#g$ corresponds to a function $f_{\#g}$ that is decidable in a complexity class C in the size of $\mathcal{C}$, we have the following results, again using Proposition 5 and [10]:

Proposition 9. *Deciding whether a combination-computable disjunction-free fuzzy* HEX *program where each external predicate $\#g$ corresponds to a function $f_{\#g}$ that is decidable in a complexity class C in the size of $\mathcal{C}$ has a fuzzy answer set is in* NEXPTIME^C.

If we restrict ourselves to functions decidable in exponential time, we get, similar as in [10], that the exponential grounding covers for the complexity of the functions:

Proposition 10. *Deciding whether a combination-computable disjunction-free fuzzy* HEX *program where each external predicate $\#g$ corresponds to a function $f_{\#g}$ that is decidable in exponential time in the size of $\mathcal{C}$ has a fuzzy answer set is in* NEXPTIME.

Using Propositions 7 and 10, we then have the following:

Corollary 2. *Deciding whether a combination-computable disjunction-free fuzzy* HEX *program where each external predicate $\#g$ corresponds to a function $f_{\#g}$ that is decidable in exponential time in the size of $\mathcal{C}$ has a fuzzy answer set is* NEXPTIME*-complete.*

6 Applications: Service Ranking

In this section, we illustrate the use of fuzzy HEX programs to rank services. We model non-functional properties of services and user preferences as fuzzy HEX programs. For each fuzzy HEX program containing both service description and user preferences represented as rules, the ranking mechanism finds the fuzzy answer sets and their degree of fuzzy match. Based on these degrees, as a final step, the ranked list of corresponding services is constructed.

Assume a user wants to ship a package from Innsbruck to Vienna. The object to be shipped has a value of around 1000 euro according to user's estimation. The weight of the package is 3 Kg and the dimensions are 10/20/10 cm. Furthermore, the user has the following preferences: (1) he wants to pay at most around 70 euro for the service, (2) he wants to pay cash, (3) he wants the package to be ensured in case lost or damage, and (4) he expects the package to be delivered in at most around 36 hours.

Additionally, we have two shipping services *Muller* and *Runner* that potentially could satisfy the user's request. The delivery price for each of the two services depends on the weight, dimension of the package, the distance and delivery time requested by the client. The *Muller* provider presents the following conditions: (1) if the value of the package is at least around 1200 euro and the client payment method is cash the client gets at most 3% discount or free damage insurance, (2) the client has to buy both lost and damage insurances, (3) if the delivery time requested by the user is at least around 40 hours, the user gets a 2% discount from the delivery price. The *Runner* provider presents the following ones: (1) if the value of the package is at least around 1100 euro and the client payment method is cash the client gets at most 4% discount, (2) the client has to buy at least the damage insurance.

The following set of rules represent the background, shared knowledge:

$$
\begin{aligned}
distance(vienna, innsbruck, 485) &\leftarrow_{\otimes_L} \geq 1\\
hasWeight(pack, 3) &\leftarrow_{\otimes_L} \geq 1\\
hasDimension(pack, 10, 20, 10) &\leftarrow_{\otimes_L} \geq 1\\
hasValue(pack) &\leftarrow_{\otimes_L} around1000(pack) \geq 1\\
hasInsurance(package, lost, 5) &\leftarrow_{\otimes_L} \geq 0.8\\
hasInsurance(package, damage, 5) &\leftarrow_{\otimes_L} \geq 0.8\\
hasInsurance(X, full, A) &\leftarrow_{\otimes_L}\\
&\quad hasInsurance(X, lost, B)\otimes_L\\
&\quad hasInsurance(X, damage, C)\otimes_L\\
&\quad \#sum[B, C](A) \geq 1
\end{aligned}
$$

$$\begin{aligned} paymentCash &\leftarrow_{\otimes_L} \geq 1 \\ paymentCreditcard &\leftarrow_{\otimes_L} \geq 1 \\ hasPayment(X, paymentCash) & \\ \oplus_L hasPayment(X, paymentCreditcard) &\leftarrow_{\otimes_L} \geq 1 \end{aligned}$$

where $around1000 = Tri(900, 1000, 1100)$, and Tri is the triangle function specified in [14]. Note that the rules defining the insurance values in case of lost or damage package have a degree of truth of 0.8. This because, e.g., the exact insurance values are provided by third parties, such as external insurance companies, and service providers have an imprecise knowledge about these values.

The user request and preferences can be encoded as follows:

$$\begin{aligned} query(X) \leftarrow_{\otimes_L} \; & package(X) \otimes_L hasDeliveryPrice(X, P_D) \otimes_L \\ & leqAbout70(P_D) \otimes_L hasInsurance(X, full, I_F) \otimes_L \\ & hasDeliveryTime(X, T_D) \otimes_L leqAbout36(T_D) \\ & hasPayment(X, paymentCash) \geq 1 \end{aligned}$$

In the previous program, we again use a function defined in [14], namely the L-function: $leqAbout36 = L(36, 43)$ and $leqAbout70 = L(70, 75)$ to specify that the expected delivery time to be at most around 36 hours and the expected delivery price to be at most around 70 euro. The predicate *query* collects all packages that fulfill the constraints mentioned above.

The *Muller* service provider restrictions and preferences are encoded as follows:

$$\begin{aligned} discountV(X, 3) \oplus_L hasInsurance(X, damage, 0) \leftarrow_{\otimes_L} \; & around1200(X) \otimes_L \\ & hasPayment(X, paymentCash) \geq 1 \end{aligned}$$

$$\begin{aligned} discountT(X, 2) \leftarrow_{\otimes_L} \; & not_{\ominus_L} \; leqAbout40(T_D) \otimes_L \\ & hasDeliveryTime(X, T_D) \geq 1 \\ totalDiscount(X, D) \leftarrow_{\otimes_L} \; & discountV(X, B) \otimes_L discountT(X, C) \otimes_L \\ & \#sum[B, C](D) \geq 1 \\ price(X, P) \leftarrow_{\otimes_L} \; & hasWeight(X, W) \otimes_L hasDimension(X, D_L, D_W, D_H) \otimes_L \\ & distance(Start, End, Dist) \otimes_L hasDeliveryTime(X, T_D) \otimes_L \\ & \#deliveryP[W, D_L, D_W, D_H, Dist, T_D, f_M](P_D) \otimes_L \\ & \#disc[P_D, D](P_1) \otimes_L hasInsurance(X, damage, P_2) \otimes_L \\ & hasInsurance(X, lost, P_3) \otimes_L \#sum[P_1, P_2, P_3](P) \geq 1 \end{aligned}$$

The first rule contains a disjunction in the head used to specify that either a 3% discount for shipping or a free damage insurance is offered. The delivery price computation is done by an external predicate $\#deliveryP[w, dim_l, dim_w, dim_h, dis, time^{req}_{del}, f](P)$, where w is the weight of the package, $[dim_l, dim_w, dim_h]$ is the dimension of the package, dis is the distance from source to destination, $time^{req}_{del}$ is the delivery time requested by the client, f is the formula that defines the price computation and P is the computed delivery price for the package. External predicate $\#disc$ computes a discounted price given an initial price

Algorithm 1: Fuzzy Ranking

Data: Set of services S_{Ser}, User request Q, Background knowledge K, represented all as fuzzy HEX programs.
Result: Order list of services L_{Ser}.
begin

1 $\Omega \longleftarrow \emptyset$, where Ω is a set of tuples [*service*,*score*] , λ - the set of NFPs user is interested in;
2 $\beta \longleftarrow \emptyset$, is a set of quadruples [*service*,*nfp*,*nfpvalue*,*degree*];
3 **for** $s \in S_{Ser}$ **do**
4 **for** $nfp \in \lambda$ **do**
5 **if** $nfp \in s.nfps$ **then**
6 $fuzzy_{prog} = extractNfp(s, nfp) \cup K$;
7 $[s, nfp, nfpvalue, degree] \Leftarrow evaluate(fuzzy_{prog}, Q)$;
10 $\beta = \beta \cup [s, nfp, nfpvalue, degree]$;
end
else
11 $\beta = \beta \cup [s, nfp, 0, 1]$;
end
end
end
12 **for** $s \in \beta$ **do**
13 $score_s = 0$;
14 **for** $nfp \in \beta$ **do**
15 $nfpvalue = \beta.getNFPValue(s, nfp)$;
16 $nfpvalue_{max} = max(\beta.npf)$;
17 $score_s = score_s + degree * \frac{nfpvalue}{nfpvalue_{max}}$;
end
18 $\Omega = \Omega \cup [s, score_s]$;
end
19 $L_{Ser} \longleftarrow sort(\Omega)$;
end

and a discount. f_M is the formula used by service *Muller* to define how the delivery price should be computed. $around1200(X)$ is defined similarly as the other *around* predicates, i.e., $around1200(X) = Tri(1000, 1200, 1300)$.

Note that the used combination strategies used so far are Lukasiewizc strategies. However, in our example, one could have used different combination strategies, yielding different results though. The *Runner* service conditions can be encoded similarly as the *Muller* descriptions.

We assume that prior to the service ranking process a discovery process is performed. The discovery process identifies relevant services given a user request by considering semantic descriptions of functional and non-functional aspects of both services and requests. The actual ranking process is presented in Algorithm 1.

First, a fuzzy HEX program containing the background knowledge and a service non-functional property description is created for each service and each of its non-functional properties requested by the user (line 6). In the next step

(line 7), the query representing user preferences is evaluated given each program created before. The atoms representing non-functional properties of service are grounded as a result of the previous step and a degree of truth is associated with each of them. Quadruples of form $[service, nfp, nfpvalue, degree]$ are generated. If the non-functional property is not present in the service description the generated quadruple is of form $[service, nfp, 0, 1]$ - the degree of truth is 1 since we know for sure that the value of the NFP is 0 for the given service. The final part of the algorithm (line 15 - line 17) computes an aggregated score for each services, performing first a normalization of the NFPs values and incorporating the degree of truth of every ground atom (line 7). The results are collected in a set of tuples, where each tuple contains the *service id* and the *service score* (line 18). Finally, service scores are sorted and the final ranked list of services is returned (line 19).

The problems of service ranking and selection has been addressed in numerous approaches. Many of them have pointed out the need of fuzzy logic in modeling service descriptions and user preferences. For example, in [16] a fuzzy description logic approach is proposed for automating matching in e-marketplaces. In [19] multiple Quality of Service (QoS) values of services are evaluated and a fuzzy multi-attribute decision making algorithm is proposed to select the best services. The approach does not provide a flexible enough mechanism to model user preferences and services as proposed in our current work. In [20] fuzzy logic is used to evaluate the degree of matching between QoS provided by services and requested by clients. However, the approach is UDDI-based lacking sufficient expressivity for declarative reasoning with user preferences. Furthermore, none of the approaches mentioned before provides support for integration of external data sources or libraries, which is often required in real world settings.

7 Directions for Further Research

As future work, we plan to develop a reasoner for fuzzy HEX programs based on the DLVHEX reasoner, using the translation of fuzzy HEX programs to HEX programs presented in this paper. The implementation of the service ranking algorithm presented in Section 6 together with the evaluation of the approach is also left for the future.

References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook. Cambridge University Press, Cambridge (2003)
2. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge Press (2003)
3. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language Reference (2004)
4. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American, 34–43 (May 2001)

5. de Bruijn, J., Eiter, T., Polleres, A., Tompits, H.: On representational issues about combinations of classical theories with nonmonotonic rules. In: Lang, J., Lin, F., Wang, J. (eds.) KSEM 2006. LNCS (LNAI), vol. 4092, pp. 1–22. Springer, Heidelberg (2006)
6. de Bruijn, J., Lausen, H., Polleres, A., Fensel, D.: The web service modeling language: An overview. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 590–604. Springer, Heidelberg (2006)
7. Chung, L.: Non-Functional Requirements for Information Systems Design. In: Andersen, R., Solvberg, A., Bubenko Jr., J.A. (eds.) CAiSE 1991. LNCS, vol. 498, pp. 5–30. Springer, Heidelberg (1991)
8. Dantsin, E., Eiter, T., Gottlob, G., Voronkov, A.: Complexity and expressive power of logic programming. ACM Computing Surveys (CSUR) 33(3), 374–425 (2001)
9. Eiter, T., Ianni, G., Krennwallner, T., Schindlauer, R., Tompits, H.: dlvhex, `http://con.fusion.at/dlvhex/`
10. Eiter, T., Ianni, G., Schindlauer, R., Tompits, H.: A uniform integration of higher-order reasoning and external evaluations in answer-set programming. In: Proc. of IJCAI 2005 (2005)
11. Eiter, T., Lukasiewicz, T., Schindlauer, R., Tompits, H.: Combining Answer Set Programming with DLs for the Semantic Web. In: Proc. of KR 2004, pp. 141–151 (2004)
12. Lukasiewicz, T.: Fuzzy description logic programs under the answer set semantics for the semantic web. In: RULEML 2006, pp. 89–96. IEEE Computer Society Press, Los Alamitos (2006)
13. Lukasiewicz, T., Straccia, U.: Tightly integrated fuzzy description logic programs under the answer set semantics for the semantic web. Technical Report 1843-0703
14. Lukasiewicz, T., Straccia, U.: Tightly integrated fuzzy description logic programs under the answer set semantics for the semantic web. In: Marchiori, M., Pan, J.Z., de Sainte Marie, C. (eds.) RR 2007. LNCS, vol. 4524, pp. 289–298. Springer, Heidelberg (2007)
15. Preist, C.: A conceptual architecture for semantic web services. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298. Springer, Heidelberg (2004)
16. Ragone, A., Straccia, U., Bobillo, F., Di Noia, T., Di Sciascio, E., Donini, F.M.: Fuzzy description logics for bilateral matchmaking in e-marketplaces. Description Logics (2008)
17. Straccia, U.: Fuzzy Logic and the Semantic Web, ch. 4
18. Toma, I., Roman, D., Fensel, D., Sapkota, B., Gomez, J.M.: A multi-criteria service ranking approach based on non-functional properties rules evaluation. In: Krämer, B.J., Lin, K.-J., Narasimhan, P. (eds.) ICSOC 2007. LNCS, vol. 4749, pp. 435–441. Springer, Heidelberg (2007)
19. Tong, H., Zhang, S.: A fuzzy multi-attribute decision making algorithm for web services selection based on qos. In: Proc. of APSCC 2006, pp. 51–57. IEEE Computer Society Press, Los Alamitos (2006)
20. Wang, H.-C., Lee, C.-S., Ho, T.-H.: Combining subjective and objective qos factors for personalized web service selection. In: Expert Systems with Applications, pp. 571–584. Elsevier, Amsterdam (2007)