arXiv:1706.06944v3 [cs.LO] 17 Nov 2017

Accurate and Efficient Profile Matching in Knowledge Bases[☆]

Jorge Martinez Gil[a,][*], Alejandra Lorena Paoletti[a], Gábor Rácz[b], Attila Sali[b],
Klaus-Dieter Schewe[a]

[a]*Software Competence Center Hagenberg, Austria*
[b]*Alfréd Rényi Institute of Mathematics, Hungary*

Abstract

A profile describes a set of properties, e.g. a set of skills a person may have, a set of skills required for a particular job, or a set of abilities a football player may have with respect to a particular team strategy. Profile matching aims to determine how well a given profile fits to a requested profile and vice versa. The approach taken in this article is grounded in a matching theory that uses filters in lattices to represent profiles, and matching values in the interval [0,1]: the higher the matching value the better is the fit. Such lattices can be derived from knowledge bases exploiting description logics to represent the knowledge about profiles. An interesting first question is, how human expertise concerning the matching can be exploited to obtain most accurate matchings. It will be shown that if a set of filters together with matching values by some human expert is given, then under some mild plausibility assumptions a *matching measure* can be determined such that the computed matching values preserve the relevant rankings given by the expert. A second question concerns the efficient querying of databases of profile instances. For *matching queries* that result in a ranked list of profile instances matching a given one it will be shown how corresponding top-k queries can be evaluated on grounds of pre-computed matching values, which in turn allows the maintenance of the knowledge base to be decoupled from the maintenance of profile instances. In addition, it will be shown how the matching queries can be exploited for *gap queries* that determine how profile

---

instances need to be extended in order to improve in the rankings. Finally, the theory of matching will be extended beyond the filters, which lead to a matching theory that exploits fuzzy sets or probabilistic logic with maximum entropy semantics.

---

1. Introduction

A profile describes a set of properties, and profile matching is concerned with the problem to determine how well a given profile fits to a requested one. Profile matching appears in many application areas such as matching applicants for jobs to job requirements, matching system configurations to requirements specifications, matching team players to game strategies in sport, etc.

In order to support profile matching by knowledge-based tools the first question concerns the representation of the profiles. For this one might use just sets, e.g. the set of skills of a person could contain "knowledge of Java", "knowledge of parsing theory", "knowledge of Italian", "team-oriented person", etc. In doing so, profile matching would have to determine measures for the difference of sets. Such approaches exist, but they will usually lead only to very coarse-grained matchings. For instance, "knowledge of Java" implies "knowledge of object-oriented programming", which further implies "knowledge of programming". Thus, at least hierarchical dependencies between the terms in a profile should be taken into account, which is the case in many taxonomies for skills. Mathematically it seems appropriate to exploit partially-ordered sets or better lattices to capture such dependencies, which justifies to consider not just sets, but filters in lattices as an appropriate way of representing profiles.

However, other more general dependencies would still not be captured. For instance, the "knowledge of Java" may be linked to "two years of experience" and "application programming in web services". If such a fine-grained characterisation of skills is to be captured as well, the well established field of knowledge representation offers solutions in form of ontologies, which formally can be covered by description logics. This further permits a thorough distinction between the terminological level (aka TBox) of a knowledge base capturing abstract concepts such as "knowledge of Java" or "web services" and their relationships, and corresponding assertional knowledge (aka ABox) referring to particular instances. For example, the ABox may contain the skills of particular persons such as Lorena or Jorge linking them to "Lorena's knowledge of Java" or "Jorge's knowledge of Java", which further link to "ten years of experience" or "six years of experience", respectively. Therefore, it appears appropriate to assume that a knowledge base is used for the representation of the knowledge about abstract and concrete terms appearing in profiles.

For profile matching this raises the question, how the extended relationships could be integrated into profiles. Pragmatically, it appears justified to assume

that the knowledge base is finite. For instance, for recruiting a distinction between "n years of experience" only makes sense for positive integer values for n up to some maximum. Also, a classification of application areas will only require finitely many terms. As the relationships in a knowledge base correspond semantically to binary relations, we can exploit inverse images and thus exploit concepts such as "knowledge of Java with n years of experience" for different possible values of n—of course, the concept with a larger value of n subsumes the concept with a smaller value"—to derive a sophisticated lattice from the knowledge base. Even the knowledge of the actual instances in the ABox can be exploited to keep the derived lattice reasonably small.

As profiles can be represented by filters in lattices, profile matching can be approached by appropriate *matching measures* μ, i.e. we assign to each pair of filters a numeric *matching value* $\mu(\mathcal{F}_1, \mathcal{F}_2)$, which we can normalise to be in the interval $[0, 1]$. The meaning should be the degree to which the profile represented by $\mathcal{F}_1$ fits to the profile represented by $\mathcal{F}_2$, such that a higher matching value corresponds to a better fit. Naturally, if a given profile contains everything that is requested through another profile, the given profile should be considered a perfect fit and receive a matching value of 1. This implies that the matching measure cannot be symmetric. For instance, almost everyone would satisfy the requirements for a position of "receptionist", but it is very unlikely that for such a position a PhD-qualified candidate would be selected. This is, because the inverted matching measure is low, i.e. the requested profile for the receptionist does not fit well to the profile of the PhD-qualified candidate, in other words, s/he is considered to be over-qualified. In this way the matching measures can be handled flexibly to capture also concepts such as over-qualification or matching by means of multiple criteria, e.g. technical skills and social skills.

1.1. Our Contribution

In this paper we develop a theory of profile matching, which is inspired by the problem of matching in the recruiting domain [1] but otherwise independent from a particular application domain. As argued above our theory will be based on profiles that are defined by filters in appropriate lattices [2]. We will show that information represented in knowledge bases using highly expressive description logics similar to DL-LITE [3], $\mathcal{SROIQ}$ [4] or OWL-2 [5] can be captured adequately by filters in lattices that are derived from the TBox of the knowledge base. For a matching measure we then assign weights to the elements of the lattice, where the weighting function should satisfy some normalising constraints. This will be exploited in the definition of asymmetric *matching measures*. We argue that every matching measure can be obtained by such weights.

While these definitions constitute a justified contribution to formalise profile matching, we will investigate how matching measures can be maintained in the light of human expertise. If bias (i.e. matching decisions that are grounded in concepts not appearing in the knowledge base) can be excluded, the question is, if human-made matchings can always be covered by an appropriate matching measure. As human experts rather use rankings than precise matching values,

we formalise this problem by a notion of *ranking-preserving matching measure*. Our first main result is that under some mild assumptions—the satisfaction of plausibility constraints by the human-made matchings—a ranking-preserving matching measure can always be determined. The proof of this result requires to show the solvability of some linear inequalities. However, we also show that in general, not all relationships in human-made matchings can be preserved in a matching measure.

Our second main result shows how *matching queries* can be efficiently implemented. As matchings are usually done to determine a ranked list of matching profiles for a fixed profile—in recruiting this usually refers to the shortlisting procedure—or conversely a ranked list of profiles for which a given profile may fit, and only the k top-ranked profile instances are considered to be relevant (for some user-defined integer constant k), the problem boils down to establish an efficient implementation for top-k queries. The naive approach to first evaluate a query that determines a ranked list of profile instances, from which the "tail" starting with the $k + 1$st element is thrown away, would obviously be highly inefficient. In addition, in our case the ranking criterium is based on the matching values, which themselves require a computation on the basis of filters, so it is desirable to minimise the number of such computations [6]. Therefore, we favour an approach that separates the profiles that are determined by the underlying lattice and thus do not change very often from the profile instances in a concrete matching query, e.g. the information about current applicants for an open position. Note that there may be several such instances associated with the same profile. Then we can create data structures around pre-computed matching values. The number of such pre-computed values is far less than the number of profile pairs, and most importantly, the data structure can exploit that only the larger ones of these values will be relevant for the matching queries. Based on these ideas we contribute an algorithm for the efficient evaluation of top-k matching queries.

In addition, we take this approach to querying further to support also *gap queries*, which determine for a given profile extensions that improve the rankings for the given profile with respect to possible requested profiles. In the recruitment domain such gap queries are of particular interest for job agents, who can use the results to recommend additional training to candidates that would otherwise have low chances on the job market. Furthermore, for educational institutions the results of such queries give information about the needed qualifications that should be targeted by training courses.

Finally, we investigate further extensions to the matching theory with respect to relations between the concepts in a profile that are not covered by ontologies. In particular, the presence of a particular concept in a profile may only partially imply the presence of another concept. For instance, "knowledge of Java" and "knowledge of NetBeans" may be unrelated in a knowledge base, yet with a degree (or probability) of 0.7 the former one may imply the latter one. We therefore explore additional links between the elements of the lattice that are associated with a degree (or probability); even cycles will be permitted. We contribute an *enriched matching theory* by means of values associated to paths

4

[7]. Regarding the values associated with the added links as probabilities suggests a different approach that exploits *probabilistic logic programs*, for which we exploit the maximum entropy semantics, which interprets the additional links as adding minimal additional information. We show that matching values are the result of probabilistic queries that are obtained from sentences determined by the extended links.

1.2. Related Work

Knowledge representation is a well established branch of Artificial Intelligence. In particular, description logics have been in the centre of interest, often as the strict formal counterpart of the more vaguely used terms "ontology" or "semantic technology". At its core a description logic can be seen as a fragment of first-order logic with unary and binary predicates with a decidable implication. A TBox captures terminological knowledge, i.e. well-formed sentences (implications) of the logic, while an ABox captures instances, i.e. assertional knowledge. Many different description logics have been developed, which differ by their expressiveness (see [4] for a survey). The DL-LITE family provides a collection of description logics that appear to be mostly sufficient for our purposes, though we will also exploit partly constructs from the highly expressive description logic is $\mathcal{SROIQ}$ to highlight the relationship between knowledge representation and profile matching.

Description logics have been used in many application branches, in particular as a foundation for the semantic web [8] and for knowledge sharing [9]. For the usage in the context of the semantic web the language OWL-2, which is essentially equivalent to $\mathcal{SROIQ}(D)$, has become a standard. Ontologies have also been used in the area of recruiting in connection with profile matching (see [10] for a survey). However, while it is claimed that matching accuracy can be improved [11], the matching approach itself remains restricted to Boolean matching, which basically means to count how many requested skills also appear in a given profile [12]. Surprisingly, sophisticated taxonomies in the recruitment domain such as DISCO [13], ISCO [14] and ISCED [15] have not yet been properly linked with the much more powerful and elegant languages for knowledge representation.

With respect to foundations of a profile matching theory we already argued that it is not appropriate to define profiles just as sets of unrelated items, even though many distance measures for sets such as Jaccard or Sørensen-Dice have proven to be useful in ecological applications [16]. The first promising attempt to take hierarchical dependencies into account was done by Popov and Jebelean [17], which defines the initial filter-based measure. However, weights are not used, only cardinalities, which correspond to the special case that all concepts are equally weighted. Our matching theory is inspired by this work, but takes the filter-based approach much farther. To our best knowledge no other approach in this direction has been tried.

With respect to the analysis of matching measures, in particular in connection with human-defined matchings it is tempting to exploit ontology learning

[18] or formal concept analysis (FCA) [19, 20]. FCA has been exploited successfully in many areas, also in combination with ontologies [21] (capturing structure) and rough sets [22] (capturing vagueness). A first attempt to exploit formal concept analysis for the learning of matching measures has been reported in [23]. However, it turned out that starting from matching relations, the derived concept lattices still require properties to be examined that are expressed in terms of the original relation, so the ideas to exploit formal concept analysis were abandoned. Regarding ontology learning a first application to e-recruitment has been investigated in [24]. The resulting learning algorithms can be combined with our results on the existence of ranking-preserving matching measures. Remotely related to our objective to determine matching measures that are in accordance with human-made matchings is the research on human preferences in ranking [25] and on product ranking with user credibility [26].

Top-k queries have attracted a lot of attention in connection with ranking problems. Usually, they are investigated in the context of relational databases [27], and the predominant problem associated with them is efficiency [28]. This is particularly the case for join queries [29] or for the determination of a dominant query [30]. Extensions in the direction of fuzzy logic [31] and probabilities have also been tried [32]. For our purposes here, most of the research on top-k query processing is only marginally relevant, because it is not linked to the specific problem of finding the best matches. On one hand this brings with it the additional problem that the matching values need to be computed as well, but on the other hand permits a simplification, as the possible matching values do not frequently change.

Concerning enriched matching with fuzzy degrees seems at first sight to lead to the NP-complete problem of finding longest paths in a weighted directed graph, but in our case the weighting is multiplicative with values in [0,1]. This enables an interpretation using fuzzy filters [33]. For the probabilistic extension our research will be based on the probabilistic logic with maximum entropy semantics in [34, 35], for which sophisticated reasoning methods exist [36].

1.3. Organisation of the Article

The remainder of this article is organised as follows. In Section 2 we introduce fundamentals from knowledge representation with description logics. We particularly emphasise the features in the description logics DL-LITE and $\mathcal{SROIQ}$ without adopting them completely. Actually, we leave it to the particular application domain to decide, which knowledge representation is most appropriate. However, we require that such a knowledge base gives rise to a lattice that captures the information found in profiles. We show how the roles give rise to particular subconcepts and thus can be omitted from further consideration. Thus we show how we can obtain a lattice that is needed for our matching theory from a knowledge base that captures general terminology of an application domain. For instance, we envision that on one hand for recruiting an extension of the various taxonomies such as ISCO, DISCO and ISCED perfectly makes sense even without this connection to matching, while on the other hand matching has to exploit the available knowledge sources.

In Section 3 we introduce the fundamentals of our approach to profile matching. We start with profiles defined by filters in lattices and define weighted *matching measures* on top of them. Naturally, the lattices are derived from knowledge bases as discussed in Section 2. We further discuss the lattice of *matching value terms*, which symbolically characterise possible matching values.

Section 4 is then dedicated to the relationship between the filter-based matching theory and matchings by human experts. That is, the section is dedicated to the problem to determine, if and how a matching measure as defined in Section 3 can be obtained from human-defined matching values. For this we first derive plausibility constraints that human-made matching should fulfil in order to exclude unjustified bias. We then show that if the plausibility constraints are satisfied, weights can be defined in such a way that particular rankings based on the corresponding matching measure coincide with the human-made ones. The rankings we consider are restricted to either the same requested profile or the same given profile plus requested profiles in the same relevance class. This permits minimum updates to existing matching measures in order to establish compliance with human expertise.

In Section 5 the issue of queries is addressed, which concerns the implementation of the matching theory from Section 3. First we present an approach to implement top-k queries for matching, which extend naturally to matching queries under multiple criteria, e.g. capturing over-qualification. The second class of queries investigated concerns gaps, i.e. possible enlargements of profiles that lead to better rankings of a profile instance.

Section 6 is dedicated to enriched matchings that exploit additional links between concepts in the knowledge base. This takes the matching theory from Section 3 further. First we discuss maximum length matching, which is based on a fuzzy set interpretation of such links. Second, we discuss an interpretation in probabilistic logic with maximum entropy semantics. Naturally, for the enriched matching theory it would be desirable to address again the issues of preservation of human-defined rankings and efficient querying, but these topics are still under investigation and thus left out of this article.

Finally, we conclude the article in Section 7 with a brief summary and discussion of open questions that need to be addressed to apply our matching theory in practice.

2. Profiles and Knowledge Bases

This section is dedicated to a brief introduction of our understanding of knowledge bases that form the background for our approach to profile matching. In the introduction we already outlined that we consider a profile to describe a set of properties, and that dependencies between such properties should be taken into account. Therefore, our proposal is to exploit description logics for the representation of domain knowledge. Thus, for the representation of knowledge we adopt the fundamental distinction between *terminological* and *assertional* knowledge that has been used in description logics since decades. In

accordance with basic definitions about description logics [4, pp. 17ff.] for the former one we define a language, which defines the TBox of a knowledge base, while the instances define corresponding ABoxes.

A TBox consists of concepts, roles and constraints on them. The description logic used here is derived from DL-LITE [3] and $\mathcal{SROIQ}$ [4], which we use as role models for the features that in many application domains need to be supported.

$\mathcal{S}$ stands for the presence of a top concept $\top$, a bottom concept $\bot$, intersection $C_1 \sqcap C_2$, union $C_1 \sqcup C_2$, and for concepts $\forall R.C$ and $\exists R.C$ (the semantics of these will be defined later).

$\mathcal{R}$ stands for role chains $R_1 \circ R_2$ and role hierarchies $R_1 \sqsubseteq R_2$ (the latter ones we do not need).

$\mathcal{O}$ stands for nominals, i.e. we provide individual I_0 and permit to use concepts of the form $\{a\}$. Then in combination with $\mathcal{S}$ also enumerations $\{a_1, \ldots, a_n\} = \{a_1\} \sqcup \cdots \sqcup \{a_n\}$ are enabled.

$\mathcal{I}$ stands for inverse atomic roles R_0^-.

$\mathcal{Q}$ stands for quantified cardinality restrictions $\geq m.R.C$ and $\leq m.R.C$ (the semantics of these will be defined later).

We believe that in most application domains—definitely for job recruiting—it is advisable to exploit these features in a domain knowledge base, except role hierarchies and inverse roles. Therefore, let us assume that C_0, I_0 and R_0 represent not further specified sets of basic concepts, individuals and roles, respectively. Then *atomic concepts A*, *concepts C* and *roles R* are defined by the following grammar:

$$
\begin{aligned}
R &= R_0 \mid R_1 \circ R_2 \\
A &= C_0 \mid \top \mid \geq m.R.C \text{ (with } m > 0) \mid \{I_0\} \\
C &= A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C
\end{aligned}
$$

Definition 2.1. A *TBox* $\mathcal{T}$ comprises concepts C and roles R as defined by the grammar above plus a finite set of constraints of the form $C_1 \sqsubseteq C_2$ with concepts C_1 and C_2.

Each assertion $C_1 \sqsubseteq C_2$ in a TBox $\mathcal{T}$ is called a *subsumption axiom*. Note that Definition 2.1 only permits subsumption between concepts, not between roles, though it is possible to define more complex terminologies that also permit role subsumption (as in $\mathcal{SROIQ}$). As usual, we use several shortcuts: (1) $C_1 \equiv C_2$ can be used instead of $C_1 \sqsubseteq C_2 \sqsubseteq C_1$, (2) $\bot$ is a shortcut for $\neg\top$, (3) $\{a_1, \ldots, a_n\}$ is a shortcut for $\{a_1\} \sqcup \cdots \sqcup \{a_n\}$, (4) $\leq m.R.C$ is a shortcut for $\neg \geq m+1.R.C$, and (5) $= m.R.C$ is a shortcut for $\geq m.R.C \sqcap \leq m.R.C$.

Example 2.1. With a skill "programming" we would like to associate other properties such as "years of experience", "application area", and "degree of

complexity", which defines a complex aggregate structure. In a TBox this may lead to subsumption axioms such as

$$\text{programming} \sqsubseteq \exists \text{experience}.\{1, \ldots, 10\}$$
$$\text{programming} \sqsubseteq \exists \text{area}.\{\text{business}, \text{science}, \text{engineering}\} \text{ and}$$
$$\text{programming} \sqsubseteq \exists \text{complexity}.\{1, \ldots, 5\}$$

Obviously, the concepts in a TBox define a lattice $\mathcal{L}$ with $\sqcap$ and $\sqcup$ as operators for meet and join, and $\sqsubseteq$ for the partial order. For our purpose of matching we are particularly interested in named concepts, i.e. we assume that for each concept C as defined by the grammar above we also have a constraint $C_0 \equiv C$ with some atomic concept name in C_0. Then we can identify the elements of $\mathcal{L}$ with the names in C_0. In particular, in order to exploit the roles as in Example 2.1 we consider "blow-up" concepts [1] that have the form $C \sqcap \exists R.C''$, where C is a concept, for which $C \sqsubseteq \exists R.C'$ holds and $C'' \sqsubseteq C'$ holds. In particular, this becomes relevant, if C'' is defined by individuals, say $C'' = \{a_1, \ldots, a_n\}$.

Using such blow-up concepts, we can express concepts such as "programming of complexity level 4 in science with at least 3 years of experience". We tacitly assume that for matching we exploit a lattice that is defined by a TBox exploiting blow-up concepts as well as $\sqcap$, $\sqcup$ and $\sqsubseteq$.

Definition 2.2. A *structure* $\mathcal{S}$ for a TBox $\mathcal{T}$ consists of a non-empty set $\mathcal{O}$ together with subsets $\mathcal{S}(C_0) \subseteq \mathcal{O}$ and $\mathcal{S}(R_0) \subseteq \mathcal{O} \times \mathcal{O}$ for all basic concepts R_0 and basic roles R_0, respectively, and individuals $\bar{a} \in \mathcal{O}$ for all $a \in I_0$. $\mathcal{O}$ is called the base set of the structure.

We first extend the interpretation of basic concepts and roles and to all concepts and roles as defined by the grammar above, i.e. for each concept C we define a subset $\mathcal{S}(C) \subseteq \mathcal{O}$, and for each role R we define a subset $\mathcal{S}(R) \subseteq \mathcal{O} \times \mathcal{O}$ as follows:

$$\mathcal{S}(R_1 \circ R_2) = \{(x, z) \mid \exists y.(x, y) \in \mathcal{S}(R_1) \wedge (y, z) \in \mathcal{S}(R_2)\}$$
$$\mathcal{S}(\top) = \mathcal{O} \qquad \mathcal{S}(\{a\}) = \{\bar{a}\} \qquad \mathcal{S}(\neg C) = \mathcal{O} - \mathcal{S}(C)$$
$$\mathcal{S}(\geq m.R.C) = \{x \in \mathcal{O} \mid \#\{y \mid (x, y) \in \mathcal{S}(R) \wedge y \in \mathcal{S}(C)\} \geq m\}$$
$$\mathcal{S}(C_1 \sqcap C_2) = \mathcal{S}(C_1) \cap \mathcal{S}(C_2) \qquad \mathcal{S}(C_1 \sqcup C_2) = \mathcal{S}(C_1) \cup \mathcal{S}(C_2)$$
$$\mathcal{S}(\exists R.C) = \{x \in \mathcal{O} \mid (x, y) \in \mathcal{S}(R) \text{ for some } y \in \mathcal{S}(C)\}$$
$$\mathcal{S}(\forall R.C) = \{x \in \mathcal{O} \mid (x, y) \in \mathcal{S}(R) \Rightarrow y \in \mathcal{S}(C) \text{ for all } y\}$$

In doing so, we can consider concepts as predicate symbols C of arity 1 in first-order logic, and roles as predicate symbols of arity 2. Then the extension of the structure defines a set of ground instances of the form $C(a)$ and $R(a, b)$. An *ABox* for a TBox $\mathcal{T}$ is usually defined as a finite set of such ground atoms. Thus, an ABox is de facto defined by a structure, for which in addition we usually assume consistency.

Definition 2.3. A structure $\mathcal{S}$ for a TBox $\mathcal{T}$ is *consistent* if $\mathcal{S}(C_1) \subseteq \mathcal{S}(C_2)$ holds for all assertions $C_1 \sqsubseteq C_2$ in $\mathcal{T}$.

For the following we always consider a concept C in a TBox as representation of abstract properties, e.g. "knowledge of Java", and individuals in the ABox as concrete properties such as the "Java knowledge of Lisa".

Definition 2.4. Given a finite consistent structure, a *profile* P is a subset of the base set $\mathcal{O}$. The *representing filter* of a profile P is the filter $\mathcal{F}(P) \subseteq \mathbb{F}$ of the lattice $\mathcal{L}$ defined by the TBox with $\mathcal{F}(P) = \{C \in \mathcal{L} \mid \exists p \in P.\ p \in \mathcal{S}(C)\}$.

3. Weighted Profile Matching Based on Lattices and Filters

In this section we present the formal definitions underlying our approach to profile matching. In the previous section we have seen how to obtain lattices from knowledge bases. Our theory is therefore based on such lattices $\mathcal{L}$. We have further seen that a profile, understood as a set P of concept instances in an ABox, always gives rise to a representing filter $\mathcal{F}$ in the lattice $\mathcal{L}$. Therefore, our theory exploits filters in lattices. We first define the notion of a *matching measure* as a function defined on pairs of filters. If μ is such a matching measure and $\mathcal{F}, \mathcal{G}$ are filters, the $\mu(\mathcal{F}, \mathcal{G})$ will be a real number in the interval $[0,1]$, which we will call a *matching value*. Matching measures will exploit weights assigned to concepts in the lattice $\mathcal{L}$. Finally in this section we discuss a specific lattice, the lattice of *matching value terms* that is defined by the matching measures.

3.1. Filter-Based Matching

As stated above, profiles are to describe sets of properties, and profile matching should result in values that determine how well a given profile fits to a requested one, so we base our theory on lattices. Throughout this section let $(\mathcal{L}, \leq)$ be a lattice. Informally, for $A, B \in \mathcal{L}$ we have $A \leq B$, if the property A subsumes property B, e.g. for skills this means that a person with skill A will also have skill B.

Definition 3.1. A *filter* is a non-empty subset $\mathcal{F} \subseteq \mathcal{L}$, such that for all C, C' with $C \leq C'$ whenever $C \in \mathcal{F}$ holds, then also $C' \in \mathcal{F}$ holds.

We concentrate on filters $\mathcal{F}$ in order to define matching measures.

Definition 3.2. Let $\mathbb{F} \subseteq \mathcal{P}(\mathcal{L})$ denote the set of filters. A *weighting function* on $\mathcal{L}$ is a function $w : \mathcal{P}(\mathcal{L}) \to [0,1]$ satisfying

(1) $w(\mathcal{L}) = 1$, and
(2) $w(\bigcup_{i \in I} A_i) = \sum_{i \in I} w(A_i)$ for pairwise disjoint A_i $(i \in I)$.

Definition 3.3. A *matching measure* is a function $\mu : \mathbb{F} \times \mathbb{F} \to [0,1]$ such that $\mu(\mathcal{F}_1, \mathcal{F}_2) = w(\mathcal{F}_1 \cap \mathcal{F}_2)/w(\mathcal{F}_2)$ holds for some weighting function w on $\mathcal{L}$.

Example 3.1. Take a simple lattice $\mathcal{L}$ with only five elements: $\mathcal{L} = \{C_1, C_2, C_3, C_4, C_5\}$. The lattice structure is shown in Figure 3.1. Then we obtain seven filters for this lattice, each generated by one or two elements of the lattice. These filters are also shown in Figure 3.1.
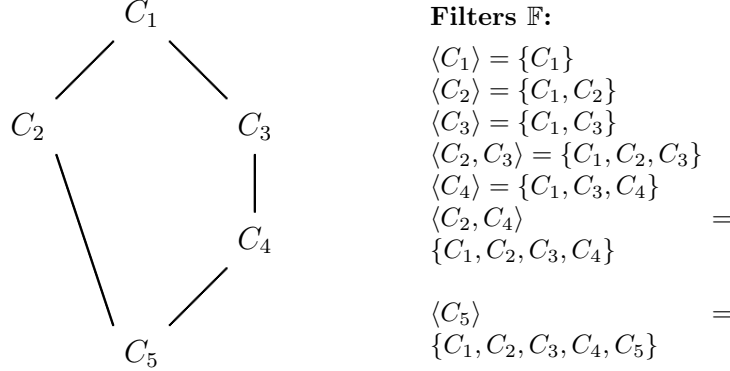
C_1

C_2 C_3

C_4

C_5

Filters $\mathbb{F}$:

$\langle C_1 \rangle = \{C_1\}$
$\langle C_2 \rangle = \{C_1, C_2\}$
$\langle C_3 \rangle = \{C_1, C_3\}$
$\langle C_2, C_3 \rangle = \{C_1, C_2, C_3\}$
$\langle C_4 \rangle = \{C_1, C_3, C_4\}$
$\langle C_2, C_4 \rangle = \{C_1, C_2, C_3, C_4\}$

$\langle C_5 \rangle = \{C_1, C_2, C_3, C_4, C_5\}$

Figure 3.1: A simple lattice and its filters

	$\langle C_1 \rangle$	$\langle C_2 \rangle$	$\langle C_3 \rangle$	$\langle C_2, C_3 \rangle$	$\langle C_4 \rangle$	$\langle C_2, C_4 \rangle$	$\langle C_5 \rangle$
$\langle C_1 \rangle$	1	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{9}$	$\frac{1}{10}$
$\langle C_2 \rangle$	1	1	$\frac{1}{3}$	$\frac{2}{3}$	$\frac{1}{6}$	$\frac{4}{9}$	$\frac{2}{5}$
$\langle C_3 \rangle$	1	$\frac{1}{4}$	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{3}{10}$
$\langle C_2, C_3 \rangle$	1	1	1	1	$\frac{1}{2}$	$\frac{2}{3}$	$\frac{3}{5}$
$\langle C_4 \rangle$	1	$\frac{1}{4}$	1	$\frac{1}{2}$	1	$\frac{2}{3}$	$\frac{2}{5}$
$\langle C_2, C_4 \rangle$	1	1	1	1	1	1	$\frac{9}{10}$
$\langle C_5 \rangle$	1	1	1	1	1	1	1

Table 1: A matching measure μ on the lattice $\mathcal{L}$

If we now define weights $w(C_1) = \frac{1}{10}$, $w(C_2) = \frac{3}{10}$, $w(C_3) = \frac{1}{5}$, $w(C_4) = \frac{3}{10}$, $w(C_5) = \frac{1}{10}$, then we obtain the matching measure values $\mu(\mathcal{F}, \mathcal{G})$ shown in Table 1. In the table the row label is $\mathcal{F}$ and the column label is $\mathcal{G}$.

Obviously, every matching measure μ is defined by weights $w(C) = w(\{C\}) \in [0,1]$ for the elements $C \in \mathcal{L}$. With this we immediately obtain $w(\mathcal{F}) = \sum_{C \in \mathcal{F}} w(C)$ and $w(\mathcal{L} - \mathcal{F}) = 1 - w(\mathcal{F})$. Then $\mu(\mathcal{F}_1, \mathcal{F}_2) = \sum_{C \in \mathcal{F}_1 \cap \mathcal{F}_2} w(C) \cdot \left(\sum_{C \in \mathcal{F}_2} w(C) \right)^{-1}$ expresses, how much of (requested) profile represented by $\mathcal{F}_2$ is contained in the (given) profile represented by $\mathcal{F}_1$.

Example 3.2. The matching measure μ_{pj} defined in [17] uses simply cardinalities:

$$\mu_{pj}(\mathcal{F}_1, \mathcal{F}_2) = \#(\mathcal{F}_1 \cap \mathcal{F}_2)/\#\mathcal{F}_2$$

Thus, it is defined by the weighting function w on $\mathcal{L}$ with $w(A) = \#A/\#\mathcal{L}$, i.e. all properties have equal weights.

Note that in general matching measures are not symmetric. If $\mu(\mathcal{F}_1, \mathcal{F}_2)$ expresses how well a given profile (represented by $\mathcal{F}_1$) matches a requested

	$\langle C_1\rangle$	$\langle C_2\rangle$	$\langle C_3\rangle$	$\langle C_2,C_3\rangle$	$\langle C_4\rangle$	$\langle C_2,C_4\rangle$	$\langle C_5\rangle$
$\langle C_1\rangle$	1	$\frac{a}{a+b}$	$\frac{a}{a+c}$	$\frac{a}{a+b+c}$	$\frac{a}{a+c+d}$	$\frac{a}{a+b+c+d}$	$\frac{a}{a+b+c+d+e}$
$\langle C_2\rangle$	1	1	$\frac{a}{a+c}$	$\frac{a+b}{a+b+c}$	$\frac{a}{a+c+d}$	$\frac{a+b}{a+b+c+d}$	$\frac{a+b}{a+b+c+d+e}$
$\langle C_3\rangle$	1	$\frac{a}{a+b}$	1	$\frac{a+c}{a+b+c}$	$\frac{a+c}{a+c+d}$	$\frac{a+c}{a+b+c+d}$	$\frac{a+c}{a+b+c+d+e}$
$\langle C_2,C_3\rangle$	1	1	1	1	$\frac{a+c}{a+c+d}$	$\frac{a+b+c}{a+b+c+d}$	$\frac{a+b+c}{a+b+c+d+e}$
$\langle C_4\rangle$	1	$\frac{a}{a+b}$	1	$\frac{a+c}{a+b+c}$	1	$\frac{a+c+d}{a+b+c+d}$	$\frac{a+c+d}{a+b+c+d+e}$
$\langle C_2,C_4\rangle$	1	1	1	1	1	1	$\frac{a+b+c+d}{a+b+c+d+e}$
$\langle C_5\rangle$	1	1	1	1	1	1	1

Table 2: Matching measure terms for the lattice $\mathcal{L}$

profile (represented by $\mathcal{F}_2$), then $\mu(\mathcal{F}_2, \mathcal{F}_1)$ measures what is "too much" in the given profile that is not required in the requested profile.

Example 3.3. Take the lattice $\mathcal{L}$ from Example 3.1 shown in Figure 3.1. Let $\mathcal{G} = \langle C_2, C_3\rangle$ be the filter that represents the requirements. If we take filters $\mathcal{F}_1 = \langle C_3\rangle$ and $\mathcal{F}_2 = \langle C_4\rangle$ representing given profiles, then we have $\mu(\mathcal{F}_1, \mathcal{G}) = \mu(\mathcal{F}_2, \mathcal{G}) = \frac{1}{2}$, i.e. both given filters match the requirements equally well. However, if we also consider the *inverse matching measure* $\bar{\mu}$ with values $\bar{\mu}(\mathcal{F}_1, \mathcal{G}) = \mu(\mathcal{G}, \mathcal{F}_1) = 1$ and $\bar{\mu}(\mathcal{F}_2, \mathcal{G}) = \mu(\mathcal{G}, \mathcal{F}_2) = \frac{1}{2}$, then we see that $\mathcal{F}_1$ matches "better", as $\mathcal{F}_2$ contains C_4, which is not required at all.

The example shows that it may make sense to consider not just a single matching measure μ, but also its inverse—In the recruiting area this corresponds to "over-qualification" [1]—$\bar{\mu}$, several matching measures defined on different lattices, or weighted aggregates of such measures.

3.2. The Lattice of Matching Value Terms

We know that a matching measure μ on $\mathbb{F}$ is defined by weights $w(C)$ for each concept $C \in \mathcal{L}$, i.e.

$$\mu(\mathcal{F}, \mathcal{G}) = \frac{\sum_{C \in \mathcal{F} \cap \mathcal{G}} w(C)}{\sum_{C \in \mathcal{G}} w(C)}$$

with $w(C) > 0$. In case $\mathcal{G} \subseteq \mathcal{F}$, the right hand side becomes 1. In all other cases the actual value on the right hand side depends on the weights $w(C)$. Let us call the right hand side expression (including 1) a *matching value term* (mvt).

Example 3.4. Let us look at the lattice $\mathcal{L}$ from Example 3.1. With $w(C_1) = a$, $w(C_2) = b$, $w(C_3) = c$, $w(C_4) = d$ and $w(C_5) = e$ we obtain the matching value terms shown in Table 2.

Definition 3.4. Let $\mathbb{V}$ denote the set of all matching value terms (including 1) for the lattice $\mathcal{L}$. A partial order $\leq$ on $\mathbb{V}$ is defined by $v_1 \leq v_2$ iff each substitution of positive values for $w(C)$ results in values $\bar{v}_1, \bar{v}_2 \in [0, 1]$ with $\bar{v}_1 \leq \bar{v}_2$.

Let us first look at the following three special cases:

(1) If v_2 differs from v_1 by a summand $w(C)$ added to the nominator, i.e. $\mathcal{F}_2 = \mathcal{F}_1 \cup \{C\}$ and $\mathcal{G}_2 = \mathcal{G}_1$, then we obviously obtain $v_1 \leq v_2$.
(2) If v_1 differs from v_2 by a summand $w(C)$ added to the denominator, i.e. $\mathcal{G}_1 = \mathcal{G}_2 \cup \{C\}$ and $\mathcal{F}_2 = \mathcal{F}_1$, then we obviously obtain $v_1 \leq v_2$.
(3) If v_2 differs from v_1 by a summand $w(C)$ added to both the nominator and the denominator, i.e. $\mathcal{G}_2 = \mathcal{G}_1 \cup \{C\}$ and $\mathcal{F}_2 = \mathcal{F}_1 \cup \{C\}$, then we also obtain $v_1 \leq v_2$.

Lemma 3.1. *The partial order $\leq$ on the set $\mathbb{V}$ of matching value terms is the reflexive, transitive closure of the relation defined by the cases (1), (2) and (3) above.*

Proof. Let $v_i = \dfrac{\sum_{C \in \mathcal{F}_i} w(C)}{\sum_{C \in \mathcal{G}_i} w(C)}$ with $\mathcal{F}_i \subseteq \mathcal{G}_i$ $(i = 1, 2)$. If v_2 results from v_1 by a sequence of the operations (1), (2) and (3) above, we obviously get $\mathcal{F}_2 = \mathcal{F}_1 \cup \mathcal{H}_1 \cup \mathcal{H}_3$ and $\mathcal{G}_2 = (\mathcal{G}_1 \cup \mathcal{H}_3)\mathcal{H}_2$ subject to the conditions $\mathcal{H}_1 \subseteq \mathcal{G}_1 - \mathcal{H}_2$, $\mathcal{H}_2 \subseteq \mathcal{G}_1 = \mathcal{F}_1$ and $\mathcal{H}_3 \cap \mathcal{G}_1 = \emptyset$. Thus, if v_2 does not result from v_1 by such a sequence of operations, we either find a $C \in \mathcal{G}_2 - \mathcal{F}_2$ with $C \notin \mathcal{G}_1$ or there is some $D \in \mathcal{F}_1$ with $D \notin \mathcal{F}_2 \cap \mathcal{G}_1$. In the first case $\bar{v}_2$ can be made arbitrarily small by a suitable substitution, in the second case $\bar{v}_1$ can be made arbitrarily large. Therefore, we must have $v_1 \not\leq v_2$. $\qquad\square$

With this characterisation of the partial order on $\mathbb{V}$ we can show that $\mathbb{V}$ is in fact a lattice.

Theorem 3.2. *$(\mathbb{V}, \leq)$ is a lattice with top element 1 and bottom element $\dfrac{w(\top)}{\sum_{C \in \mathcal{L}} w(C)}$. The join is given by $v_1 \sqcup v_2 = \dfrac{\sum_{C \in \mathcal{F}_1 \cup \mathcal{F}_2} w(C)}{\sum_{C \in (\mathcal{G}_1 \cup \mathcal{F}_2) \cap (\mathcal{F}_1 \cup \mathcal{G}_2)} w(C)}$, and the meet is given by $v_1 \sqcap v_2 = \dfrac{\sum_{C \in \mathcal{F}_1 \cap \mathcal{F}_2} w(C)}{\sum_{C \in \langle (\mathcal{G}_1 - \mathcal{F}_1) \cup (\mathcal{G}_2 - \mathcal{F}_2) \cup (\mathcal{F}_1 \cap \mathcal{F}_2) \rangle} w(C)}$, where $\langle S \rangle$ denotes the filter generated by the subset S.*

Proof. The expressions for meet and join are well-defined, as the sum in the nominator always ranges over a subset of the range of the sum in the denominator.

Now let $v_i = \dfrac{\sum_{C \in \mathcal{F}_i} w(C)}{\sum_{C \in \mathcal{G}_i} w(C)}$ with $\mathcal{F}_i \subseteq \mathcal{G}_i$. For the join the summands $w(C)$ in the nominator of $v_1 \sqcup v_2$ not appearing in the nominator v_i are those $C \in \mathcal{F}_j - \mathcal{F}_i$

for $\{i, j\} = \{1, 2\}$. Among these, $C \in \mathcal{F}_j - \mathcal{G}_i$ define summands also added to the denominator of $v_1 \sqcup v_2$, i.e. each such C defines an operation of type (3) above to increase v_i. The remaining $C \in (\mathcal{F}_j \cap \mathcal{G}_i) - \mathcal{F}_i$ define operations of type (1) above to increase v_i. Furthermore, $C \in \mathcal{G}_i - \mathcal{G}_j$ define those summands $w(C)$ in the denominator of v_i that do not appear in the denominator of $v_1 \sqcup v_2$, so they define operations of type (2) above to increase v_i. That is, $v_1 \sqcup v_2$ results from v_i by a sequence of operations of types (1), (2) and (3), which shows $v_i \leq v_1 \sqcup v_2$.

Conversely, let $v_1, v_2 \leq v'$. As v' must result from v_i by a sequence of operations of types (1), (2) and (3), it must contain all summands $w(C)$ with $C \in \mathcal{F}_1 \cup \mathcal{F}_2$ in its nominator, and these summands must then also appear in its denominator. Furthermore, as a consequence of increasing v_i by operations of type (2), the denominator must not contain summands $w(C)$ with $C \in (\mathcal{G}_1 \cup \mathcal{G}_2) - (\mathcal{G}_1 \cap \mathcal{G}_2)$ unless $C \in \mathcal{F}_1 \cup \mathcal{F}_2$. That is, the denominator of v' only contains summands $w(C)$ with $C \in (\mathcal{G}_1 \cap \mathcal{G}_2) \cup \mathcal{F}_1 \cup \mathcal{F}_2$, which define the denominator of $v_1 \sqcup v_2$. so we obtain $v_1 \sqcup v_2 \leq v'$, which proves that the join is indeed defined by the equation above.

For the meet we can argue analogously. Comparing v_i with $v_1 \sqcap v_2$ the former one contains additional summands $w(C)$ in its nominator with $C \in \mathcal{F}_i - \mathcal{F}_j$. Out of these, if $C \in \mathcal{F}_i \cap (\mathcal{G}_j - \mathcal{F}_j)$, then summands only appear in the nominator, which gives rise to an operation of type (1) increasing $v_1 \sqcap v_2$. If $C \in \mathcal{F}_i - \mathcal{G}_j$ holds, then the summand $w(C)$ has been added to both the nominator and denominator of v_i, which gives rise to an operation of type (3) to increase $v_1 \sqcap v_2$. If $C \in \mathcal{G}_j - \mathcal{G}_i - \mathcal{F}_j$ holds, then the summand $w(C)$ appears in the denominator of $v_1 \sqcap v_2$, but not in the denominator of v_i, which defines an operation of type (2) to increase $v_1 \sqcap v_2$. That is, v_i results from $v_1 \sqcap v_2$ by a sequence of operations of type (1), (2) and (3) above, which shows $v_1 \sqcap v_2 \leq v_i$.

Conversely, if $v' \leq v_1, v_2$ holds, then the nominator of v' can only contain summands $w(C)$ with $C \in \mathcal{F}_1 \cap \mathcal{F}_2$. If we remove all $C \in \mathcal{F}_i - \mathcal{F}_j$ also from the denominator $\mathcal{G}_i$, which corresponds to operation (3), we obtain $\mathcal{G}_i - (\mathcal{F}_i - \mathcal{F}_j) = (\mathcal{G}_i - \mathcal{F}_i) \cup (\mathcal{G}_i \cap \mathcal{F}_j)$. Furthermore, operations of type (2) that decrease v_i can only add summands to the denominator, so in total we get the union $(\mathcal{G}_1 - \mathcal{F}_1) \cup (\mathcal{G}_2 - \mathcal{F}_2) \cup (\mathcal{F}_1 \cap \mathcal{F}_2)$. However, operations of type (3) can only be applied, if nominator and denominator are defined by filters. This shows $v' \leq v_1 \sqcap v_2$, which proves that the meet is indeed defined by the equation above.

The statements concerning the top and bottom element in $\mathbb{V}$ with respect to $\leq$ are obvious.

$\square$

Example 3.5. Figure 3.2 shows the lattice $(\mathbb{V}, \leq)$ of matching value terms for the lattice $\mathcal{L}$ and the set of filters $\mathbb{F}$ from Example 3.1.

Definition 3.5. A relation $r \subseteq \mathbb{F} \times \mathbb{F}$ is called *admissible* iff the following conditions hold:

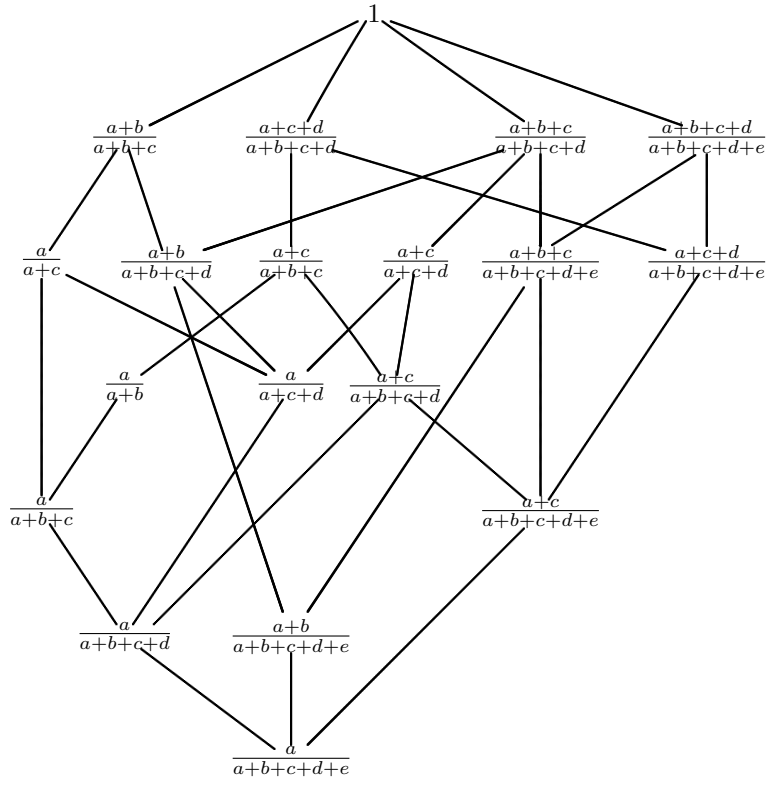(1) If $\mathcal{G} \subseteq \mathcal{F}$ holds, then $r(\mathcal{F}, \mathcal{G})$ holds.

Figure 3.2: The lattice of matching value terms for the lattice $\mathcal{L}$

(2) If $r(\mathcal{F}, \mathcal{G})$ holds and $C \notin \mathcal{F}$, then also $r(\mathcal{F}, \mathcal{G} - \{C\})$ holds.

(3) If $r(\mathcal{F}, \mathcal{G})$ holds, then also $r(\mathcal{F} \cup \{C\}, \mathcal{G} \cup \{C\})$ holds for all $C \in \mathcal{L}$.

Let us concentrate on a single admissible relation $r \subseteq \mathbb{F} \times \mathbb{F}$.

Lemma 3.3. *Let $r \subseteq \mathbb{F} \times \mathbb{F}$ be an admissible relation. Define the set*

$$\mathcal{R} = \{v \in \mathbb{V} \mid \exists \mathcal{F}, \mathcal{G} \in \mathbb{F}. \ v = mvt(\mathcal{F}, \mathcal{G}) \wedge r(\mathcal{F}, \mathcal{G})\}$$

of matching value terms. Then $\mathcal{R}$ is a filter in the lattice $(\mathbb{V}, \leq)$.

Proof. Let $v_1 \in \mathcal{R}$, say $v_1 = mvt(\mathcal{F}_1, \mathcal{G}_1)$ such that $r(\mathcal{F}_1, \mathcal{G}_1)$ holds. Let $v_1 \leq v_2$ for some other mvt $v_2 \in \mathbb{V}$. Without loss of generality we can assume that v_1 and v_2 differ by one of the three possible cases (1), (2) and (3) used for Lemma 3.1. We have to show $v_2 \in \mathcal{R}$.

(1) In this case v_2 differs from v_1 by a summand $w(C)$ added to the nominator, i.e. $v_2 = mvt(\mathcal{F}_2, \mathcal{G}_1)$ with $\mathcal{F}_2 = \mathcal{F}_1 \cup \{C\}$ for $C \in \mathcal{G}_1 - \mathcal{F}_1$. Then $r(\mathcal{F}_2, \mathcal{G}_1)$ holds due to property (3) of Definition 3.5, which gives $v_2 \in \mathcal{R}$ in this case.

(2) In this case v_1 differs from v_2 by a summand $w(C)$ added to the denominator, i.e. $v_2 = mvt(\mathcal{F}_1, \mathcal{G}_2)$ with $\mathcal{G}_2 = \mathcal{G}_1 - \{C\}$ for some $C \notin \mathcal{F}_1$. Then $r(\mathcal{F}_1, \mathcal{G}_2)$ holds due to property (2) of Definition 3.5, which gives $v_2 \in \mathcal{R}$ also in this case.

(3) In this case v_2 differs from v_1 by a summand $w(C)$ added to both the nominator and the denominator, i.e. $v_2 = mvt(\mathcal{F}_2, \mathcal{G}_2)$ with $\mathcal{G}_2 = \mathcal{G}_1 \cup \{C\}$ and $\mathcal{F}_2 = \mathcal{F}_1 \cup \{C\}$ for some $C \notin \mathcal{G}_1$. Again, due to property (3) of Definition 3.5 we obtain $r(\mathcal{F}_2, \mathcal{G}_2)$, which gives $v_2 \in \mathcal{R}$ and completes the proof.

$\square$

4. Learning Matching Measures from User-Defined Matchings

In the previous Section 3 we developed a general theory of matching exploiting filters in lattices. Such lattices can be derived from knowledge bases as shown in Section 2. We now address the problem how to learn a matching measure from matching values that are given by a human domain expert. For this let $(\mathcal{L}, \leq)$ be a finite lattice, and let $\mathbb{F}$ denote the set of all its filters. Note that each filter $\mathcal{F} \in \mathbb{F}$ is uniquely determined by its minimal elements, so we can write $\mathcal{F} = \langle C_1, \ldots, C_k \rangle$. The matching knowledge of a human expert can be represented be a mapping $h : \mathbb{F} \times \mathbb{F} \to [0, 1]$. Though human experts will hardly ever provide complete information, we will assume in the sequel that h is total.

However, the matching values as such are merely used to determine rankings, whereas their concrete value is of minor importance. Therefore, instead of asking whether there exists a matching measure μ on $\mathbb{F}$ such that $\mu(\mathcal{F}, \mathcal{G}) = h(\mathcal{F}, \mathcal{G})$ holds for all pairs of filters, we investigate the slightly weaker problem to find a

ranking-preserving matching measure μ on $\mathbb{F}$, i.e. the matching measure should imply the same rankings.

At least this is the case for rankings with respect to a fixed requested profile. For a fixed given profile a bit more care is needed, as the following example shows.

Example 4.1. Consider the following example from the recruiting domain. Let

$$\mathcal{F} = \{\text{Skill}, \text{Roman\_language}, \text{Programming}\},$$
$$\mathcal{G}_1 = \{\text{Skill}, \text{Roman\_language}, \text{Italian}\} \qquad \text{and}$$
$$\mathcal{G}_2 = \{\text{Skill}, \text{Programming}, \text{Java}\}.$$

It makes perfectly sense to rank *given* profiles $\mathcal{G}_1$ and $\mathcal{G}_2$ with respect to a *requested* profile $\mathcal{F}$, i.e. to consider $h(\mathcal{G}_1, \mathcal{F})$ and $h(\mathcal{G}_2, \mathcal{F})$ and to preserve a ranking between these two in a weighted matching measure, even if such a requested profile appears to be rather odd.

However, if $\mathcal{F}$ is a given profile (which may not be unrealistic), then it appears doubtful, if a ranking for $\mathcal{G}_1$ (emphasising language skills) and $\mathcal{G}_2$ (emphasising programming skills) makes sense at all.

Therefore, it seems plausible to classify profiles with respect to their *relevance* for $\mathcal{F}$ using an equivalence relation $\sim_{\mathcal{F}}$ on $\mathbb{F}$ defined as $\mathcal{G}_1 \sim_{\mathcal{F}} \mathcal{G}_2$ iff $\mathcal{F} \cap \mathcal{G}_1 = \mathcal{F} \cap \mathcal{G}_2$. In doing so our claim becomes that rankings by h should be preserved for a given profile $\mathcal{F}$ in *relevance classes* for $\mathcal{F}$.

Definition 4.1. A matching measure μ on $\mathbb{F}$ is called *ranking-preserving* with respect to $h : \mathbb{F} \times \mathbb{F} \to [0, 1]$ if for all filters the following two conditions hold:

(1) $\mu(\mathcal{F}_1, \mathcal{G}) > \mu(\mathcal{F}_2, \mathcal{G})$ holds, whenever $h(\mathcal{F}_1, \mathcal{G}) > h(\mathcal{F}_2, \mathcal{G})$ holds;
(2) $\mu(\mathcal{F}, \mathcal{G}_1) > \mu(\mathcal{F}, \mathcal{G}_2)$ holds, whenever $h(\mathcal{F}, \mathcal{G}_1) > h(\mathcal{F}, \mathcal{G}_2)$ holds, provided that $\mathcal{G}_1$ and $\mathcal{G}_2$ are in the same relevance class with respect to $\mathcal{F}$, i.e. $\mathcal{G}_1 \sim_{\mathcal{F}} \mathcal{G}_2$.

Thus, this section is dedicated to finding conditions for h that guarantee the existence of a ranking-preserving matching measure μ with respect to h.

4.1. Plausibility Constraints

We are looking for plausibility constraints for the mapping h that should be satisfied in the absence of bias, i.e. the assessment of the human expert is not grounded in hidden concepts. If such plausibility conditions are satisfied we explore the existence of a ranking-preserving matching measure μ. First we show the following simple lemma.

Lemma 4.1. *Let μ be a matching measure on $\mathbb{F}$. Then for all filters $\mathcal{F}, \mathcal{F}_1$, $\mathcal{F}_2, \mathcal{G} \in \mathbb{F}$ the following conditions hold, provided that the arguments of μ are filters:*

(1) $\mu(\mathcal{F}, \mathcal{G}) = 1$ *for $\mathcal{G} \subseteq \mathcal{F}$.*

(2) $\mu(\mathcal{F}, \mathcal{G}) = \mu(\mathcal{F} \cap \mathcal{G}, \mathcal{G})$.

(3) $\mu(\mathcal{F}, \mathcal{G}) < \mu(\mathcal{F}, \mathcal{G} - \{C\})$ *holds for* $C \in \mathcal{G} \setminus \mathcal{F}$.

(4) $\mu(\mathcal{F}, \mathcal{G}) \leq \mu(\mathcal{F} \cup \{C\}, \mathcal{G} \cup \{C\})$.

(5) *If* $\mu(\mathcal{F}_1, \mathcal{G}) < \mu(\mathcal{F}_2, \mathcal{G})$ *holds, then* $\mu(\mathcal{F}_1 \cup \{C\}, \mathcal{G}) < \mu(\mathcal{F}_2 \cup \{C\}, \mathcal{G})$ *holds for every* $C \in \mathcal{G} - \mathcal{F}_1 - \mathcal{F}_2$.

(6) *If* $\mathcal{F} \cap \mathcal{F}_1 \cap \mathcal{G} = \mathcal{F} \cap \mathcal{F}_2 \cap \mathcal{G}$ *holds, then* $\mu(\mathcal{F}_1, \mathcal{G}) > \mu(\mathcal{F}_2, \mathcal{G}) \Leftrightarrow \mu(\mathcal{F}, \mathcal{F}_1 \cap \mathcal{G}) < \mu(\mathcal{F}, \mathcal{F}_2 \cap \mathcal{G})$.

(7) *If* $\mu(\mathcal{F}, \mathcal{G}_1) < \mu(\mathcal{F}, \mathcal{G}_2)$ *holds, then for every* $C \in \mathcal{G}_1 \cap \mathcal{G}_2$ *also* $\mu(\mathcal{F} \cup \{C\}, \mathcal{G}_1) < \mu(\mathcal{F} \cup \{C\}, \mathcal{G}_2)$ *holds, provided that* $\mathcal{F} \cap \mathcal{G}_1 = \mathcal{F} \cap \mathcal{G}_2$ *holds.*

(8) *If* $\mu(\mathcal{F}_1, \mathcal{G}) < \mu(\mathcal{F}_2, \mathcal{G})$ *and* $\mathcal{G}' \cap \mathcal{F}_i = \mathcal{G} \cap \mathcal{F}_i$ *hold, then also* $\mu(\mathcal{F}_1, \mathcal{G}') < \mu(\mathcal{F}_2, \mathcal{G}')$ *holds.*

Proof. Properties (1), (2), (5) and (8) are obvious from the Definition 3.3 of matching measures. For property (6) both sides of the equivalence are equivalent to $w(\mathcal{F}_1 \cap \mathcal{G}) > w(\mathcal{F}_2 \cap \mathcal{G})$.

For property (3) we have

$$\mu(\mathcal{F}, \mathcal{G}) = \frac{w(\mathcal{F} \cap \mathcal{G})}{w(\mathcal{G} - \{C\}) + w(C)} < \frac{w(\mathcal{F} \cap (\mathcal{G} - \{C\}))}{w(\mathcal{G} - \{C\})} = \mu(\mathcal{F}, \mathcal{G} - \{C\}) \,.$$

For property (4) the case $C \in \mathcal{F}$ is trivial. In case $C \in \mathcal{G} - \mathcal{F}$ holds, we get

$$\mu(\mathcal{F}, \mathcal{G}) = \frac{w(\mathcal{F} \cap \mathcal{G})}{w(\mathcal{G})} \leq \frac{w(\mathcal{F} \cap \mathcal{G}) + w(C)}{w(\mathcal{G})} = \mu(\mathcal{F} \cup \{C\}, \mathcal{G} \cup \{C\}) \,.$$

In case $C \notin \mathcal{G}$ first note that for any values a, b, c with $a \leq b$ we get $ab + ac \leq ab + bc$ and thus $\dfrac{a}{b} \leq \dfrac{a+c}{b+c}$.

Thus, we get $\mu(\mathcal{F}, \mathcal{G}) = \dfrac{w(\mathcal{F} \cap \mathcal{G})}{w(\mathcal{G})} \leq \dfrac{w(\mathcal{F} \cap \mathcal{G}) + w(C)}{w(\mathcal{G}) + w(C)} = \mu(\mathcal{F} \cup \{C\}, \mathcal{G} \cup \{C\})$.

For property (7) assume that we have $\mu(\mathcal{F}, \mathcal{G}_1) < \mu(\mathcal{F}, \mathcal{G}_2)$. This is equivalent with $w(\mathcal{G}_1) > w(\mathcal{G}_2)$ by the definition of μ and $\mathcal{F} \cap \mathcal{G}_1 = \mathcal{F} \cap \mathcal{G}_2$. On the other hand, for $C \in \mathcal{G}_1 \cap \mathcal{G}_2$, $(\mathcal{F} \cup \{C\}) \cap \mathcal{G}_1 = (\mathcal{F} \cup \{C\}) \cap \mathcal{G}_2$ also holds.

That is, we obtain $\mu(\mathcal{F} \cup \{C\}, \mathcal{G}_1) < \mu(\mathcal{F} \cup \{C\}, \mathcal{G}_2)$ as claimed. $\qquad \square$

Informally phrased property (1) states that whenever all requirements in a requested profile $\mathcal{G}$ (maybe even more) are satisfied by a given profile $\mathcal{F}$, then $\mathcal{F}$ is a perfect match for $\mathcal{G}$. Property (2) states that the matching value indicating how well the given profile $\mathcal{F}$ fits to the requested one $\mathcal{G}$ only depends on $\mathcal{F} \cap \mathcal{G}$, i.e. the properties in the given profile that are relevant for the requested one. Property (3) states that if a requirement not satisfied by a given profile $\mathcal{F}$ is removed from the requested profile $\mathcal{G}$, the given profile will become a better match for the restricted profile. Property (4) covers two cases. If $C \in \mathcal{G}$ holds, then simply the profile $\mathcal{F} \cup \{C\}$ satisfies more requirements than $\mathcal{F}$, so the

matching value should increase. The case $C \notin \mathcal{G}$ is a bit more tricky, as the profile $\mathcal{G} \cup \{C\}$ contains an additional requirement, which is satisfied by the enlarged profile $\mathcal{F} \cup \{C\}$. In this case the matching value should increase, because the percentage of requirements that are satisfied increases. Property (5) states that the given profile $\mathcal{F}_1$ is better suited for the required profile $\mathcal{G}$ than the given profile $\mathcal{F}_2$, then adding a new required property C to both given profiles preserves the inequality between the two matching values. Property (6) states that if the given profile $\mathcal{F}_1$ is better suited for the required profile $\mathcal{G}$ than the given profile $\mathcal{F}_2$, then relative to $\mathcal{G}$ the profile $\mathcal{F}_2$ is less over-qualified than $\mathcal{F}_1$ for any other required profile $\mathcal{F}$, provided the intersections of $\mathcal{F} \cap \mathcal{G}$ with the two given profiles coincide. Property (7) states that if $\mathcal{F}$ fits better to $\mathcal{G}_2$ than to $\mathcal{G}_1$ and the relevance of $\mathcal{F}$ with respect to $\mathcal{G}_1$ and $\mathcal{G}_2$ (expressed by the intersection) is the same, then adding property C to $\mathcal{F}$ that is requested in both $\mathcal{G}_1$ and $\mathcal{G}_2$ preserves this dependency, i.e. $\mathcal{F} \cup \{C\}$ fits better to $\mathcal{G}_2$ than to $\mathcal{G}_1$. Property (8) states that if $\mathcal{F}_2$ fits better to $\mathcal{G}$ than $\mathcal{F}_1$, then this is also the case for any other requested filter $\mathcal{G}'$ that preserves the relevance of $\mathcal{G}$ for both filters $\mathcal{F}_1$ and $\mathcal{F}_2$.

Thus, disregarding for the moment our theory of matching measures, all eight properties in Lemma 4.1 appear to be reasonable. Therefore, we require them as *plausibility constraints* that a human-defined mapping $h : \mathbb{F} \times \mathbb{F} \to [0,1]$ should satisfy.

Definition 4.2. A function $h : \mathbb{F} \times \mathbb{F} \to [0,1]$ *satisfies the plausibility constraints*, if the following conditions are satisfied, provided that the arguments of h are filters:

(1) $h(\mathcal{F}, \mathcal{G}) = 1$ for $\mathcal{G} \subseteq \mathcal{F}$,
(2) $h(\mathcal{F}, \mathcal{G}) = h(\mathcal{F} \cap \mathcal{G}, \mathcal{G})$,
(3) $h(\mathcal{F}, \mathcal{G}) < h(\mathcal{F}, \mathcal{G} - \{C\})$ for any concept $C \in \mathcal{G} \setminus \mathcal{F}$, and
(4) $h(\mathcal{F}, \mathcal{G}) \leq h(\mathcal{F} \cup \{C\}, \mathcal{G} \cup \{C\})$ for any concept C.
(5) If $h(\mathcal{F}_1, \mathcal{G}) < h(\mathcal{F}_2, \mathcal{G})$ holds, then $h(\mathcal{F}_1 \cup \{C\}, \mathcal{G}) < h(\mathcal{F}_2 \cup \{C\}, \mathcal{G})$ holds for every $C \in \mathcal{G} - \mathcal{F}_1 - \mathcal{F}_2$.
(6) If $\mathcal{F} \cap \mathcal{F}_1 \cap \mathcal{G} = \mathcal{F} \cap \mathcal{F}_2 \cap \mathcal{G}$ holds, then $h(\mathcal{F}_1, \mathcal{G}) > h(\mathcal{F}_2, \mathcal{G}) \Leftrightarrow h(\mathcal{F}, \mathcal{F}_1 \cap \mathcal{G}) < h(\mathcal{F}, \mathcal{F}_2 \cap \mathcal{G})$.
(7) If $h(\mathcal{F}, \mathcal{G}_1) < h(\mathcal{F}, \mathcal{G}_2)$ holds, then for every $C \in \mathcal{G}_1 \cap \mathcal{G}_2$ also $h(\mathcal{F} \cup \{C\}, \mathcal{G}_1) < h(\mathcal{F} \cup \{C\}, \mathcal{G}_2)$ holds, provided that $\mathcal{F} \cap \mathcal{G}_1 = \mathcal{F} \cap \mathcal{G}_2$ holds.
(8) If $h(\mathcal{F}_1, \mathcal{G}) < h(\mathcal{F}_2, \mathcal{G})$ and $\mathcal{G}' \cap \mathcal{F}_i = \mathcal{G} \cap \mathcal{F}_i$ hold, then also $h(\mathcal{F}_1, \mathcal{G}') < h(\mathcal{F}_2, \mathcal{G}')$ holds.

Note that condition (5) in Definition 4.2 also implies a reverse implication, i.e. $h(\mathcal{F}_1 \cup \{C\}, \mathcal{G}) < h(\mathcal{F}_2 \cup \{C\}, \mathcal{G})$ implies $h(\mathcal{F}_1, \mathcal{G}) \leq h(\mathcal{F}_2, \mathcal{G})$. Also note that (6) and (4) imply (2) by substitution into (6) of filters in (2) as follows. $\mathcal{F}_1 = \mathcal{F}$, $\mathcal{F}_2 = \mathcal{F} \cap \mathcal{G}$, $\mathcal{G} = \mathcal{G}$, $\mathcal{F} = \top$. (6) gives

$$h(\mathcal{F}, \mathcal{G}) > h(\mathcal{F} \cap \mathcal{G}, \mathcal{G}) \iff h(\top, \mathcal{F} \cap \mathcal{G}) < h(\top, \mathcal{F} \cap \mathcal{G}),$$

thus implying $h(\mathcal{F}, \mathcal{G}) \leq h(\mathcal{F} \cap \mathcal{G}, \mathcal{G})$. The inequality in the other direction is a straightforward corollary of (4).

Example 4.2. Take a lattice with top element A and direct successors B, C, D, E. Assume that we have $h(\{A, B, E\}, \{A, B, C, D\}) < h(\{A, C, E\}, \{A, B, C, D\})$. Then plausibility constraint (4) implies that $h(\{A, B, E\}, \{A, B, C, D\}) < h(\{A, B, D, E\}, \{A, B, C, D\})$ and $h(\{A, C, E\}, \{A, B, C, D\}) \leq h(\{A, C, D, E\}, \{A, B, C, D\})$ hold. Furthermore, plausibility constraint (4) implies that also

$$h(\{A, B, D, E\}, \{A, B, C, D\}) \leq h(\{A, C, D, E\}, \{A, B, C, D\})$$

must hold.

4.2. Linear Inequalities

Let $h : \mathbb{F} \times \mathbb{F} \to [0, 1]$ be a human-defined function that satisfies the plausibility constraints. Assume the lattice $\mathcal{L}$ contains $n + 2$ elements $C_0, \ldots, C_{n+1}$ with top- and bottom elements C_0 and C_{n+1}, respectively. From this we will now derive a set of linear inequalities of the form $\sum_{x \in U} x < \sum_{x \in V} x$, where the elements in U and V correspond to $C_1, \ldots, C_n$.

Lemma 4.2. *Let $h : \mathbb{F} \times \mathbb{F} \to [0, 1]$ be a human-defined function that satisfies the plausibility constraints. Then there exists a partial order $\prec$ on the set of terms $\Sigma_I = \{\sum_{i \in I} x_i \mid I \subseteq \{1, \ldots, n\}\}$ and a mapping $\Phi : \mathbb{F} \to \{\Sigma_I \mid I \subseteq \{1, \ldots, n\}\}$ such that the following conditions hold:*

(1) *For all filters $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{G}$ we have $h(\mathcal{F}_1, \mathcal{G}) < h(\mathcal{F}_2, \mathcal{G})$ implies $\Phi(\mathcal{F}_1 \cap \mathcal{G}) \prec \Phi(\mathcal{F}_2 \cap \mathcal{G})$;*

(2) *For all filters $\mathcal{F}$, $\mathcal{G}_1$, $\mathcal{G}_2$ with $\mathcal{F} \cap \mathcal{G}_1 = \mathcal{F} \cap \mathcal{G}_2$ we have $h(\mathcal{F}, \mathcal{G}_1) < h(\mathcal{F}, \mathcal{G}_2)$ implies $\Phi(\mathcal{G}_2) \prec \Phi(\mathcal{G}_1)$;*

(3) *For $J \subset I$ we have $\Sigma_J \prec \Sigma_I$;*

(4) *For $\Sigma_J \prec \Sigma_I$ and $k \notin J \cup I$ we also have $\Sigma_J + x_k \prec \Sigma_I + x_k$.*

Proof. Let $\mathcal{L}$ contain $n+2$ elements $C_0, \ldots, C_{n+1}$ with top- and bottom elements C_0 and C_{n+1}, respectively. Define $\Phi(\mathcal{F}) = \sum_{C_i \in \mathcal{F} - \{C_0, C_{n+1}\}} x_i$.

For $\mathcal{G} = \mathcal{L}$ the inequality $h(\mathcal{F}_1, \mathcal{G}) < h(\mathcal{F}_2, \mathcal{G})$ defines the inequality $\Phi(\mathcal{F}_1) \prec \Phi(\mathcal{F}_2)$. We show that inequalities between sums of variables defined this way satisfy the requirements of the Lemma.

For $\mathcal{G} \neq \mathcal{L}$ the inequality $h(\mathcal{F}_1, \mathcal{G}) < h(\mathcal{F}_2, \mathcal{G})$ implies $h(\mathcal{F}_1 \cap \mathcal{G}, \mathcal{G}) < h(\mathcal{F}_2 \cap \mathcal{G}, \mathcal{G})$ due to plausibility constraint (2). Plausibility constraint (8) then gives rise to $h(\mathcal{F}_1 \cap \mathcal{G}, \mathcal{L}) < h(\mathcal{F}_2 \cap \mathcal{G}, \mathcal{L})$, which defines the inequality $\Phi(\mathcal{F}_1 \cap \mathcal{G}) \prec \Phi(\mathcal{F}_2 \cap \mathcal{G})$ needed.

For $\mathcal{F} = \{C_0\}$ the inequality $h(\mathcal{F}, \mathcal{G}_2) < h(\mathcal{F}, \mathcal{G}_1)$ implies using plausibility constraint (6) with $\mathcal{G} = \mathcal{L}$ that $h(\mathcal{G}_1, \mathcal{G}) < h(\mathcal{G}_2, \mathcal{G})$, so the inequality $\Phi(\mathcal{G}_1) < \Phi(\mathcal{G}_2)$ is again the same as the one defined by the right hand side $\mathcal{L}$.

Let $\mathcal{F} \neq \{C_0\}$ with $\mathcal{F} \cap \mathcal{G}_1 = \mathcal{F} \cap \mathcal{G}_2$ (due to plausibility constraint (1) we can ignore $\mathcal{F} = \mathcal{L}$). the inequality $h(\mathcal{F}, \mathcal{G}_2) < h(\mathcal{F}, \mathcal{G}_1)$ defines again $\Phi(\mathcal{G}_1) \preceq \Phi(\mathcal{G}_2)$. According to plausibility constraint (7) we also have $h(\mathcal{F} - \{C\}, \mathcal{G}_2) \leq h(\mathcal{F} - \{C\}, \mathcal{G}_1)$ for $C \in \mathcal{G}_1 \cap \mathcal{G}_2$. As $\mathcal{F} \cap \mathcal{G}_i \subseteq \mathcal{G}_1 \cap \mathcal{G}_2$ holds, we obtain $h(\{C_0\}, \mathcal{G}_2) \leq h(\{C_0\}, \mathcal{G}_1)$, so the derived inequality is again the same as for the case $\mathcal{F} = \{C_0\}$. This shows the claimed properties (1) and (2) of the lemma.

In order to see the claimed property (3) we fix $\mathcal{F} = \{C_0\}$ and exploit plausibility constraint (3) using induction on the size of $I \setminus J$.

From plausibility constraint (5) we obtain $h(\mathcal{F}_1 \cup \{C_k\}, \mathcal{G}) < h(\mathcal{F}_2 \cup \{C_k\}, \mathcal{G})$ for $C_k \notin \mathcal{F}_1 \cup \mathcal{F}_2$, which gives the claimed property (4) $\Sigma_J + x_k \prec \Sigma_I + x_k$.

Finally extend the obtained partial order $\preceq$ to a total one preserving properties (3) and (4).

$\square$

Note that we obtain directly a partial order for the "worst case", i.e. the lattice $\mathcal{L}$, in which all C_i $(i = 1, \ldots, n)$ are pairwise incomparable. Thus we could have first extended h to this case, where all subsets correspond to filters, and then used the arguments in the proof.

With Lemma 4.2 we reduce the problem of finding a ranking-preserving matching measure to a problem of solving a set of linear inequalities. We will exploit the properties in this lemma for the proof of our main result in the next subsection. First we investigate a general condition for realisability.

Definition 4.3. Let $\mathcal{P}$ be a set of linear inequalities on the set of terms $\{\sum_{i \in I} x_i \mid I \subseteq \{1, \ldots, n\}\}$. We say that $\mathcal{P}$ is *realisable*, if there is a substitution $v : \{x_1, \ldots, x_n\} \to \mathbb{R}^+$ of the variables by positive real numbers such that $\sum_{i \in I} x_i$ precedes $\sum_{j \in J} x_j$ in $\mathcal{P}$ iff $\sum_{i \in I} v(x_i) < \sum_{j \in J} v(x_j)$ holds.

As all sums are finite, it is no loss of generality to seek substitutions by rational numbers, and further using the common denominator it suffices to consider positive integers only.

For convenience we introduce the notation $U \prec V$ for multisets U, V over $\{x_1, \ldots, x_n\}$ to denote the inequality $\sum\limits_{x_i \in U} m_U(x_i) x_i < \sum\limits_{x_j \in V} m_V(x_j) x_j$, where m_U and m_V are the multiplicities for the two multisets.

Theorem 4.3. $\mathcal{P}$ *is realisable iff there is no positive integer combination of inequalities in $\mathcal{P}$ that results in $A \prec B$ with $B \subseteq A$ as multisets, i.e. $m_B(x_i) \leq m_A(x_i)$ for all $i = 1, \ldots, n$.*

Proof. The necessity of the condition is obvious, since if $\mathcal{P}$ is realizable and $A \prec B$ is a positive integer combination of inequalities from $\mathcal{P}$, then $\sum_{x_i \in A} m_A(x_i) v(x_i) < \sum_{x_j \in B} m_B(x_j) v(x_j)$ follows for the realizer substitution $v \colon X \to \mathbb{R}_{>0}$ that contradicts to $m_B(x_i) \leq m_A(x_i)$ for all $i = 1, 2, \ldots, n$.

To prove that the condition is sufficient we use *Fourier–Motzkin elimination*. That is, we do induction on the number of variables. For one variable, the statement is trivial. For the sake of convenience the inequalities are transformed into the following standard form

$$\alpha_{i_1} x_{i_1} + \alpha_{i_2} x_{i_2} + \ldots + \alpha_{i_k} x_{i_k} - \beta_{j_1} x_{j_1} - \beta_{j_2} x_{j_2} - \ldots - \beta_{j_p} x_{j_p} < 0$$

for $\alpha_{i_t}, \beta_{j_r} \in \mathbb{N}$. The condition is now that no positive integer combination of the inequalities result in an inequality with all variables having nonnegative coefficient. Consider variable x_q. Let $\mathcal{P}_0$ denote the set of those inequalities

that do not involve x_q, while $\mathcal{P}_1$ denote the set of those inequalities that do involve x_q.

CASE 1. x_q occurs only with negative coefficients in inequalities of $\mathcal{P}_1$. Let us assume that the set of inequalities $\mathcal{P}_0$ is realizable, i.e., there is a substitution of the other variables that makes those inequalities valid. Then this substitution of the other variables and a large enough value for x_q will satisfy the inequalities of $\mathcal{P}_1$, as well.

CASE 2. x_q occurs only with positive coefficients in inequalities of $\mathcal{P}_1$. Let $\mathcal{P}_1'$ denote the set of inequalities obtained from $\mathcal{P}_1$ by removing occurrences of x_q from each inequality in $\mathcal{P}_1$. We claim that $\mathcal{P}_0 \cup \mathcal{P}_1'$ satisfies the property that no positive integer combination of inequalities of $\mathcal{P}_0 \cup \mathcal{P}_1'$ that results in $A \prec B$ with $B \subseteq A$ as multisets, i.e., $m_B(x_i) \leq m_A(x_i)$ for all $i = 1, 2, \ldots, n$. Indeed, if such a combination existed, then replacing the inequalities of $\mathcal{P}_1'$ by their counterparts of $\mathcal{P}_1$, an invalid positive integer linear combination of $\mathcal{P}$ would be obtained. Since $\mathcal{P}_0 \cup \mathcal{P}_1'$ has one less variable than $\mathcal{P}$, by induction hypothesis it is realizable. The inequalities in $\mathcal{P}_0 \cup \mathcal{P}_1'$ are all strict, so this substitution of the other variables and a small enough value for x_q will satisfy the inequalities of $\mathcal{P}_1$, as well.

CASE 3. x_q occurs with positive and negative coefficients, as well in $\mathcal{P}_1$. For every pair of inequalities such that the coefficient of x_q is positive in one of them and negative in the other one we create a new inequality which is a positive integer combination of the two so that the coefficient of x_q is reduced to zero, denote the set of inequalities obtained in such way $\mathcal{P}^*$. Let $\mathcal{P}' = \mathcal{P}_0 \cup \mathcal{P}^*$. It is clear that there exists no positive integer combination of inequalities of $\mathcal{P}'$ that have all variables with non-negative coefficients, since any such combination is also a positive integer combination of inequalities of $\mathcal{P}_0 \cup \mathcal{P}_1$. Thus, by the induction hypothesis $\mathcal{P}'$ realizable, so there exists a substitution of the other variables than x_q that makes every inequality in $\mathcal{P}'$ valid. Each inequality in $\mathcal{P}_1$ that has x_q with positive coefficient gives an upper bound for x_q with this substitution. Also, the inequalities of $\mathcal{P}_1$ containing x_q with negative coefficient give lower bounds for x_q. We claim that the largest lower bound obtained so is smaller than the smallest upper bound. Indeed, consider the pair of inequalities giving these bounds. If the upper bound were less than or equal of the lower bound, then the positive integer combination of these two inequalities that is included in $\mathcal{P}^*$ would be violated by the given substitution of the variables other than x_q. □

4.3. Ranking-Preserving Matching Measures

We now use Theorem 4.3 to prove the existence of ranking-preserving matching measures. As $\mathcal{P}$ is defined by h we can assume that $\sum_{i \in I} x_i$ precedes $\sum_{j \in J} x_j$ for $I \subset J$. We can then also extend $\mathcal{P}$ to a partial order $\hat{\mathcal{P}}$ on multisets of variables by adding the same variable(s) to both sides. Indeed, $\Sigma_J \prec \Sigma_I \iff \Sigma_{J \setminus I} \prec \Sigma_{I \setminus J}$ by (4) of Lemma 4.2. Clearly, $\mathcal{P}$ is realisable iff $\hat{\mathcal{P}}$

is realisable. Here a partial order is considered realized iff the collection of strict inequalities is realized as a set of linear inequalities in the sense of Theorem 4.3

Lemma 4.4. *Let $\mathcal{P}$ be a partial order on the set of terms $\{\Sigma_I \mid I \subseteq \{1,\ldots,n\}\}$ for $\Sigma_I = \sum_{i \in I} x_i$ such that $\Sigma_J < \Sigma_I$ holds for $J \subset I$ and $\Sigma_J + x_k < \Sigma_I + x_k$ holds, whenever $\Sigma_J < \Sigma_I$ holds and $k \notin J \cup I$. Then $\mathcal{P}$ is realisable.*

Proof. Assume that $\mathcal{P}$ is not realisable. Then according to Theorem 4.3 there exist inequalities $U_1 < V_1,\ldots,U_k < V_k$ in $\mathcal{P}$ such that $V = \biguplus_{i=1}^{k} V_i \subseteq \biguplus_{i=1}^{k} U_i = U$ as multisets and

$$\sum_{x \in U} \sum_{j=1}^{k} m_{U_j}(x)x < \sum_{x \in V} \sum_{j=1}^{k} m_{V_j}(x)x.$$

Let this system of inequalities be minimal, so each subset violates the condition in Theorem 4.3. We may assume without loss of generality that $U_i \cap V_i = \emptyset$. Taking the inequalities in some order let

$$A_i = \sum_{x} \sum_{j=1}^{i} m_{U_j}(x)x \qquad \text{and} \qquad B_i = \sum_{x} \sum_{j=1}^{i} m_{V_j}(x)x.$$

Then for $i < k$ there always exists some x with $\sum_{j=1}^{i} m_{U_j}(x) < \sum_{j=1}^{i} m_{V_j}(x)$, while $A_i \prec B_i$. On the other hand $\sum_{j=1}^{k} m_{U_j}(x) \geq \sum_{j=1}^{k} m_{V_j}(x)$, while $A_k \prec B_k$.

Let V_i' be the multiset $B_i - A_i$, i.e. the multiset of all x with $m_{B_i}(x) > m_{A_i}(x)$ such that $m_{V_i'}(x) = m_{B_i}(x) - m_{A_i}(x)$ holds. Each $x \in V_i'$ is a witness for the violation of the condition in Theorem 4.3. In particular, we have $V_i' \neq \emptyset$ for all $i < k$, but $V_k' = \emptyset$.

Let $V_{i+1}'' = V_i' \cap V_{i+1}'$ as multisets, so $m_{V_{i+1}''}(x) = \min(m_{V_i'}(x), m_{V_{i+1}'}(x))$, i.e. x will at most be added to B_i to give B_{i+1}, but not to A_i. In particular, $V_{i+1}'' \subseteq V_i'$. Take the complement U_{i+1}' such that $V_i' = U_{i+1}' \uplus V_{i+1}''$.

As $U_i < V_i$ is in $\mathcal{P}$, we also have $U_i' < V_i$ in $\mathcal{P}$ for all $i > 1$ (U_1' is not yet defined). Indeed, using $U_i \cap V_i = \emptyset$ one can see that $U_{i+1}1' = V_i' \cap U_{i+1}$.

Let $B_1' = V_1 = V_1'$. Then proceed inductively defining $W_i = B_i' - U_{i+1}'$ as well as $A_{i+1}' = U_{i+1}' \uplus W_i$ and $B_{i+1}' = V_{i+1} \uplus W_i$, which gives $A_{i+1}' < B_{i+1}'$ in $\hat{\mathcal{P}}$ and $B_i' = A_{i+1}'$. That is, we obtain a chain

$$B_1' \leq A_2' < B_2' \leq \cdots < B_{k-1}' \leq A_k' < B_k'.$$

Complement these definitions by $U_1' = B_k' \cap U_1 = A_1'$, and $X_0 = B_k' - U_1 = X_1$. Proceed inductively defining

$$C_1 = U_1' \uplus X_0 \prec V_1 \uplus X_1 = B_1' \uplus X_1 = D_1 \text{ and } X_{i+1} = X_i - (U_{i+1} - A_{i+1}')$$

This gives $C_{i+1} = A_{i+1}' \uplus X_i \prec B_{i+1}' \uplus X_{i+1} = D_{i+1}$ and $C_{i+1} \leq D_i$. Due to this construction we also have $X_i \supseteq X_{i+1}$ for all i.

Furthermore, for all elements x in X_0 there exists a maximal i with $x \in V_i - U_{i+1}$ and also $x \in A'_j \cap B'_j$ for all $j \geq i+1$, i.e. $x \in W_i$ and x is added to both sides of $U'_{i+1} < V_{i+1}$ to give the inequality $A'_{i+1} < B'_{i+1}$. To prove this property we proceed as follows: For $x \in V_k$ there is nothing to show. For $x \notin V_k$ we have $x \in W_{k-1}$, i.e. x has been added on both sides of $U'_k < V_k$ to yield $A'_k < B'_k$. In particular, $x \notin U'_k$, but $x \in B'_{k-1}$. Then either $x \in V_{k-1}$ and we are done again or x has been added on both sides of $U'_{k-1} < V_{k-1}$, thus proceeding this way we reach a minimal i with the given property—each x added to $U'_2 < V_2$ appears in V_1, so the process always stops.

On the other hand the assumed non-realisability implies $m_{B_k}(x) \leq m_{A_k}(x)$, so there exists a j with $x \in U_j \notin U'_j$ (we must have $j < i$ for the i given by the just shown property). This implies $x \in X_{j-1} - X_j$, so this (occurrence of) x will not appear in X_k. As we can do this for all $x \in X_0$, we obtain $X_k = \emptyset$, which implies $D_k = C_1$. This defines a cycle in $\hat{\mathcal{P}}$ contradicting the fact that it is a partial order. Therefore, $\mathcal{P}$ must be realisable.

$\square$

Example 4.3. To illustrate the construction in the proof take the following inequalities:

$$
\begin{aligned}
U_1 = & \quad x_1 + x_2 \; < \; x_3 + x_4 & = V_1 \\
U_2 = & \quad x_2 + x_3 \; < \; x_5 & = V_2 \\
U_3 = & \quad x_4 + x_5 \; < \; x_1 + x_3 & = V_3 \\
U_4 = & \quad x_3 \; < \; x_2 & = V_4
\end{aligned}
$$

Their combination (adding up the left and right hand sides) give the following:

$$ x_1 + 2x_2 + 2x_3 + x_4 + x_5 < x_1 + x_2 + 2x_3 + x_4 + x_5, $$

i.e. multiplicities on the right are always smaller or equal as those on the left. According to Theorem 4.3 this means that the system is not realisable.

Taking the inequalities in the given order gives the following sums :

$$
\begin{aligned}
A_1 = & \quad x_1 + x_2 < x_3 + x_4 & = B_1 \\
A_2 = & \quad x_1 + 2x_2 + x_3 < x_3 + x_4 + x_5 & = B_2 \\
A_3 = & \quad x_1 + 2x_2 + x_3 + x_4 + x_5 < x_1 + 2x_3 + x_4 + x_5 & = B_3 \\
A_4 = & \quad x_1 + 2x_2 + 2x_3 + x_4 + x_5 < x_1 + x_2 + 2x_3 + x_4 + x_5 & = B_4
\end{aligned}
$$

In the first three inequalities we always have at least one x_i on the right hand side that has a larger multiplicity than on the left hand side, so it is a *witness* for the violation of the condition in Theorem 4.3, i.e. we have realisability for the corresponding subsystem of inequalities.

In the proof of Theorem 4.5 we take these witnesses into the set V'_i, i.e. we have:

$$ V'_1 = \{x_3, x_4\} \quad V'_2 = \{x_4, x_5\} \quad V'_3 = \{x_3\} \quad V'_4 = \emptyset $$

Whenever we proceed from one of these partial sums to the next some of the witnesses will no longer be witnesses—these we collect in the sets U_i'—while others remain witnesses—these will be collected in V_i''. Thus, we obtain:

$$U_2' = \{x_3\} \qquad\qquad V_2'' = \{x_4\}$$
$$U_3' = \{x_4, x_5\} \qquad\quad V_3'' = \emptyset$$
$$U_4' = \{x_3\} \qquad\qquad V_4'' = \emptyset$$

As $U_i' \subseteq U_i$, i.e. we have subsets of the left hand sides of the original inequalities (for $i > 1$), we can remove some of the summands and still have an inequality $U_i' < V_i$ in $\mathcal{P}$:

$$x_3 < x_5 \qquad x_4 + x_5 < x_1 + x_3 \qquad x_3 < x_2$$

Next we add to each of these inequalities the same sum to the left and right hand side—we denote the iequalities as $A_i' < B_i'$ such that the left hand side of the second inequality becomes V_1', and we inductively obtain a chain

$$B_1' = A_2' < B_2' = \cdots < B_{k-1}' = A_k' < B_k'.$$

The inequalities are as follows:

$$x_3 + x_4 < x_4 + x_5 \qquad x_4 + x_5 < x_1 + x_3 \qquad x_1 + x_3 < x_1 + x_2$$

So far we only exploited the realisability of the proper subsets of inequalities, not the non-realisability of the whole set of inequalities. This is, where the left hand side of the first original inequality comes in, which we intersect with the right hand side of the last constructed inequality. In our case this gives

$$U_1' = \{x_1, x_2\}.$$

As $U_1' \subseteq U_1$ holds, we can again reduce the left hand side of our first inequality (in this case there is nothing to be taken away), and add an additional inequality, so we obtain now:

$$x_1 + x_2 < x_3 + x_4 \quad x_3 + x_4 < x_4 + x_5 \quad x_4 + x_5 < x_1 + x_3 \quad x_1 + x_3 < x_1 + x_2$$

This case this gives us already $X_0 = \emptyset$ and a cycle in $\mathcal{P}$ contradicting the fact that we have a linear order.

Example 4.4. Let us replace the third inequality in Example 4.3 by $x_4 + x_5 < x_1 + x_2 + x_3$. Then the combination of all four inequalities (adding up the left and right hand sides) gives the following:

$$x_1 + 2x_2 + 2x_3 + x_4 + x_5 < x_1 + x_2 + 2x_3 + x_4 + x_5,$$

i.e. according to Theorem 4.3 the system is not realisable.

We take again the inequalities in the given order gives the following sums (which in the proof we denoted as $A_i \prec B_i$ for $i = 1, \ldots, 4$:

$$x_1 + x_2 < x_3 + x_4$$
$$x_1 + 2x_2 + x_3 < x_3 + x_4 + x_5$$
$$x_1 + 2x_2 + x_3 + x_4 + x_5 < x_1 + x_2 + 2x_3 + x_4 + x_5$$
$$x_1 + 2x_2 + 2x_3 + x_4 + x_5 < x_1 + 2x_2 + 2x_3 + x_4 + x_5$$

Again in the first three sums we have witnesses on the right hand side with larger multiplicities than on the left hand side, so we have realisability according to Theorem 4.3. Taking these witnesses into the set V_i' we obtain:

$$V_1' = \{x_3, x_4\} \qquad V_2' = \{x_4, x_5\} \qquad V_3' = \{x_3\} \qquad V_4' = \emptyset$$

Next we collect in the sets U_i' witnesses from the $(i-1)$th sum that are no longer witness in V_i', and in V_i'' we collect witnesses that retain these properties. Thus, we obtain:

$$U_2' = \{x_3\} \qquad\qquad V_2'' = \{x_4\}$$
$$U_3' = \{x_4, x_5\} \qquad\qquad V_3'' = \emptyset$$
$$U_4' = \{x_3\} \qquad\qquad V_4'' = \emptyset$$

As $U_i' \subseteq U_i$, i.e. we have subsets of the left hand sides of the original inequalities (for $i > 1$, which give rise to the following inequalities $U_i' < V_i$ in $\mathcal{P}$ for $i = 2, \ldots, 4$:

$$x_3 < x_5 \qquad x_4 + x_5 < x_1 + x_2 + x_3 \qquad x_3 < x_2$$

Adding to each of these inequalities the same sum to the left and right hand side—we denote the iequalities as $A_i' < B_i'$—such that the left hand side of the $(i+1)$th inequality equals the right hand side of the ith inequality, we obtain a chain

$$x_3 + x_4 < x_4 + x_5 \quad x_4 + x_5 < x_1 + x_2 + x_3 \quad x_1 + x_2 = x_3 < x_1 + 2x_2$$

Now $U_1' = \{x_1, x_2\}$ gives rise to the additional first inequality $x_1 + x_2 < x_3 + x_4$, which together with the other three does not yet define a cycle.

In this case we have $X_0 = \{x_2\} = X_1$ and $X_2 = X_3 = X_4 = \emptyset$, which leads to the following inequalties $C_i < D_i$ for $i = 1, \ldots, 4$:

$$x_1 + 2x_2 < x_2 + x_3 + x_4 \quad x_2 + x_3 + x_4 < x_4 + x_5$$
$$x_4 + x_5 < x_1 + x_2 + x_3 \quad x_1 + x_2 + x_3 < x_1 + 2x_2$$

These again defines a cycle in $\hat{\mathcal{P}}$ contradicting the fact that we have a linear order.

From Lemma 4.4 together with Lemma 4.2 we immediately obtain the following main result of this section.

26

Theorem 4.5. *Let $h : \mathbb{F} \times \mathbb{F} \to [0, 1]$ be a human-defined function that satisfies the plausibility constraints. Then there esists a matching measure μ that is ranking-preserving with respect to h.*

Proof. According to Lemma 4.2 h defines a partial order on the set of terms $\{\Sigma_i \mid I \subseteq \{1, \ldots, n\}\}$, which according to Lemma 4.4 is realisable. Then extend a substitution $v : \{x_1, \ldots, x_n\} \to \mathbb{R}^+$ that gives $\sum_{i \in I} v(x_i) \leq \sum_{j \in J} v(x_j)$ for $\sum_{i \in I} x_i \preceq \sum_{j \in J} x_j$ in $\mathcal{P}$ to x_0 and x_{n+1}. This defines the weighting function w with $w(C_i) = \dfrac{v(x_i)}{\sum_{i=0}^{n+1} v(x_i)}$ and a corresponding matching measure μ. Due to properties (1) and (2) of Lemma 4.2 μ is ranking-preserving with respect to h. $\square$

Note that Theorem 4.5 only states the existence of a ranking preserving matching measure μ. However, we obtain solutions for the linear inequations defined by h by minimising $x_1 + \cdots + x_n - 1$ under the conditions $\sum_{x_j \in V} x_j - \sum_{x_i \in U} x_i > 0$. For this linear optimisation problem the well-known *simplex algorithm* and matching learning approaches [24] can be exploited.

4.4. Totally Preserving Matching Measures

While Theorem 4.5 guarantees the existence of a ranking-preserving matching measure μ for a human-defined function $h : \mathbb{F} \times \mathbb{F} \to [0, 1]$ that satisfies the plausibility constraints, it may still be the case that not all inequalities defined by h are preserved by μ. Another more general problem would be to find out, if from such a human-defined function h we can regain a matching measure μ on $\mathbb{F}$ that preserves all the inequalities defined by h.

If w is a weighting function on $\mathcal{L}$, then it defines a lattice homomorphism $\hat{w} : \mathbb{V} \to [0, 1]$ on the lattice of matching value terms. Let $mvt(\mathcal{F}, \mathcal{G})$ denote the matching value term defined by the filters $\mathcal{F}$ and $\mathcal{G}$.

Definition 4.4. Let h be a matching function satisfying the plausibility constraints, and let w be a weighting function. Then w *totally preserves* h, if for all $\mathcal{F}, \mathcal{G}, \mathcal{F}', \mathcal{G}' \in \mathbb{F}$ $h(\mathcal{F}, \mathcal{G}) \geq h(\mathcal{F}', \mathcal{G}')$ holds iff $\hat{w}(mvt(\mathcal{F}, \mathcal{G})) \geq \hat{w}(mvt(\mathcal{F}', \mathcal{G}'))$ holds.

The following is a counterexample to the existence of a weighting function w totally preserving an arbitrary human-defined function $h : \mathbb{F} \times \mathbb{F} \to [0, 1]$ that satisfies the plausibility constraints. Whether additional necessary conditions of matching measures can be claimed to define plausibility constraints, is an open problem.

Example 4.5. Let $(\mathcal{L}, \leq)$ be a lattice with top element x_1, bottom element y, and $n - 1$ pairwise incomparable elements x_i ($i = 2, \ldots, n, n \geq 4$). Filters of this lattice are of the form $\mathcal{F}_U = \{x_1\} \cup \{x_u : u \in U\}$ for any $U \subseteq \{2, 3, \ldots, n\}$ and $\mathcal{L}$ itself.

Let $w(x_i) = w_i$ and $w(y) = w_y$, then the lattice of matching value terms $(\mathbb{V}, \leq)$ consists of fractions $\frac{w_1 + \sum_{a \in A} w_a}{w_1 + \sum_{b \in B} w_b}$ where $A \subsetneq B \subseteq \{2, 3, \ldots, n\}$ and

$\frac{w_1 + \sum_{a \in A} w_a}{w_y + \sum_{i=1}^{n} w_i}$. From Theorem 3.2 we immediately obtain that $\frac{w_1 + \sum_{a \in A} w_a}{w_1 + \sum_{b \in B} w_b} \leq$ $\frac{w_1 + \sum_{c \in C} w_c}{w_1 + \sum_{d \in D} w_d}$ holds in lattice $(\mathbb{V}, \leq)$ iff $A \subseteq C$ and $C \setminus B = D \setminus B$.

Let us consider the filter in $(\mathbb{V}, \leq)$ generated by $\frac{w_1 + \sum_{a \in A} w_a}{w_1 + \sum_{b \in B} w_b}$ with $A = \{x_2, \ldots, x_k\}$ and $B = \{x_2, \ldots, x_k, x_{k+1}, \ldots, x_\ell\}$ for some $k + 1 < \ell$. We need a set of weights w_i, such that $\exists t$ with

$$t \leq \frac{w_1 + \sum_{c \in C} w_c}{w_1 + \sum_{d \in D} w_d} \iff \frac{w_1 + \sum_{a \in A} w_a}{w_1 + \sum_{b \in B} w_b} \leq \frac{w_1 + \sum_{c \in C} w_c}{w_1 + \sum_{d \in D} w_d}$$

in lattice $(\mathbb{V}, \leq)$. We claim that such a weight assignment does not exist.

Indeed, suppose that w_i is a good one, and assume without loss of generality that $w_k = \min_{i \in \{1,2,\ldots,k\}} w_i$ and $w_{k+1} = \min_{i \in \{k+1,k+2,\ldots,\ell\}} w_i$. Let $U = B \setminus \{k, k+1\}$ and $V = B \setminus \{k\}$. As $\frac{w_1 + \sum_{a \in A} w_a}{w_1 + \sum_{b \in B} w_b} \not\leq \frac{w_1 + \sum_{u \in U} w_u}{w_1 + \sum_{v \in V} w_v}$ in lattice $(\mathbb{V}, \leq)$ we must have

$$\frac{w_1 + \sum_{u \in U} w_u}{w_1 + \sum_{v \in V} w_v} < \frac{w_1 + \sum_{a \in A} w_a}{w_1 + \sum_{b \in B} w_b}$$

numerically. This is equivalent to

$$\left(w_1 + \sum_{u \in U} w_u \right) \left(w_1 + \sum_{b \in B} w_b \right) < \left(w_1 + \sum_{a \in A} w_a \right) \left(w_1 + \sum_{v \in V} w_v \right).$$

Writing in more comprehensible form the above inequality is

$$(w_1 + w_2 + \ldots + w_{k-1} + w_{k+2} + \ldots + w_\ell)(w_1 + w_2 + \ldots + w_\ell) <$$
$$(w_1 + w_2 + \ldots + w_{k-1} + w_{k+1} + \ldots + w_\ell))(w_1 + w_2 + \ldots + w_k).$$

After simplification this is equivalent with

$$(w_1 + w_2 + \ldots + w_{k-1} + w_{k+2} + \ldots + w_\ell)(w_{k+2} + \ldots + w_\ell) < w_{k+1} w_k,$$

which contradicts that $w_{k-1} w_{k+2} \geq w_{k+1} w_k$ by the choice of indices.

5. Matching Queries

In this section we address matching queries. According to Section 2 our starting point is a knowledge base with a fixed TBox and an ABox that is determined by a set of profiles. Actually, we can consider the ABox and the profiles to define a large database. Each profile P defines a filter $\mathcal{F}$ in the lattice $\mathcal{L}$ that is derived from the TBox. The purpose of matching queries is to match a profile to other profiles. This can basically two forms:

(1) If a profile P_r describes a set of requested properties, then we are looking for those profiles that fit well to the requirements. If μ is a matching measure, this means to determine profiles P such that $\mu(\mathcal{F}(P), \mathcal{F}(P_r))$ is high, where $\mathcal{F}(P)$ is the representing filter of the profile P according to Definition 2.4.

(2) If a profile P_g describes a set of given properties, then we are looking for those profiles to which this profile fits well, i.e. to determine profiles P with high matching values $\mu(\mathcal{F}(P_g), \mathcal{F}(P))$.

In both cases we expect an ordered answer, where the order is given by decreasing matching values.

In the recruiting area case (1) refers to a job offer of a company, which defines a requested profile, for which best matching candidates are sought. Usually, these candidates are taken from a well-defined set of applicants, where different applicants may have the same profile. Here case (2) refers to the opposite, a person searching for a suitable job.

In both cases we are usually only interested in high matching values, which motivates to look at top-k queries with some user-defined positive integer k. We want to obtain the k profiles with the highest matching values, or better— as there may be many equally-ranked profiles, which make it impossible to determine exactly the k best ones—we want to get those profiles, for which there are less than k better-ranked profiles. For our two cases this means to determine the results of the following two queries:

(1) $top_k(\mu(\cdot, \mathcal{G})) = \{P \mid |\{P' \mid \mu(\mathcal{F}(P'), \mathcal{G}) > \mu(\mathcal{F}(P), \mathcal{G})\}| < k\}$ with $\mathcal{G} = \mathcal{F}(P_r)$

(2) $top_k(\mu(\mathcal{F}, \cdot)) = \{P \mid |\{P' \mid \mu(\mathcal{F}, \mathcal{F}(P')) > \mu(\mathcal{F}, \mathcal{F}(P))\}| < k\}$ with $\mathcal{F} = \mathcal{F}(P_g)$

We will investigate such top-k queries in Subsection 5.1. We will concentrate on case (1), as case (2) can be handled in a completely analogous way.

In addition we will address gap queries, which for a profile P aim at a minimal enlargement—the difference constitutes the gap—such that P will appear in the result of sufficiently many top-k queries. Again the two cases above can be distinguished, which can be formalised by the following two queries using user-defined positive integers k and ℓ:

(1) For a given profile P determine a profile P' such that $\mathcal{F}(P') - \mathcal{F}(P)$ is minimal with respect to the condition

$$\exists P_1, \ldots, P_\ell . P' \in top_k(\mu(\cdot, \mathcal{F}(P_i))) \qquad \text{for all } i = 1, \ldots, \ell$$

(2) For a given profile P determine a profile P' such that $\mathcal{F}(P') - \mathcal{F}(P)$ is minimal with respect to the condition

$$\exists P_1, \ldots, P_\ell . P' \in top_k(\mu(\mathcal{F}(P_i), \cdot)) \qquad \text{for all } i = 1, \ldots, \ell$$

In the recruiting application area for a given profile of a job seeker a gap query determines additional skills that should be obtained by some training or

$\mathcal{F}_1 = \{C_1\}$
$\mathcal{F}_2 = \{C_2, C_1\}$
$\mathcal{F}_3 = \{C_3, C_1\}$
$\mathcal{F}_4 = \{C_3, C_2, C_1\}$
$\mathcal{F}_5 = \{C_4, C_3, C_2, C_1\}$

	$\mathcal{F}_1$	$\mathcal{F}_2$	$\mathcal{F}_3$	$\mathcal{F}_4$	$\mathcal{F}_5$
$\mathcal{F}_1$	1	$\frac{1}{5}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{10}$
$\mathcal{F}_2$	1	1	$\frac{1}{4}$	$\frac{5}{8}$	$\frac{1}{2}$
$\mathcal{F}_3$	1	$\frac{4}{5}$	1	$\frac{1}{2}$	$\frac{2}{5}$
$\mathcal{F}_4$	1	1	1	1	$\frac{4}{5}$
$\mathcal{F}_5$	1	1	1	1	1

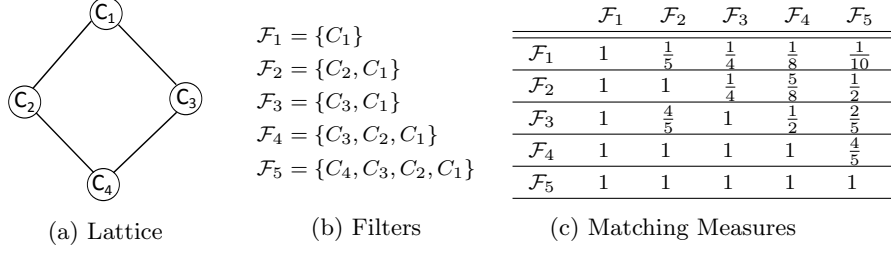(a) Lattice   (b) Filters   (c) Matching Measures

Figure 5.1: A lattice, its filters and corresponding matching values

education in order to increase chances on the job market, while a gap query for a requested job offer indicates additional incentives that should be added to make the job suitable for a larger number of highly skilled candidates. We will investigate such gap queries in Subsection 5.2 focusing again on case (1), while case (2) can be handled analogously.

5.1. Top-k Queries

For a set of profiles $\mathbb{P}$ defined by filters in a lattice $\mathcal{L}$ we denote by φ the conditions to be met by profiles in order to be selected, then $\mathbb{P}_\varphi$ denotes the set of profiles in $\mathbb{P}$ satisfying φ and $P_r \in \mathbb{P}$ is a *required* profile driving the selection by holding the conditions φ.

> For all $P \in \mathbb{P}_\varphi$ and $P' \in (\mathbb{P} - \mathbb{P}_\varphi)$, select λ profiles, where $\lambda \geq k$, out of the set of profiles $\mathbb{P}_\varphi$ such that P is selected and P' is not selected if $\mu(\mathcal{F}(P), \mathcal{F}(P_r)) > \mu(\mathcal{F}(P'), \mathcal{F}(P_r))$ and no subset of $\mathbb{P}_\varphi$ satisfies this property.

In order to obtain the best-k matching profiles (either job or applicant profiles) we first need to query for filters representing those profiles. In the sequel we permit to simply write $\mu(P', P)$ instead of $\mu(\mathcal{F}(P'), \mathcal{F}(P))$.

5.1.1. Preliminaries

Let $\mathcal{F}_r$ be a filter representing the required profile P_r. Then, consider l filters $\mathcal{F}_1, \ldots, \mathcal{F}_\ell \in \mathbb{F}$ representing profiles that satisfy φ, i.e. they match P_r with matching values above a threshold $t \in [0, 1]$, i.e. $\mu(\mathcal{F}_i, \mathcal{F}_r) \geq t$ for $i = 1, \ldots, \ell$. In addition, every filter $\mathcal{F}_i$ represents a finite number k_i of profiles $P_1^i, \ldots, P_{k_i}^i$ matching P_r, so $\mu(P_j^i, P_r) \geq t$ for all $i = 1, \ldots, \ell$ and $j = 1, \ldots, k_i$.

Then we have $\sum_{i=1}^{\ell} k_i = \lambda$ and $\sum_{i=1}^{\ell-1} k_i < k$. Furthermore, any non-selected filter $\mathcal{F}_{\ell+1}$ satisfies $\mu(\mathcal{F}_{\ell+1}, \mathcal{F}_r) < t$.

Example 5.1. Let $\mathcal{L}$ be a lattice with four elements—$\mathcal{L} = \{C_1, C_2, C_3, C_4\}$—where C_1 is the top element, C_4 is the bottom element, and C_2, C_3 are incomparable. This defines five filters $\mathbb{F} = \{\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3, \mathcal{F}_4, \mathcal{F}_5\}$, as shown in Figure 5.1 (a) and (b), respectively.

If we assign the following weights to the elements of $\mathcal{L}$, $w(C_1) = \frac{1}{10}, w(C_2) = \frac{2}{5}, w(C_3) = \frac{3}{10}$, and $w(C_4) = \frac{1}{5}$ and calculate the matching values $\mu(\mathcal{F}_i, \mathcal{F}_j)$ (for $1 \leq i, j \leq 5$), we obtain the result shown in Figure 5.1(c).

Assume a requested profile A and four given profiles $\{B, C, D, E\}$. Let A, B be represented by $\mathcal{F}_4$, C by $\mathcal{F}_2$ and D, E by $\mathcal{F}_3$, respectively using the filters in Figure 5.1. Then we obtain the matching values $\mu(B, A) = 1$, $\mu(C, A) = \frac{5}{8}$ and $\mu(D, A) = \mu(E, A) = \frac{1}{2}$.

If we choose $k = 3$, the final answer is $\{B, C, D, E\}$ with $\lambda = 4$ and with $t = \frac{1}{2}$, i.e. all profiles are in the answer to the query $top_3(\mu(\cdot, \mathcal{F}(A)))$. If we choose $k = 2$, the answer is $\{B, C\}$ with $\lambda = 2$ and with $t = \frac{5}{8}$, i.e. the answer to the query $top_2(\mu(\cdot, \mathcal{F}(A)))$ contains just the profiles B and C.

In order to obtain the best l filters satisfying φ, we first need to know the minimum matching value representing l filters. Thus, we start by selecting any t. If less than ℓ solutions are found, we decrease t. If more than ℓ solutions are found, we increase t. The search stops when the ℓ filters satisfying $\mu(\mathcal{F}_i, \mathcal{F}_r) \geq t$ for $i = 1, \ldots, \ell$ are found.

With the obtained value t we query for the related k profiles P_g where $\mu(P_g, P_r) \geq t$. This assumes to be given the matching measures between all filters in $\mathcal{L}$ and ultimately, between all profiles represented by filters.

As seen in Example 5.1, matching values between filters define a matrix, where columns represent the required filters $\mathcal{F}_r$ and rows represent the given filters $\mathcal{F}_g$ as depicted in Figure 5.1(c). Obtaining the solutions in $top_k(\mu(\cdot, \mathcal{G}))$ or $top_k(\mu(\mathcal{F}, \cdot))$ refers to one column or one row in this matrix, respectively. The process is analogous although, the perspective is different. While reading the measures from the columns provides the so called *fitness* between profiles $\mu(P_g, P_r)$, the measures read from rows are the inverted measure $\mu(P_r, P_g)$.

If we focus on columns, when querying for a particular $\mathcal{F}_r$, there would be $\mathcal{F}_i$ $(i = 1, \ldots, \ell)$ where $\mu(\mathcal{F}_i, \mathcal{F}_r) \geq t$. We assume all elements are in total order according to the $\leq$ relation of $\mu(\mathcal{F}_i, \mathcal{F}_r)$. The advantage is that when searching for any given ℓ and t we only need to point to the right element in the column and search for the next consecutive $\ell - 1$ elements in descending order of μ. The process is analogous if searching on rows.

5.1.2. Data Structures

We explain next how we organize profiles. We first assume an identification label ρ_i representing the number i of rows and, σ_i representing the number i of columns in $\mathcal{M}$ where $i > 0$.

Definition 5.1. Given a *required* filter $\mathcal{F}_r$, for every element μ_i in column σ_i in $\mathcal{M}$ representing $\mu(\mathcal{F}_{g_x}, \mathcal{F}_r)$ there is a *profile record*

$$(\mu_i, n_i^>, n_i^=, n_i^<, next, prev, p)$$

describing the matching profiles P_g where:

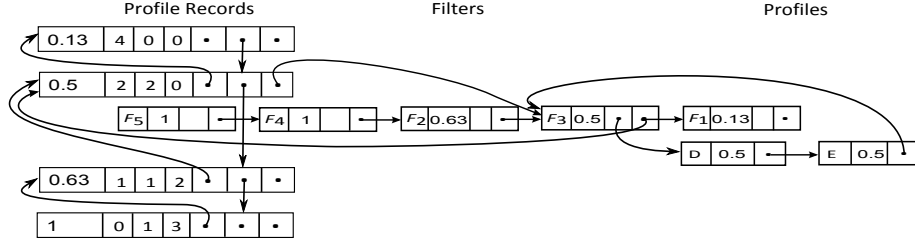- $n_i^>$ denotes the number of profiles P_g where $\mu(P_g, P_r) > \mu_i$,

Figure 5.2: Linked list of matching measures

- $n_i^=$ denotes the number of profiles P_g where $\mu(P_g, P_r) = \mu_i$,

- $n_i^<$ denotes the number of profiles P_g where $\mu(P_g, P_r) < \mu_i$,

- *next* is a reference to the next matching value in σ_i where $\mu(\mathcal{F}_{g_{(x+1)}}, \mathcal{F}_r) \geq \mu_i$,

- *prev* is a reference to the next matching value in σ_i where $\mu(\mathcal{F}_{g_{(x-1)}}, \mathcal{F}_r) \leq \mu_i$,

- *p* is a reference to a linked-list of filters matching $\mathcal{F}_r$.

The numbers $n_i^>, n_i^=, n_i^<$ are significantly important when determining the number of profiles represented by a filter without actually querying for them, i.e., if $(n_i^> + n_i^=) \geq k$ for a given μ_i we get all profiles needed.

References *Next* and *Prev* make possible to track the following greater or smaller matching value by following the references. Every profile record contains additionally a reference p to the related profiles in a σ_i column where they are organized in a transitive closure structure, ordered by the $\leq$ elements of μ.

Note that as matching values are pre-computed, the approach also works for simultaneous (or aggregated) matching with multiple matching measures.

Example 5.2. Figure 5.2 shows a representation of profile records corresponding to $\mathcal{F}_4$ (filter representing profile A) from Example 5.1. Note that only the relation to filters and profiles of $\mu = 0.5$ are shown in here in order to simplify the graph.

Organizing data in this rings and spiders structure known from network databases leads to a better performance on the search for the top-k elements. Starting by fetching a column σ_i and together with $n_i^>$, $n_i^=$, $n_i^<$ calculate the profile records needed to get the k matching profiles. Then, following the ordered linked-list structure of filters, and profiles afterward, until the $k(\lambda)$ elements are found. The definition of profile records on rows ρ_i of $\mathcal{M}$ is analogous to Definition 5.1.

ID	Required Filter	Fitness	Greater Fitness	Equal Fitness	Lesser Fitness	Next ID
1	$\mathcal{F}_4$	1	0	1	3	2
2	$\mathcal{F}_4$	0.63	1	1	2	3
3	$\mathcal{F}_4$	0.50	2	2	0	4
4	$\mathcal{F}_4$	0.13	4	0	0	null

(a) Relation *ProfileRecords*

ID	Required Filter	Required Profile	Given Filter	Given Profile	Fitness	Next ID
1	$\mathcal{F}_4$	A	$\mathcal{F}_4$	B	1	null
2	$\mathcal{F}_4$	A	$\mathcal{F}_2$	C	0.63	null
3	$\mathcal{F}_4$	A	$\mathcal{F}_3$	D	0.5	4
4	$\mathcal{F}_4$	A	$\mathcal{F}_3$	E	0.5	null

(b) Relation *MatchingProfiles*

Figure 5.3: Example of relations *ProfileRecords* and *MatchingProfiles*

5.1.3. Algorithmic Solution

We now present an implementation of the matrix of matching values, profile records and linked-list of profiles in a relational database schema, and an algorithm that implements our definition of top-k queries. Our implementation approach is designed on a relational database schema for the storage and maintenance of filters, profiles and matching measures of an instance of $\mathcal{L}$.

The relational schema is composed of 9 relations although, we present only two relations: *ProfileRecords* and *MatchingProfiles* that describe profile records as in Definition 5.1 and the linked-lists of matching profiles, respectively. Figure 5.3 shows an example of the relations representing the elements involved in Example 5.1.

For every *RequiredFilter* in *ProfileRecords* there is a number of matching measure, that represent the number of elements per column σ_i in $\mathcal{M}$. The attribute *NextID* in *ProfileRecords* is a reference to another tuple (ID) in the relation defining the $\leq$ relation of elements of *Fitness*. Attributes *GreaterFitness*, *EqualFitness* and *LesserFitness* represent, respectively, the elements $n_i^>$, $n_i^=$, $n_i^<$ from profile records as in Definition 5.1.

In turns, for every *RequiredFilter*($\mathcal{F}_r$) in *ProfileRecords* there is a finite number of *GivenFilters* in *MatchingProfiles* that match the requirements in $\mathcal{F}_r$. The attribute *NextID* is a reference to another tuple (ID) in the relation defining the $\leq$ relation of elements of *Fitness*. Note that a value of *null* represents the end of the list for the *GivenFilter*.

Filters in a lattice $\mathcal{L}$ represent the properties of profiles via the hierarchical dependency of concepts in $\mathcal{L}$. Thus, for every *required* profile P_r in $\mathbb{P}$ there is a *required* filter $\mathcal{F}_r \in \mathcal{L}$ representing the profile. Then retrieving the top-k candidate profiles for a required filter from our relational schema is mainly performed by querying on relations *ProfileRecords* and *MatchingProfiles*.

Algorithm: Top-k
Input:
    required filter: $\mathcal{F}_r$, number of matching profiles: k, matching threshold: μ
Output:

```
        matching profiles: $P_{g_1}, \ldots, P_{g_k}$, measures: $\mu_1, \ldots, \mu_k$
Begin
1    CREATE relation $Results = (\text{GivenProfile, Fitness, NextID})$
2    $(\text{fitness, count, nextid}) := \pi_{(3,5,7)}\big(\sigma_{\substack{(2=\mathcal{F}_r, \\ max(Fitness))}} \; (ProfileRecords) \big)$
3    WHILE (count $< k$) OR (fitness $\geq \mu$) DO
4        $Results \leftarrow \pi_{(5,6,7)}\big(\sigma_{\substack{(6=\text{fitness}, \\ 2=\mathcal{F}_r)}}(MatchingProfiles)\big)$
5        next := $Results.NextID$
6        WHILE (next IS NOT NULL) DO
7            $Results \leftarrow \pi_{5,6,7}\big(\sigma_{(2=\mathcal{F}_r)}(MatchingProfiles) \underset{1=3}{\bowtie} Results\big)$
8        END WHILE
9        $(\text{fitness, total, nextid}) := \pi_{(3,5,7)}\big(\sigma_{(1=\text{nextid})} \; (ProfileRecords) \big)$
10         count := count + total
11    END WHILE
12    RETURN $(\pi_{1,2}(Results))$
End
```

The algorithm Top-k returns an ordered list of top-k profiles matching a given filter. We use relational algebra notation thus, σ, π and $\bowtie$ are the *selection*, *projection* and *natural join* operators, respectively. Numeric subscripts are used to denote relation attributes. For instance, $\pi_1(MatchingProfiles)$ is the projection of attribute *ID* of relation *MatchingProfiles*.

The algorithm accepts as inputs: the required filter $\mathcal{F}_r$, the number k of matching profiles and the minimum matching value μ to search for. The outputs are: the k matching profiles $P_{g_1}, \ldots, P_{g_k}$ and their matching measures $\mu_1, \ldots, \mu_k$.

With $\mathcal{F}_r$, the algorithm fetches the tuple with the greatest value of *Fitness* in *ProfileRecords* (line 2) and follows the references on *NextID* (line 9) until the k tuples are reached or $\mu(\mathcal{F}_g, \mathcal{F}_r) < t_i$ (line 3). Then, for every $\mathcal{F}_g$ in *MatchingProfiles*, the algorithm queries on the linked-list of profiles (lines 6-8) and appends them in the temporary relation *Results* (line 7). Note that by using "fitness $\geq \mu$" in line 3, we include the $\lambda - k$ elements as in Definition **??**. The algorithm finishes by returning the elements of *GivenProfile* and *Fitness* on the tuples of *Results*.

An implementation of B-Tree indexes on elements of *Fitness* (*ProfileRecords* and *MatchingProfiles*) in order to access the sorted elements, as well as indexes on *RequiredFilter* (*ProfileRecords* and *MatchingProfiles*) for random access is expected to improve performance. The implementation of a parallel processing on the search of matching profiles given the required profile records by calculating $(n_i^> + n_i^=)$ is another point of improvement.

5.2. Gap Queries

Let P be a profile and let $\mathcal{F} = \mathcal{F}(P)$ be its representing filter in the lattice $\mathcal{L}$. Assume that a matching measure μ on $\mathcal{L}$ is fixed. We concentrate on the following case of gap queries—precisely (k, ℓ)-*gap queries* for fixed positive integers k and ℓ—to determine a profile P' such that $\mathcal{F}(P') - \mathcal{F}$ is minimal with

respect to the condition

$$\exists P_1, \ldots, P_\ell . P' \in top_k(\mu(\cdot, \mathcal{F}(P_i))) \qquad \text{for all } i = 1, \ldots, \ell.$$

Here and in the following minimality always refers to set inclusion. We proceed as follows:

(1) Determine $top_\ell(\mu(\mathcal{F}, \cdot)) = \{P_1, \ldots, P_m\}$ with $m \geq \ell$. Let $\mathcal{G}_i = \mathcal{F}(P_i)$ for $i = 1, \ldots, m$ denote the filters corresponding to the profiles resulting from this top-ℓ query.

(2) Determine $top_k(\mu(\cdot, \mathcal{G}_i)) = \{P_1^i, \ldots, P_{k_i}^i\}$ with $k_i \geq k$ for all $i = 1, \ldots, m$.

(3) Let $\mathcal{D}_i^{(j)} = \mathcal{F}(P_i^j) - \mathcal{F}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, k_i$. Each such $\mathcal{D}_i^{(j)}$ determines a gap with respect to the top-k query with $\mathcal{G}_i$. As we are only interest in a minimal gap we discard all those pairs (i, j), for which another i' exists with $\mathcal{D}_i^{(j)} \supseteq \mathcal{D}_{i'}^{(j)}$.

(4) Now select ℓ different indices $i_1, \ldots, i_\ell$ out of $\{1, \ldots, m\}$ and for each i_x select k different indices $j_1(i_x), \ldots, j_k(i_x)$ out of $\{1, \ldots, k_{i_x}\}$ such that $\mathcal{D}_{i_x}^{(j_y(i_x))}$ (for $x = 1, \ldots, \ell$ and $y = 1, \ldots, k$) has not been discarded in (3).

(5) For each such selection of indices the union $E = \bigcup_{i=1}^{\ell} \mathcal{D}_{i_x}^{j_h(i_x)}$ defines a *gap candidate*.

Due to our construction each gap candidate E satisfies the condition for the profile $P' = P \cup E$ there exists at least ℓ profiles such such P' appears in the result of the top-k query for all these profiles. This gives rise to the following main result.

Theorem 5.1. *Each minimal gap candidate E is a result for the (k, ℓ) gap query for profile P.*

6. Enriched Matching

In this section we extend the matching theory from Section 3. So far the developed theory is grounded in matching measures that are defined on pairs of filters in a lattice $\mathcal{L}$. The lattice $\mathcal{L}$ is derived from a TBox of a knowledge base representing the application knowledge. Thus $C_1 \leq C_2$ holds in $\mathcal{L}$, if C_1 and C_2 are concepts satisfying $C_1 \sqsubseteq C_2$ in the TBox. Logically this means that the implication $C_1(x) \Rightarrow C_2(x)$ holds for all individuals x. However, as already mentioned in the introduction the presence of a particular concept in a profile may only partially imply the presence of another one. Therefore, we capture such additional partial implications by enriching the lattice by extra weighted edges. Then we investigate how this can be used to find the most suitable matchings. As these extra edges can introduce cycles, the notion of filter is no longer applicable. Therefore, we use a graph-based approach to extend the given and requested profiles. We show that this approach corresponds to the
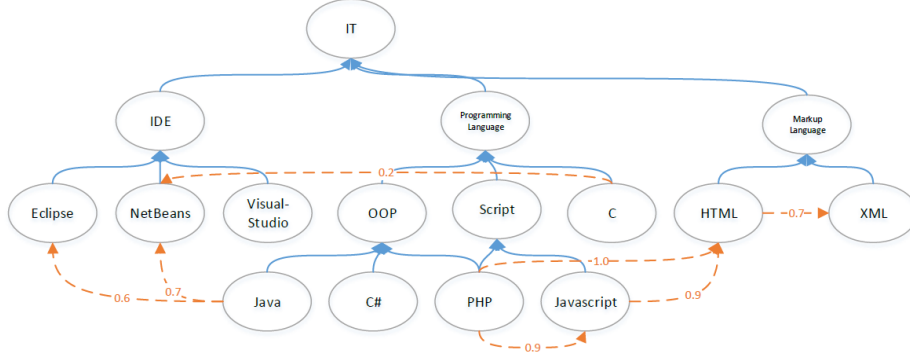
Figure 6.1: Fragment of a graph with lattice edges (solid) and extra edges (dashed) and assignment of degrees.

definition of fuzzy filters [33]. We also provide an interpretation in probabilistic logic with maximum entropy semantics [34, 35].

For the extended matching theory we could investigate again the problems of matching learning and querying that we dealt with in Sections 4 and 5 for the case of the filter-based matching theory. However, such investigations are still subject to ongoing research and beyond the scope of this article.

6.1. Maximum Length Matching

Let $G = (V, E)$ be a directed weighted graph where V is a finite non-empty set of nodes and $E = \{E_O \cup E_E\} \subseteq V \times V$ is a set of edges. Each node represents a property and a directed edge $e = (v_i, v_j) \in E_O$ expresses that v_i is a specialization of v_j (write also $v_i \leq v_j$ for this). Therefore, we claim that the subgraph (V, E_O) still defines a lattice with the partial order $\leq$, and we call edges in E_O *lattice edges*. A directed edge $e = (v_i, v_j) \in E_E$ (called *extra edge*) represents a conditional dependency between the v_i and v_j, namely that property v_i may imply property v_j. For node $s, t \in V$ let $p_E(s, t)$ denote the set of all directed paths from s to t.

Let $d : E \to (0, 1]$ be a function with $d(e) = 1$ for all lattice edges $e \in E_O$. For all extra edges $e \in E_E$ the value $d(e)$ expresses the degree (or conditional probability) to which property v_j holds, when property v_i is given. See Figure 6.1 for a fragment of such a graph.

Definition 6.1. A *fuzzy set* on S is a mapping $f : S \to [0, 1]$. A fuzzy set is called empty if f is identically zero on S. For $t \in [0, 1]$ the set $f_t = \{s \in S | f(s) \geq t\}$ is called a *level subset* of the fuzzy set f.

If $(S, \preceq)$ is a lattice and f is a fuzzy set on S, then f is called a *fuzzy filter*, if for all $t \in [0, 1]$, f_t is either empty or a filter of S.

Note that as S is a finite set, we can define a fuzzy set by enumerating all elements in S with their assigned values (called the grade of membership) if

that value is greater than zero, i.e. $f = \{(s, f(s)) | s \in S \text{ and } f(s) > 0\}$. For fuzzy sets f, g in S, we define their intersection and their union as $(f \cap g)(s) := \min\{f(s), g(s)\}$ and $(f \cup g)(s) := \max\{f(s), g(s)\}$ for all $s \in S$, respectively.

A weighting function w on a lattice $\mathcal{L}$ can be easily extended to fuzzy sets on $\mathcal{L}$ by $w(f) = \sum_{C \in \mathcal{L}} f(C) \cdot w(\{C\})$. If f, g are fuzzy filters on $\mathcal{L}$, then the matching measure μ defined by the weighting function f naturally extends to

$$\mu(f, g) = \frac{w(f \cap g)}{w(g)}.$$

We will now show how to derive fuzzy filters from graphs as above, i.e. lattices that are extended by extra edges. For this we extend the representing filters of a set of properties $O \subseteq V$ to fuzzy sets. The extension $\widehat{O}$ of O with respect to E is defined as the set of all the properties $v \in V$ that are reachable from O via a directed path containing lattice and extra edges, and the grade of membership assigned to each reachable v is the length of the longest path between O and v, i.e.

$$\widehat{O} = \{(v, \gamma_v) | v \in V \text{ and } \exists v' \in O : |p_E(v', v)| \geq 1 \text{ and } \gamma_v = \max_{v' \in O, p \in p_E(v', v)} length(p)\},$$

where $length(p) = \prod_{i=1}^{n-1} d((v_i, v_{i+1}))$ is the product of the degrees assigned to the edges on the path p. If the edge weights are interpreted as probabilities, then the length of longest path from v' to v has to be interpreted as the conditional probability of property v under the condition v'.

In general, finding the longest path between two nodes in a graph is a NP-hard problem. In our case, however, the length of a path is defined as the product of the degrees of the edges on the path, so it is $exp(-L)$ with

$$L = -\max_{p \in p_E(v', v)} \log\{\prod_{i=1}^{n-1} d((v_i, v_{i+1}))\} = \min_{p \in p_E(v', v)} \sum_{i=1}^{n-1} -\log d((v_i, v_{i+1})).$$

Moreover, as $d(v_i, v_{i+1}) \in (0, 1]$ holds for all edges, we have $-\log d((v_i, v_{i+1})) \geq 0$, i.e. the determination of the longest paths can be reduced to a single-source shortest path problem with non-negative edge weights, which can be solved with Dijkstra's algorithm in $O(|E| + |V| \log |V|)$ time [37].

Theorem 6.1. *Let $(\mathcal{L}, E_O \cup E_E)$ be a directed graph with degree function d extending the lattice $(\mathcal{L}, \leq)$, and let $O \subseteq \mathcal{L}$ be a non-empty subset. Then the extension $\widehat{O}$ of O with respect to E is a fuzzy filter in $\mathcal{L}$.*

Proof. For $t \in [0, 1]$ we have that $\widehat{O}_t = \{s \in \mathcal{L} \mid \widehat{O}(s) \geq t\}$ is a filter in $\mathcal{L}$ if for all $s, s' \in \mathcal{L}$ with $s \leq s'$ whenever $s \in \widehat{O}_t$ holds, then also $s' \in \widehat{O}_t$ holds, i.e. if $\widehat{O}(s) \geq t$, then $\widehat{O}(s') \geq t$.

Let s be in $\widehat{O}_t$ and let $p_{a,s} = (a = s_{i_1}, s_{i_2}, \ldots, s_{i_k} = s)$ be a path of maximal between O and s. We have to show that if an $s' \in \mathcal{L}$ is a generalization of s, i.e. $s \leq s'$, then a path in $p_{a,s'}$ exists such that $length(p_{a,s'}) \geq length(p_{a,s})$.

As $s \leq s'$ holds, there exists a path $p_{s,s'} = (s = s_{j_1}, s_{j_2}, \ldots, s_{j_l} = s')$ from s to s' of length 1. If $p_{a,s}$ and $p_{s,s'}$ do not have any node in common except s, we

can concatenate them to a path $p_{a,s'} = (a = s_{i_1}, \ldots, s_{i_k}, s_{j_1}, \ldots sj_l = s')$, the length of which is $length(p_{a,s}) \cdot 1 \geq length(p_{a,s})$.

Otherwise, proceed by induction. Let t be the first common node. Then take the paths $p_{a,t} = (a = s_{i_1}, \ldots, s_{i_x} = t)$ and the $p_{t,s'} = (t = s_{j_y}, s_{j_{y+1}}\ldots, s_{j_l} = s')$. We have $length(p_{a,t}) \geq length(p_{a,s})$, $d(e) \leq 1$ for all $e \in E$ and $length(p_{t,s'}) = 1$. If these paths have only t in common, then we can concatenate them, otherwise apply the induction hypothesis.

$\square$

Example 6.1. For the graph in Figure 6.1 take the following sets of properties

$$O_1 = \{Java, Netbeans, XML\} \text{ and } O_2 = \{Java, PHP, Eclipse\}$$

These generate the following fuzzy filters:

$$\widehat{O_1} = \{(Java, 1.0), (Netbeans, 1.0), (XML, 1.0), (OOP, 1.0), (PL, 1.0),$$
$$(IT, 1.0), (IDE, 1.0), (ML, 1.0)\}$$

and $\quad \widehat{O_2} = \{(Java, 1.0), (PHP, 1.0), (Eclipse, 1.0), (OOP, 1.0), (PL, 1.0),$
$$(IT, 1.0), (Script, 1.0), (IDE, 1.0), (Netbeans, 0.7),$$
$$(Javascript, 0.9), (HTML, 1.0), (ML, 1.0), (XML, 0.7)\}$$

This gives rise to the intersection fuzzy filter

$$\widehat{O_1} \cap \widehat{O_2} = \{(Java, 1.0), (OOP, 1.0), (PL, 1.0), (IT, 1.0), (IDE, 1.0),$$
$$(Netbeans, 0.7), (XML, 0.7), (ML, 1.0)\}$$

Assuming a weighting function w that assigns the same weight to all elements we obtain the matching value $\mu(\widehat{O_1}, \widehat{O_2}) = \frac{7.4}{8}$.

6.2. Probabilistic Matching

The lattice and the extra edges can be handled from an information-theoretical point of view with probabilistic logic programs [35], or from set theoretic point of view with probabilistic models [34] as well. Here we adapt the latter approach using apply the maximum entropy model to define a probabilistic matching method.

6.2.1. Preliminaries

Let Θ be a finite set. Let $R := \{a_1, \ldots, a_l\}$ be a set of subsets of the power set $\mathcal{P}(\Theta)$ of Θ, namely $a_i \in \mathcal{P}(\Theta)$, $i = 1, \ldots, l$. The elements of R are called *events*.

Definition 6.2. Let X be some set. Let $\mathcal{A}$ be a subset of $\mathcal{P}(X)$. $\mathcal{A}$ is a *σ-algebra* over X, denoted by $\mathcal{A}(X)$, if:

- $X \in \mathcal{A}$,
- if $Y \in \mathcal{A}$, then $(X \setminus Y) \in \mathcal{A}$, and

- if $Y_1, Y_2, \ldots$ is a countable collection of sets in $\mathcal{A}$, then their union $\bigcup\limits_{n=1}^{\infty} Y_n$ is in $\mathcal{A}$ as well.

The set of full conjunction over R is given by $\Omega := \left\{ \bigcap\limits_{i=1}^{l} e_i | e_i \in \{a_i, \neg a_i\} \right\}$, where $a_i \in \mathcal{P}(\Theta)$, $i = 1, \ldots, l$, and $\neg a_i = \Theta \setminus a_i$. It is well known fact that the 2^l elements of Ω are mutually disjoint and span the set R (any a_i can be expressed by a disjunction of elements of Ω). Therefore, the smallest σ-algebra $\mathcal{A}(R)$ that contains R is identical to $\mathcal{A}(\Omega)$. For that reason we restrict the set of *elementary events* (set of possible worlds) to Ω instead of the underlying Θ.

Definition 6.3. A *measurable space* $(\Omega, \mathcal{A})$ over a set $R := \{a_1, \ldots, a_l\}$ is defined by

$$\Omega := \left\{ \bigcap\limits_{i=1}^{l} e_i | e_i \in \{a_i, \neg a_i\} \right\} \qquad \text{and} \qquad \mathcal{A} = \mathcal{A}(\Omega) = \mathcal{P}(\Omega).$$

If $(\Omega, \mathcal{A})$ is a measurable space over R with $\Omega = \{\omega_1, \ldots, \omega_n\}$, a *discrete probability measure* P (P-model) is an assignment of non-negative numerical values to the elements of Ω, which sum up to 1, i.e. $p_i = P(\omega_i) \geq 0$ for $i = 1, \ldots, n$ and $\sum a_i = 1$.

The n-tuple $\vec{p} = (p_1, \ldots, p_n)$ is called a *probability vector* (P-vector). W_Ω (respectively V_Ω) denotes the set of all possible P-models (P-vectors) for $(\Omega, \mathcal{A})$.

Definition 6.4. Let $(\Omega, \mathcal{A})$, $P \in W_\Omega$ $a, b \in \mathcal{A}, P(a) > 0$, and $[l, u] \subseteq [0, 1]$. A *sentence* in $(\Omega, \mathcal{A})$ is a term $\langle P(b|a) = \delta; \delta \in [l, u] \rangle$ or $P(b|a)[l, u]$ (also use $P(a) = P(a|\Omega)$ as a shortcut), where $P(b|a) = P(a \cap b)/P(a)$ denotes the conditional probability of the event b given a. Use $P(a) = P(a|\Omega)$ as a shortcut. The sentence is *true in* $P \in W_\Omega$ iff $P(b|a) \in [l, u]$. Otherwise it is *false*.

A sentence $P(b|a)[l, u]$ defines two inequalities, namely $P(b|a) \leq u$ (be less than the upper bound) and $P(b|a) \geq l$ (be greater than the lower bound). These inequalities can be further transformed in the following way:

$$P(b|a) \leq u \Leftrightarrow P(a \cap b) \leq u * P(a) \Leftrightarrow P(a \cap b) \leq u * (P(a \cap b) + P(a \cap \neg b))$$
$$P(b|a) \geq l \Leftrightarrow P(a \cap b) \geq l * P(a) \Leftrightarrow P(a \cap b) \geq l * (P(a \cap b) + P(a \cap \neg b))$$

Rearranging the inequalities and using the elementary probabilities $p_i, i = 1, \ldots, n$ we obtain

$$P(b|a) \leq u \Leftrightarrow (1 - u) \cdot \sum_{i : w_i \in a \cap b} p_i + u \cdot \sum_{j : w_j \in a \cap \neg b} p_j \geq 0$$
$$P(b|a) \geq l \Leftrightarrow (1 - l) \cdot \sum_{i : w_i \in a \cap b} p_i - l \cdot \sum_{j : w_j \in a \cap \neg b} p_j \geq 0$$

Note, that if $u = 1$ (respectively, $l = 0$), then the first (second) inequality is always satisfied as $p_i \geq 0$.

Definition 6.5. Let $DB := \{c_1, \ldots, c_m\}$ be a set of m sentences in $(\Omega, \mathcal{A})$. W_{DB} is defined as the set of all P-models $P \in W_\Omega$ in which $c_1, \ldots, c_m$ are true. We call $c_1, \ldots, c_m$ *constraints* on W_Ω, and W_{DB} denotes the set of all elements of W_Ω that are consistent with the constraints in DB.

If W_{DB} is empty, the information in DB is inconsistent. If W_{DB} contains more than one element, the information in DB is incomplete for determining a single P-model.

6.2.2. Maximum entropy model

If W_{DB} contains more than one element, the information in DB is incomplete for determining a single P-model. Therefore, we must add further constraints to the system to get a unique model. It was shown in [35] that the maximum entropy model adds the lowest amount of additional information between single elementary probabilities to the system. Moreover, the maximum entropy model also satisfies the principle of indifference and the principle of independence:

- The *principle of indifference* states that if we have no reason to expect one event rather than another, all the possible events should be assigned the same probability.

- The *principle of independence* states the if the independence of two events a and b in a P-model ω is given, any knowledge about the event a does not change the probability of b (and vice verse) in ω, formally $P(b|a) = P(b)$.

In order to obtain the consistent P-model to a DB that has the maximum entropy, we have to solve the following linear optimization problem:

Definition 6.6. Let $DB := \{c_1, \ldots, c_m\}$ be a set of m sentences in $(\Omega, \mathcal{A})$ with $\Omega = \{\omega_1, \ldots, \omega_n\}$. Let W_{DB} (respectively V_{DB}) be the set of all P-models $P \in W_\Omega$ (P-vectors $\vec{p} \in V_\Omega$) in which $c_1, \ldots, c_m$ are true.

The maximum entropy problem is to obtain

$$\max_{\vec{v} = (p_1, \ldots, p_n) \in [0,1]^n} -\sum_{i=1}^{n} p_i \log p_i$$

subject to the following conditions:

$$p_i \geq 0 \text{ for } i = 1, \ldots, n \qquad \text{and} \qquad \sum_{i=1}^{n} p_i = 1$$

$$(1-u) \cdot \sum_{i:w_i \in a \cap b} p_i + u \cdot \sum_{j:w_j \in a \cap \neg b} p_j \geq 0 \qquad \text{for all } c = P(b|a)[l, u] \in DB, l > 0$$

$$(1-l) \cdot \sum_{i:w_i \in a \cap b} p_i - l \cdot \sum_{j:w_j \in a \cap \neg b} p_j \geq 0 \qquad \text{for all } c = P(b|a)[l, u] \in DB, u < 0$$

In the following let $me[DB]$ denote the P-model that solve the maximum entropy problem, provided such model exists.

Definition 6.7. Let DB be a set of sentences and let $c = \langle P(b|a)[l, u]\rangle$ be a sentence. We say that c is a *maximum entropy consequence* of DB, denoted by $DB \parallel\sim^{me} c$, iff either DB is inconsistent or $me[DB](b|a) \in [l, u]$ holds.

A *probabilistic query* is an expression $QP_{DB}(b|a)$ where a and b are two events, i.e. $a, b \in \mathcal{A}$, and DB is a set of sentences.

Let DB be a set of sentences and let $QP(b|a)$ be a probabilistic query. The *answer to the query* is $\delta = me[DB](b|a) = \dfrac{me[DB](a \cap b)}{me[DB](a)}$, if $DB \parallel\sim^{me}$ $P(a)(0, 1]$. Otherwise $\delta = -1$.

The query means what is the probability of b given a with respect to DB. An answer $\delta = -1$ means that the set $DB \cup \{P(a)(0, 1]\}$ is inconsistent.

6.2.3. Probabilistic Matching

We now show how the matching problem can be expressed with probabilistic queries. For this let again $G = (V, \{E_O \cup E_E\})$ be a directed graph with degree function d. We construct DB from G in the following way:

(1) Assign a new set (event) a_v to each node v of G which contains property sets that include the property v.
(2) For each lattice edge $(v_i, v_j) \in E_O$ add a new sentence s_{ij} to the DB in the form $P(a_{v_j}|a_{v_i})[1, 1]$. The sentence means if a property set contains v_i—i.e., is an element of a_{v_i}—then it also contains v_j—is an element of a_{v_j}) as well.
(3) Then, for every extra edge $(v_i, v_j) \in E_E$ we also add a new statement in the form of $P(a_{v_j}|a_{v_i}) = [l, u]$. Here the degree of an edge can be handled in two different ways. In the first approach, let the lower bound of the interval l be equal to the degree $d(v_i, v_j)$ of the edge, and let the upper bound of the interval u be equal to 1. In the second approach, let $l = u = d(v_i, v_j)$. The latter is the stricter approach, as it adds constraints to the upper bounds as well.

A property set $A = \{v_1, ..., v_n\}$ is translated into the event $ev(A) = a_{v_1} \cap \cdots \cap a_{v_n}$. The matching value $\mu(A_g, A_r)$ for a given profile A_g with respect to a requested profile A_r is the result of the probabilistic query $QP(ev(A_r)|ev(A_g))$. In this way a matching value is interpreted as the probability that the given profile is a fit for the requested one, provided that the constructed DB is consistent.

7. Conclusions

In this article we first developed a filter-based matching theory with profiles that are defined via a knowledge base. For the knowledge base we assume some TBox and an associated ABox expressed in some suitable description logic similar to DL-LITE, $\mathcal{SROIQ}$ or OWL-2, though less expressive languages might suffice. From the knowledge base we define a lattice, such that profiles give rise

to filters in this lattice. This is the basis for the definition of weighted matching measures on pairs of such profiles. We showed that the theory is rather powerful, as fuzzy and probabilistic extensions can be captured.

As the theory cannot tell which weights would be appropriate for a particular application, we investigated the problem, how human-made matchings could be exploited to learn a suitable weighting function. We could show that under some mild assumptions—plausibility constraints that a human-defined matching should obey in order to exclude bias—a ranking-preserving matching measure exists, which can be used for the implementation of weight maintenance procedures. Ranking preservation takes all rankings with respect to fixed requested profiles and all rankings for fixed given profiles in connection with relevance classes into account. However, a total preservation of all relationships in the human-defined matchings is not possible. This may indicate that either the set of plausibility constraints requires extensions or the knowledge base needs to be enlarged. This open problem of adjustment of the matchings to unbiased human expertise requires further investigation. This also extends to the simultaneous capture of matchings from several human experts and the detection of probabilistic matching measures.

We further investigated the efficient implementation of matching queries, which requires to address top-k queries on knowledge bases with embedded computation of matching values to evaluate the ranking criterium. This problem can be solved by pre-computed matching measures, which is efficient, as updates to the weighting functions or the TBox are considered to appear not overly often. Furthermore, in practice usually only matching values above a rather high threshold are considered to be relevant. Consequences for the system architecture of a matching system are the use of relational databases for the storage of profiles, i.e. the ABox, and the pre-computed matching values, the realisation of a querying engine on top of this database, and the coupling of updates to the matching measures with the TBox and the learning algorithms.

For gap queries we adopted a rather pragmatic approach computing an extension of a given profile that will appear in the result of a top-k matching query for at least ℓ requested profiles (or the other way round). With respect to the application area of job recruiting this defines a suitable extension of the profile that will improve chances of successful matching. This is of interest for targeted decisons concerning further training and education. However, we believe that gap queries require further investigation, as it should also be taken into account how likely it is that a profile can be improved, in other words how close the necessary extensions are to the properties that are already in the profile.

Prototypes of these system components will be further developed by a company towards a knowledge-based matching system in the recruitment area. Additional components comprise crawlers for job offers and candidate profiles, extraction from natural language documents, and user-friendly update interfaces for maintaining the TBox that are suitable for the use by domain experts. For other application areas, e.g. matching in sport strategies, our research results present an open invitation to explore their suitability for these applications, which may uncover further open problems for research.

References

[1] A. L. Paoletti, J. Martinez-Gil, K.-D. Schewe, Extending knowledge-based profile matching in the human resources domain, in: Q. Chen, et al. (Eds.), Database and Expert Systems Applications (DEXA 2015) Part II, Vol. 9262 of LNCS, Springer, 2015, pp. 21–35.

[2] J. Martinez-Gil, A. L. Paoletti, G. Rácz, A. Sali, K.-D. Schewe, Maintenance of profile matchings in knowledge bases, in: L. Bellattreche, et al. (Eds.), Model and Data Engineering – 5th International Conference (MEDI 2016), Vol. 9893 of LNCS, Springer, 2016, pp. 132–141.

[3] A. Artale, D. Calvanese, R. Kontchakov, M. Zakharyaschev, The DL-Lite family and relations, J. Artif. Intell. Res. 36 (2009) 1–69.

[4] F. Baader, et al. (Eds.), The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2003.

[5] B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. F. Patel-Schneider, U. Sattler, OWL 2: The next step for OWL, Journal of Web Semantics 6 (4) (2008) 309–322.

[6] A. L. Paoletti, J. Martinez-Gil, K.-D. Schewe, Top-k matching queries for filter-based profile matching in knowledge bases, in: S. Hartmann, H. Ma (Eds.), Database and Expert Systems Applications (DEXA 2016) Part II, Vol. 9828 of LNCS, Springer, 2016, pp. 295–302.

[7] G. Rácz, A. Sali, K.-D. Schewe, Semantic matching strategies for job recruitment: A comparison of new and known approaches, in: M. Gyssens, G. R. Simari (Eds.), Foundations of Information and Knowledge Systems (FoIKS 2016), Vol. 9616 of LNCS, Springer, 2016, pp. 149–168.

[8] M. C. A. Klein, J. Broekstra, D. Fensel, F. van Harmelen, I. Horrocks, Ontologies and schema languages on the web, in: D. Fensel, et al. (Eds.), Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential, MIT Press, 2003, pp. 95–139.

[9] T. R. Gruber, Toward principles for the design of ontologies used for knowledge sharing?, Int. J. Hum.-Comput. Stud. 43 (5-6) (1995) 907–928.

[10] T. Falk, et al., Semantic-Web-Technologien in der Arbeitsplatzvermittlung, Informatik Spektrum 29 (3) (2006) 201–209.

[11] M. Mochol, H. Wache, L. J. B. Nixon, Improving the accuracy of job search with semantic techniques, in: W. Abramowicz (Ed.), Business Information Systems, 10th International Conference (BIS 2007), Vol. 4439 of Lecture Notes in Computer Science, Springer, 2007, pp. 301–313.

[12] M. Mochol, L. J. B. Nixon, H. Wache, Improving the recruitment process through ontology-based querying, in: E. P. Bontas Simperl, M. Hepp, C. Tempich (Eds.), Proceedings of the First International Workshop on Applications and Business Aspects of the Semantic Web (SEBIZ 2006), Vol. 226 of CEUR Workshop Proceedings, CEUR-WS.org, 2006.

[13] European distionary of skills and competences, http://www.disco-tools.eu.

[14] International standard classification of occupations, http://www.ilo.org/public/ english/bureau/stat/isco/isco08/ (2008).

[15] International standard classification of education, http://www.uis.unesco.org/ Education/Pages/international-standard-classification-of-education.aspx.

[16] M. Levandowsky, D. Winter, Distance between sets, Nature 234 (5) (1971) 34–35.

[17] N. Popov, T. Jebelean, Semantic matching for job search engines – a logical approach, Tech. Rep. 13-02, Research Institute for Symbolic Computation, JKU Linz (2013).

[18] J. Martínez-Gil, A. L. Paoletti, K.-D. Schewe, A smart approach for matching, learning and querying information from the human resources domain, in: M. Ivanovic, et al. (Eds.), New Trends in Databases and Information Systems – ADBIS 2016 Short Papers and Workshops, Vol. 637 of CCIS, Springer, 2016, pp. 157–167.

[19] B. Ganter, R. Wille, Formal concept analysis - mathematical foundations, Springer, 1999.

[20] B. Ganter, G. Stumme, R. Wille, Formal concept analysis: Theory and applications, Journal of Universal Computer Science 10 (8) (2004) 926.

[21] P. Cimiano, A. Hotho, G. Stumme, J. Tane, Conceptual knowledge processing with formal concept analysis and ontologies, in: P. W. Eklund (Ed.), Concept Lattices, Second International Conference on Formal Concept Analysis (ICFCA 2004), Vol. 2961 of Lecture Notes in Computer Science, Springer, 2004, pp. 189–207.

[22] B. Ganter, C. Meschke, A formal concept analysis approach to rough data tables, Transactions on Rough Sets 14 (2011) 37–61.

[23] D. Looser, H. Ma, K.-D. Schewe, Using formal concept analysis for ontology maintenance in human resource recruitment, in: F. Ferrarotti, G. Grossmann (Eds.), Ninth Asia-Pacific Conference on Conceptual Modelling (APCCM 2013), Vol. 143 of CRPIT, Australian Computer Society, 2013, pp. 61–68.

[24] J. Martínez-Gil, An overview of knowledge management techniques for e-recruitment, JIKM 13 (2).

[25] T. Tran, D. Q. Phung, S. Venkatesh, Modelling human preferences for ranking and collaborative filtering: a probabilistic ordered partition approach, Knowl. Inf. Syst. 47 (1) (2016) 157–188.

[26] R. Zhang, M. Gao, X. He, A. Zhou, Learning user credibility for product ranking, Knowl. Inf. Syst. 46 (3) (2016) 679–705.

[27] K. Chakrabarti, M. Ortega-Binderberger, S. Mehrotra, K. Porkaew, Evaluating refined queries in top-k retrieval systems, IEEE Trans. Knowl. Data Eng. 16 (2) (2004) 256–270.

[28] X. Han, X. Liu, J. Li, H. Gao, TKAP: efficiently processing top-k query on massive data by adaptive pruning, Knowl. Inf. Syst. 47 (2) (2016) 301–328.

[29] I. F. Ilyas, W. G. Aref, A. K. Elmagarmid, Supporting top-k join queries in relational databases, VLDB J. 13 (3) (2004) 207–221.

[30] X. Han, J. Li, H. Gao, TDEP: efficiently processing top-k dominating query on massive data, Knowl. Inf. Syst. 43 (3) (2015) 689–718.

[31] U. Straccia, N. Madrid, A top-k query answering procedure for fuzzy logic programming, Fuzzy Sets and Systems 205 (2012) 1–29.

[32] M. Theobald, G. Weikum, R. Schenkel, Top-k query evaluation with probabilistic guarantees, in: M. A. Nascimento, et al. (Eds.), Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB 2004), Morgan Kaufmann, 2004, pp. 648–659.

[33] P. Hájek, Mathematics of Fuzzy Logic, Kluwer Academic Publishers, 1998.

[34] C. Beierle, M. Finthammer, G. Kern-Isberner, Relational probabilistic conditionals and their instantiations under maximum entropy semantics for first-order knowledge bases, Entropy 17 (2) (2015) 852–865.

[35] G. Kern-Isberner, T. Lukasiewicz, Combining probabilistic logic programming with the power of maximum entropy, Artif. Intell. 157 (1-2) (2004) 139–202.

[36] M. Schramm, W. Ertel, Reasoning with probabilities and maximum entropy: The system PIT and its application in LEXMED, in: Operations Reseaerch, Proceedings 1999, Springer, 2000.

[37] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1) (1959) 269–271.