WSMO-MX: A Logic Programming Based Hybrid Service Matchmaker

Frank Kaufer Matthias Klusch
German Research Center for Artificial Intelligence
Deduction and Multi-Agent Systems Lab
Stuhlsatzenhausweg 3, Saarbrücken
E-mail: {frank.kaufer,klusch}@dfki.de

Abstract

In this paper, we present an approach to hybrid semantic web service matching based on both logic programming, and syntactic similarity measurement. The implemented matchmaker, called WSMO-MX, applies different matching filters to retrieve WSMO-oriented service descriptions that are semantically relevant to a given query with respect to seven degrees of hybrid matching. These degrees are recursively computed by aggregated valuations of ontology based type matching, logical constraint and relation matching, and syntactic similarity as well.

1 Introduction

The problem of efficiently retrieving relevant services in the envisioned semantic web has been solved so far by only a few approaches for services described in OWL-S [15, 10], and WSML [7, 17]. Though, existing proposals for rule based service mediation in WSMO do not provide a general purpose matchmaking scheme for services in WSML.

This, in particular, motivated us to develop a hybrid semantic matchmaker, called WSMO-MX, that applies different matching filters to retrieve WSMO services that are semantically relevant to a given query including the goal to be satisfied. Both services and goals are described in a Logic Programming (LP) variant of WSML, called WSML-MX, which is based on WSML-Rule. The hybrid matching scheme of WSMO-MX combines the ideas of hybrid semantic matching realized by OWLS-MX [10], the object-oriented structure based matching proposed by Klein & König-Ries [9], and the concept of intentional matching introduced by Keller et. al [6].

The remainder of this paper is structured as follows. In section 2, we introduce WSML-MX, align it with WSML-Rule, and describe the modelling of services in WSML-MX. Section 3 presents the hybrid semantic matching approach of our matchmaker WSMO-MX by means of its different filters of matching, which is then exemplified in section 4. We provide some details of the implementation of WSMO-MX in section 4, and briefly discuss related work and conclude in section 5 and 6, respectively.

2 Service modelling with WSML-MX

The web service modelling language WSML is the formal language for the web service modelling ontology (WSMO). However, WSML still is under development, and there is no full-fledged reasoner with WSML parser available yet. Therefore we developed a formally grounded variant of WSML called WSML-MX directly in F-Logic [8, 1]. WSML-MX is similar expressive as WSML-Rule, which has a different (more verbose) syntax as F-Logic but can be mapped into this.

Central to WSML-MX is the notion of $derivative$ which is an extended version of the object set introduced by Klein and König-Ries [9]. A derivative D_T in WSML-MX encapsulates an ordinary concept T (in this context called type) defined in a given ontology by attaching meta-information merely about the way how T can be matched with any other type. Such information is defined in terms of different meta-relations of the derivative D_T. As type T is defined to be either atomic or a complex type with relations, the derivative D_T can also have a set of relations different from T, though this set is empty by default. The structure of a derivative is shown in figure 1. Per naming convention, the identifier of a derivative D_T of type T is denoted by *T_D*n such as *Person_D42* for type *Person*.

WSML-MX uses the main and clearly motivated elements required for service matching from WSML, that are $goal$, $service$, $capabilities$, $preconditions$, and $postconditions$ but not $effect$ and $assumption$. Please note that the formal semantics of capabilities in WSML is still open. Any service in WSML-MX is modelled as a derivative with a relation called capability and a derivative of type capability as range. Pre- and postcondition are relations of the latter derivative both referring to a so called

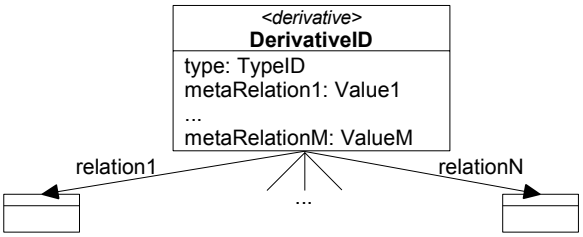


Figure 1. Derivative structure in WSML-MX

state. A state is a set of state parts, which are derivatives each defined as atomic, or as complex by means of relations with derivatives as range. Hence, any service derivative in WSML-MX can be represented as a directed object-oriented graph with derivatives considered as nodes and relations between them as edges, as shown in figure 2.

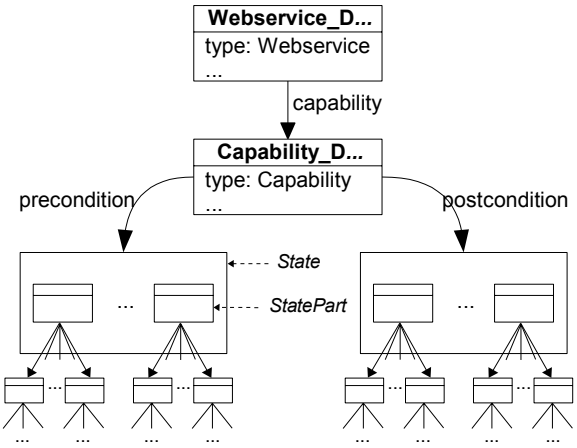


Figure 2. Service derivative in WSML-MX

The language WSML-MX allows for constraints on both relations and derivatives formulated in the full Horn fragment of F-logic. Hence, WSML-MX constraints are as expressive and, in general, only semi-decidable as are WSML-Rule axioms. In WSMO-MX, we use relative query containment for constraint matching (cf. section 3.2.3). However, matching of parts of WSML-MX expressions represented as acyclic object-oriented graphs without constraints is decidable in polynomial time. The emphasis of WSML-MX on these parts of service modelling is motivated not only by clear separation of computationally tractable elements but the fact that it allows the matchmaker for a more detailed explanatory feedback to the user in case the matching of given service and goal derivatives failed.

An example for a service in WSML-MX is shown in figure 3; the service offers tickets for any trip between any two German towns, but if the user departs from Berlin, her destination must be Hamburg.

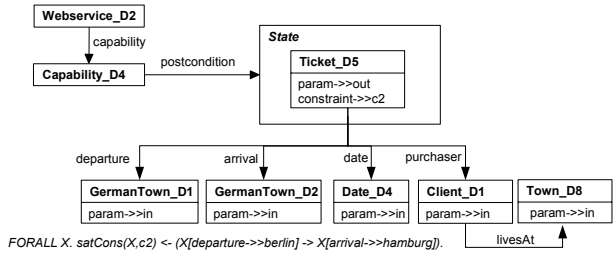


Figure 3. Example service in WSML-MX

3 Hybrid matching of derivatives

3.1 Overview

The result of matching a derivative D_G from a goal description with a derivative D_W from a service description is a vector $v \in R^7$ of aggregated valuations of ontology based type matching, logical constraint matching, relation matching, and syntactic matching. Each real-valued entry in the so called valuation vector $v = (\pi_{\equiv}, \pi_{\sqsubseteq}, \pi_{\sqsupseteq}, \pi_{\sqcap}, \pi_{\sim}, \pi_{\circ}, \pi_{\perp})$ with $\pi_i \in [0,1]$ ($i \in \{\equiv, \sqsubseteq, \sqsupseteq, \sim, \sqcap, \circ, \perp\}$) and $\sum \pi_i = 1$, denotes the extent to which both derivatives D_G and D_W match with respect to the hybrid semantic matching degrees π_i of WSMO-MX.

These degrees are the logical relations $equivalence$, $plug-in$ known from software component retrieval [18] or the similar *rule of consequences* from Hoare logic [4], $inverse-plugin$, $intersection$ and $disjunction$ ($fail$) as degrees of logic based semantic match. The degree of $fuzzy\,similarity$ refers to a non-logic based semantic match such as syntactic similarity, while the degree $neutral$ stands for neither match nor fail, hence declares the tolerance of matching failure. The set-theoretic semantics of the hybrid matching degrees are given in Table 1 based on the relations between the maximum possible instance sets of the derivatives D_G and D_W, denoted by $\mathcal{G}$ and $\mathcal{W}$. Since we use the heuristic relative query containment for the constraint matching, these sets are restricted to instances in the matchmaker knowledge base which satisfy the constraints.

In order to compute the degrees of hybrid semantic matching of given goal and service derivative, WSMO-MX recursively applies different matching filters to their preconditions and postconditions, and returns not only the aggregated matching valuation vector but also annotations of the matching process results as a kind of explanatory feedback to the user. That facilitates a more easy iterative goal refinement by the user in case of insufficient matching results. The individual matching filters and their valuation for the degrees of hybrid semantic matching are described in subsequent sections, and exemplified in section 4.

order	symbol	degree of match	pre	post
1	$\equiv$	equivalence	$\mathcal{G} = \mathcal{W}$	
2	$\sqsubseteq$	plugin	$\mathcal{G} \subseteq \mathcal{W}$	$\mathcal{W} \subseteq \mathcal{G}$
3	$\sqsupseteq$	inverse-plugin	$\mathcal{G} \supseteq \mathcal{W}$	$\mathcal{W} \supseteq \mathcal{G}$
4	$\sqcap$	intersection	$\mathcal{G} \cap \mathcal{W} \neq \emptyset$	
5	$\sim$	fuzzy similarity	$\mathcal{G} \sim \mathcal{W}$	
6	$\circ$	neutral	*by derivative specific definition*	
7	$\perp$	disjunction (fail)	$\mathcal{G} \cap \mathcal{W} = \emptyset$	

Table 1. Degrees of hybrid semantic matching of WSMO service and goal derivatives

3.2 Matching filters

3.2.1 Type matching

The matching of types T_G and T_W of the goal and service derivative D_G and D_W is performed by means of computing the degree of their semantic relation in the matchmaker ontology according to a requested type similarity relation TSR defined as meta-relation values in $D_G[typeSimRel \twoheadrightarrow TSR]$. WSMO-MX offers the following derivative type similarity relations (in F-Logic):

- $equivalent$: $T_W = T_G \vee T_W :: T_G \wedge T_G :: T_W$
- sub: $T_W :: T_G$ (T_W subtype of T_G); $super$: $T_G :: T_W$
- $sibling$: $\exists T_P.T_G :: T_P \wedge T_W :: T_P \wedge \neg(\exists T_X.\exists T_Y.T_X \in \{T_G, T_W\} \wedge T_X :: T_Y \wedge T_Y :: T_P)$; types with one immediate common ancestor (parent).
- $spouse$: $\exists T_C.T_C :: T_G \wedge T_C :: T_W \wedge \neg(\exists T_X.\exists T_Y.T_X \in \{T_G, T_W\} \wedge T_C :: T_Y \wedge T_Y :: T_X)$; types with one immediate common descendant (child)
- $comAnc$ (common ancestor): $\exists T_P.T_G :: T_P \wedge T_W :: T_P$
- $comDes$ (common descendant): $\exists T_C.T_C :: T_G \wedge T_C :: T_W$
- $relative$: exists a path in the undirected ontology graph between T_G and T_W

The maximum distance $TD \in \mathbb{N}\backslash\{0\}$ between types in the matchmaker ontology with respect to which each of the latter three relations gets evaluated to true is specified in the goal derivative in terms of $D_G[typeDistance \twoheadrightarrow TD]$. TD is the path length between both types in the undirected ontology graph; for the type relations $comAnc$ and $comDesc$ it must hold that the addition of the path lengths from both derivatives to their nearest common child/parent type is at most TD. Optionally, the same restriction can be imposed on the type relations sub and $super$ with TD greater or equal the path length from D_G to D_W.

The valuation of the type matching of D_G and D_W for each of the hybrid semantic matching degrees of WSMO-MX is listed in Table 2. If more than one type similarity relation TSR is specified in the goal, the maximum of the valuation vectors is selected as a result.

3.2.2 Relation matching

Given that the D_G and D_W are complex, the hybrid semantic matching must continue recursively with comparing their relations. Let the relation signatures of D_G and D_W be defined as follows: $D_G[R_1 \twoheadrightarrow E_1; ...; R_k \twoheadrightarrow E_k;\ S_1 \twoheadrightarrow F_1; ...;\ S_l \twoheadrightarrow F_l; ...;\ S_m \twoheadrightarrow F_m]$, and $D_W[R_1 \twoheadrightarrow G_1; ...;\ R_k \twoheadrightarrow G_k;\ T_1 \twoheadrightarrow H_1; ...;\ T_n \twoheadrightarrow H_n]$, where $R_1, ..., R_k, S_1, ..., S_m, T_1, ..., T_n$ are unique relation names with $\bigcup_{i\in[1,m]} S_i \cap \bigcup_{j\in[1,n]} T_j = \emptyset$ and derivatives $E_1, ..., E_k, G_1, ...G_k, F_1, ..., F_m, H_1, ..., H_n$ the respective ranges of the relations.

The relations $R_1, ..., R_k$ of the goal derivative D_G for which equally named relations do exist in D_W are valuated for the hybrid degree of matching by recursively matching their ranges with each other. That is, WSMO-MX attempts to match the (goal) derivatives E_τ with the (service) derivatives G_τ for all $\tau \in [1, k]$ and compute the respective valuation vectors.

We assume that for all relations S_μ, $\mu \in [1, l]$ in D_G that cannot be paired with an equally named relation in D_W (under unique name assumption for shared namespaces) there exist one so called *missing strategy* which indicates the matchmaker how to cope with this problem. Such a missing relation strategy is specified in the goal in terms of $D_G[missingStrat@(S_\mu) \twoheadrightarrow MS_\mu]$, with $MS_\mu \in \{assumeEquivalent, assumeFailed, ignore\}$.

The valuations for relations with missing strategies are given in table 3. It lists also the valuations for the relations without missing strategy ($S_l, \ldots S_m$ and $T_1, \ldots, T_n$), which depend on whether they are part of a pre- or postcondition.

The final valuation vector for the recursive relation matching between D_G and D_W is an aggregation of all valuation vectors computed for the missing relations, and those for the relation range derivative matchings. The cor-

type similarity relation	valuation vector	
	val_{pre}	val_{post}
	($\pi_{\equiv}$,$\pi_{\sqsubseteq}$,$\pi_{\sqsupseteq}$,$\pi_{\sqcap}$,$\pi_{\sim}$,$\pi_{\circ}$,$\pi_{\perp}$)	($\pi_{\equiv}$,$\pi_{\sqsubseteq}$,$\pi_{\sqsupseteq}$,$\pi_{\sqcap}$,$\pi_{\sim}$,$\pi_{\circ}$,$\pi_{\perp}$)
$equivalent$	(1 , 0 , 0 , 0 , 0 , 0 , 0)	(1 , 0 , 0 , 0 , 0 , 0 , 0)
sub	(0 , 0 , 1 , 0 , 0 , 0 , 0)	(0 , 1 , 0 , 0 , 0 , 0 , 0)
$super$	(0 , 1 , 0 , 0 , 0 , 0 , 0)	(0 , 0 , 1 , 0 , 0 , 0 , 0)
$sibling$	(0 , 0 , 0 , 1 , 0 , 0 , 0)	(0 , 0 , 0 , 1 , 0 , 0 , 0)
$comAnc$	(0 , 0 , 0 , 1 , 0 , 0 , 0)	(0 , 0 , 0 , 1 , 0 , 0 , 0)
$spouse$	(0 , 0 , 0 , 0 , 1 , 0 , 0)	(0 , 0 , 0 , 0 , 1 , 0 , 0)
$comDes$	(0 , 0 , 0 , 0 , 1 , 0 , 0)	(0 , 0 , 0 , 0 , 1 , 0 , 0)
$relative$	(0 , 0 , 0 , 0 , 1 , 0 , 0)	(0 , 0 , 0 , 0 , 1 , 0 , 0)

Table 2. Valuation of type matching for hybrid matching degrees

missing strategy	valuation vector	
	$val_{pre,webservice}/val_{post,goal}$	$val_{post,webservice}/val_{pre,goal}$
	($\pi_{\equiv}$,$\pi_{\sqsubseteq}$,$\pi_{\sqsupseteq}$,$\pi_{\sqcap}$,$\pi_{\sim}$,$\pi_{\circ}$,$\pi_{\perp}$)	($\pi_{\equiv}$,$\pi_{\sqsubseteq}$,$\pi_{\sqsupseteq}$,$\pi_{\sqcap}$,$\pi_{\sim}$,$\pi_{\circ}$,$\pi_{\perp}$)
$assumeEquivalent$	(1 , 0 , 0 , 0 , 0 , 0 , 0)	(1 , 0 , 0 , 0 , 0 , 0 , 0)
none	(0 , 1 , 0 , 0 , 0 , 0 , 0)	(0 , 0 , 1 , 0 , 0 , 0 , 0)
$ignore$	(0 , 0 , 0 , 0 , 0 , 1 , 0)	(0 , 0 , 0 , 0 , 0 , 1 , 0)
$assumeFailed$	(0 , 0 , 0 , 0 , 0 , 0 , 1)	(0 , 0 , 0 , 0 , 0 , 0 , 1)

Table 3. Valuation of relation matching with missing strategies for hybrid matching degrees

responding relation matching algorithm is outlined in the subsequent section (cf. algorithm 5).

3.2.3 Constraint matching

Let D a derivative, C a F-Logic rule body and X_D a free variable in C, then we call c a *constraint* of D, if $D[constraint{\rightarrow\!\!\!\!\rightarrow}c]$. and $\forall X_D.satCons(X_D, c) \leftarrow C.$ holds. Variable X_D is bound with potential instances of D, and $satCons$ verifies whether such an instance satisfies c. A derivative can have zero or many constraints including a special constraint for nominals; the respective meta-relation *oneOf* denoted as $D[oneOf{\rightarrow\!\!\!\!\rightarrow}\{i_1, \ldots, i_m\}]$ means that an instance of D has to be one of $i_1, \ldots, i_m$.

In WSMO-MX, the matching of logical constraints of goal and service derivatives is performed by means of so called relative query containment. That is, any clause A is relatively contained in clause B, or B relatively implies A, with respect to a given knowledge base $\mathcal{KB}$, denoted by $A \sqsubseteq_{\mathcal{KB}} B$, if the answer set $Q_{\mathcal{KB}}(A)$ of querying $\mathcal{KB}$ with A, is a subset of $Q_{\mathcal{KB}}(B)$. Under the open world assumption, $\mathcal{KB}$ does not contain all possible instances of a query (universal closure), hence relative query containment can only be considered as an approximation of logical implication (query containment) which is, in general, undecidable for first-order languages such as F-Logic [2]. An alternative would be to approximate logical implication by means of clause theta-subsumption [13] which is, in general, NP-complete decidable [3]. Since fast deterministic algorithms for partial testing of theta-subsumption are also known [14], the correct but incomplete theta-subsumption relation is used as a consequence relation in many ILP systems [12], and the matchmaker LARKS [16].

However, for pragmatic reasons of implementation, WSMO-MX uses relative query containment for matching constraints over the instances stored in the matchmaker ontology. For each derivative D of type T, WSMO-MX determines a set of potential instances against which its constraints are evaluated as queries. This set comprises all instances of the concept T and instances of derivatives of type T:

$$\forall D, X_D.\, potentialInstance(D, X_D) \leftarrow \exists T.\, D[type{\rightarrow\!\!\!\!\rightarrow}T] \wedge (X_D : T \vee (\exists D_T.\, D_T[type{\rightarrow\!\!\!\!\rightarrow}T] \wedge X_D : D_T)).$$

The constraint matching filter then returns only those instances of this set which satisfy all constraints of D:

$$\forall X_D, D.\, satAllCons(X_D, D) \leftarrow potentialInstance(D, X_D) \wedge (\forall C.\, D[constraint{\rightarrow\!\!\!\!\rightarrow}C] \rightarrow satCons(X_D, C)) \wedge ((\exists X.\, D[oneOf{\rightarrow\!\!\!\!\rightarrow}X]) \rightarrow D[oneOf{\rightarrow\!\!\!\!\rightarrow}X_D]).$$

The valuation of constraint matching is determined by the type of the set relation ρ, which is defined as $\mathcal{I}_{\mathcal{KB}}(D_G)\ \rho\ \mathcal{I}_{\mathcal{KB}}(D_W)$ over the set $\mathcal{I}_{\mathcal{KB}}(D) := \{X_D | satAllCons(X_D, D)\}$ of matching instances of derivative D with respect to the given knowledge base $\mathcal{KB}$ of the matchmaker (cf. table 4).

set relation	valuation vector	
	val_{pre}	val_{post}
$\mathcal{I}_{\mathcal{KB}}(D_G)\ \rho\ \mathcal{I}_{\mathcal{KB}}(D_W)$	($\pi_{\equiv}$,$\pi_{\sqsubseteq}$,$\pi_{\sqsupseteq}$,$\pi_{\sqcap}$,$\pi_{\sim}$,$\pi_{\circ}$,$\pi_{\perp}$)	($\pi_{\equiv}$,$\pi_{\sqsubseteq}$,$\pi_{\sqsupseteq}$,$\pi_{\sqcap}$,$\pi_{\sim}$,$\pi_{\circ}$,$\pi_{\perp}$)
$\mathcal{I}_{\mathcal{KB}}(D_G) = \mathcal{I}_{\mathcal{KB}}(D_W)$	(1 , 0 , 0 , 0 , 0 , 0 , 0)	(1 , 0 , 0 , 0 , 0 , 0 , 0)
$\mathcal{I}_{\mathcal{KB}}(D_G) \supseteq \mathcal{I}_{\mathcal{KB}}(D_W)$	(0 , 0 , 1 , 0 , 0 , 0 , 0)	(0 , 1 , 0 , 0 , 0 , 0 , 0)
$\mathcal{I}_{\mathcal{KB}}(D_G) \subseteq \mathcal{I}_{\mathcal{KB}}(D_W)$	(0 , 1 , 0 , 0 , 0 , 0 , 0)	(0 , 0 , 1 , 0 , 0 , 0 , 0)
$\mathcal{I}_{\mathcal{KB}}(D_G) \cap \mathcal{I}_{\mathcal{KB}}(D_W) \neq \emptyset$	(0 , 0 , 0 , 1 , 0 , 0 , 0)	(0 , 0 , 0 , 1 , 0 , 0 , 0)
$\mathcal{I}_{\mathcal{KB}}(D_G) \cap \mathcal{I}_{\mathcal{KB}}(D_W) = \emptyset$	(0 , 0 , 0 , 0 , 0 , 0 , 1)	(0 , 0 , 0 , 0 , 0 , 0 , 1)

Table 4. Valuation of constraint matching for hybrid matching degrees

3.2.4 Syntactic matching

The filter of WSMO-MX for syntactic matching of goal and service derivatives, D_G and D_W, is intended to complement those for semantic matching as described above. For this purpose, it transforms the description of each derivative into a weighted keyword vector as known from information retrieval, and applies one of the selected syntactic similarity metrics cosine, extended Jaccard, loss-of-information (LOI), and weighted LOI [10], depending on the user preferences specified as instances of the following meta-relations of goal derivatives D_G.

- $D_G[synSimUsage \rightarrow\!\!\!\!\rightarrow U]$ with $U \in \{alternative, compensative, complementary\}$ specifies whether syntactic matching shall be performed either as an exclusive alternative to semantic matching, or only in case of semantic matching failure, or in any case.
- $D_G[synSimScope \rightarrow\!\!\!\!\rightarrow S]$ with $S \in \{scpType, scpRelation, scpDescription\}$ denotes whether only the types, or the relations, or the whole text of the description of the derivatives are used for syntactic matching. In case of *scpType*, all type names (no relation names) of the derivative are recursively unfolded in the matchmaker ontology and the resulting set of primitive components used to compute a weighted keyword vector, whereas for *scpRelation* only the relation names of the derivative are used for this purpose. Any combination of scopes is allowed.
- $D_G[synSimMetric \rightarrow\!\!\!\!\rightarrow M]$ with $M \in \{cosine, loi, loiWeighted, jaccard\}$ specifies which IR similarity metric to use. For details of computation, we refer to [10].
- $D_G[synSimMinDegree \rightarrow\!\!\!\!\rightarrow \alpha]$ with $\alpha \in [0, 1]$ specifies the minimum degree of syntactic similarity required (threshold).

The valuation of syntactic matching is considered only with respect to the degree of fuzzy similarity $\pi_{\sim}$ and set to 0, if the computed syntactic similarity value does not exceed α, and to 1 otherwise.

3.2.5 Parameter matching

A derivative can be tagged to be an input and/or output parameter by the meta-relation $param$. The parameter matching filter checks whether goal and service derivative are differently tagged and returns no valuation vector but an annotation indicating the deviations. This allows the service requester to understand the interface of the service and if needed to adjust the interface as it was expected and denoted by the parameters tags in the goal description.

3.2.6 Intentional matching

Optionally, WSMO-MX does perform a kind of intentional matching of goal and service derivatives. For this purpose, we adopt the approach proposed by Keller et al. [6]. In particular, the semantics of their notions of $\exists$-intention and $\forall$-intention correspond with the evaluation of our meta-relation $existentialIntention$ to $true$ and $false$, respectively. The valuation vector of hybrid semantic matching can be "intentionally recomputed" by its multiplication with the transformation matrix that corresponds to the requested combination of intended provision of relevant instances as it is declared for the goal and the service derivative by the requester and provider, respectively.

The case in which $\forall$-intentions are declared for both derivatives, D_G and D_W, is equal to not using intentions at all, hence can simply be ignored by WSMO-MX. As a consequence, there remain three cases for each pre- and post-condition matching. These are computed by means of six intentional matching matrices (to be multiplied with the valuation vector) of which we show only those for the postcondition matching cases [1]: (1) $I_{post,\exists G,\forall W}$: only D_G has an $\exists$-intention, (2) $I_{post,\forall G,\exists W}$: only D_W has an $\exists$-intention, (3) $I_{post,\exists G,\exists W}$: both derivatives have $\exists$-intentions. The matrices are defined as follows.

[1] For the cases of precondition matching the lines and columns for $\pi_{\sqsubseteq}$ and $\pi_{\sqsupseteq}$ in the matrices have to be inverted.

$$I_{post,\exists G,\forall W} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$I_{post,\forall G,\exists W} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & \frac{1}{5} & 0 & 0 & \frac{1}{5} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$I_{post,\exists G,\exists W} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} \\ \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Due to space restrictions, we refer the interested reader for more details to [5].

3.3 WSMO-MX matching algorithm

The request for a semantically relevant service is specified by the user as a goal derivative in WSML-MX, together with a matching configuration $Conf$. The configuration contains default values for minimum syntactic similarity degree, weights for the aggregation of different matching filter results, and the minimum valuation of each degree of hybrid matching returned by the matchmaker. WSMO-MX takes the precondition state and the postcondition state of each advertised service from its local knowledge base (cf. algorithm 1), and then matches them pairwise with the states of the given goal (cf. algorithm 2). In case of no preconditions, the result of their matching is set to equivalence by default.

The state of the goal is matched with that of the service by matching their state part derivatives (cf. algorithm 3) and then recursively by the pairwise matching of relation range derivatives of equally named relations (cf. algorithm 5). Subsequently, WSMO-MX computes the *maximum* weighted bipartite graph match, where nodes of the graph correspond to the goal and service state parts and the computed valuation vectors act as weights of edges existing between matched state parts.

At each step in the recursion, the parameter matching filter is applied first, since its result, an annotation record, is not valuated for any of the hybrid matching degrees. Then each of the semantic matching filters (type, constraint, and relation matching) is applied. Syntactic matching is performed in case one of these filters fails (compensative), or complementary in any case, if not specified differently. The user can also ask for just a first coarse-grained filtering by means of exclusively syntactic matching without any semantic matching.

Finally, all valuation vectors computed during recursive matching of goal and service derivatives are aggregated into one single valuation vector. For aggregation, each individual valuation vector is weighted for the respective matching filter as specified in the configuration ($Conf$) for the given goal; the weighting is assumed to be equal by default. This aggregated valuation of hybrid matching degrees is then recomputed with respect to the intentions of the considered derivatives (cf. in section 3.2.6).

The overall result of the matching process is a ranked list of services with their hybrid matching valuation vector, and annotations. Services are ranked with respect to the maximum value of hybrid semantic matching degrees in descending order (cf. table 1), starting with $\pi_{\equiv}$.

Algorithm 1 *WSMO-MX matching of query (goal G, configuration $Conf$) with registered services in WSML-MX: matchGoal*

```
1:  function MATCHGOAL(G, Conf)
        WS := GETREGISTEREDWEBSERVICES( )
2:      S_{G,pre} := GETPRECONDITION(G)
3:      S_{G,post} := GETPOSTCONDITION(G)
4:      Conf_pre := Conf + (modus : pre)
5:      Conf_post := Conf + (modus : post)
6:      W_matched := empty set
7:      for all W ∈ WS do
8:          S_{W,pre} := GETPRECONDITION(W)
9:          S_{W,post} := GETPOSTCONDITION(W)
10:         (Val_{W,pre}, Ann_{W,pre}) :=
                MATCHSTATES(S_{G,pre}, S_{W,pre}, Conf_pre)
11:         (Val_{W,post}, Ann_{W,post}) :=
                MATCHSTATES(S_{G,post}, S_{W,post}, Conf_post)
12:         W_matched += (W, Val_{W,pre}, Val_{W,post},
                          Ann_{W,pre}, Ann_{W,post})
13:     end for
14:     return W_matched
15: end function
```

3.4 Implementation

WSMO-MX has been fully implemented in Java 5 and F-Logic using the F-Logic reasoner OntoBroker[2]. Its main components are a matching engine which is interfaced with an ontology manager communicating with the reasoner. Type and constraint matching is done directly within the OntoBroker, whereas the WSMO-MX matching engine

[2]developed by Ontoprise, `http://www.ontoprise.de`

Algorithm 2 *matchStates*

function MATCHSTATES($\mathcal{S}_G, \mathcal{S}_W, Conf$)
  ▷ build bipartite weighted graph from
  ▷ matching state parts of goal and webservice
  $Graph$:= empty graph
  for all $StatePart_G \in \mathcal{S}_G$ **do**
    for all $StatePart_W \in \mathcal{S}_W$ **do**
      (Val_W, Ann_W) := MATCHDERIVATIVES $(StatePart_G, StatePart_W, Conf)$
      if ¬ ISFAIL(Val_W) **then**
        $Graph$ += $edge(StatePart_G, StatePart_W, Val_W, Ann_W)$
      end if
    end for
  end for

  ▷ find maximum weighted graph matching
  M := GETGRAPHMATCHING($Graph$)
  (Val, Ann) := GETVALANN($M, Conf$)

  ▷ valuate not matched state parts
  $\mathcal{S}_{G-M}$:= NOTMATCHEDSTATEPARTS($\mathcal{S}_G, M$)
  $\mathcal{S}_{W-M}$:= NOTMATCHEDSTATEPARTS($\mathcal{S}_W, M$)
  for all $StatePart_G \in \mathcal{S}_{G-M}$ **do**
    Val += VALSTATEPART($goal, Conf$)
    Ann += $(G, W, state, (StatePart_G, notMatched, goal))$
  end for
  for all $StatePart_W \in \mathcal{S}_{W-M}$ **do**
    Val += VALSTATEPART($webservice, Conf$)
    Ann += $(G, W, state, (StatePart_W, notMatched, webservice))$
  end for

  ▷ normalize cumulated valuation
  Val /= $|M| + \mathcal{S}_{G-M} + \mathcal{S}_{W-M}$
  return (Val, Ann)
end function

Algorithm 3 *matchDerivatives*

function MATCHDERIVATIVES(D_G, D_W, $Conf$)

  if $D_G = D_W$ **then return** $((1,0,0,0,0,0,0),())$
  end if

  Ann_{Params} := MATCHPARAMS(D_G, D_W)
  $synMatchUsage$:= GETSYNMATCHINGUSAGE($D_G, Conf$)

  if $synMatchUsage$ = $alternative$ **then**
    (Val_{Syn}, Ann_{Syn}) := MATCHSYNTACTIC($D_G, D_W, Conf$)
    if ¬ISFAIL(Val_{Syn}) **then**
      Ann += $Ann_{Params} + Ann_{Syn}$
      return (Val_{Syn}, Ann)
    end if
  end if

  (Val_{Sem}, Ann_{Sem}) := MATCHSEMANTIC($D_G, D_W, Conf$)

  if ISFAIL(Val_{Sem}) **then**
    if $synMatchUsage$ = $compensative$ **then**
      (Val_{Syn}, Ann_{Syn}) := MATCHSYNTACTIC($D_G, D_W, Conf$)
      if ¬ISFAIL(Val_{Syn}) **then**
        Ann += $Ann_{Params} + Ann_{Syn}$
        return (Val_{Syn}, Ann)
      end if
    end if
  else if $synMatchUsage$ = $complementary$ **then**
    $(Val_{Syn}, AnnSyn)$:= MATCHSYNTACTIC($D_G, D_W, Conf$)
    if ¬ISFAIL($ValSyn$) **then**
      Ann += $Ann_{Params} + Ann_{Syn} + Ann_{Sem}$
      Val := AGGREGATEVAL($Val_{Sem}, Val_{Syn}, Conf$)
      return (Val, Ann)
    end if
  else
    Ann += $Ann_{Params} + Ann_{Sem}$
    Val := AGGREGATEVAL($Val_{Sem}, null, Conf$)
    return (Val, Ann)
  end if
end function

Algorithm 4 *matchSemantic*

function MATCHSEMANTIC(D_G, D_W, $Conf$)
  (Val_{Type}, Ann_{Type}) := MATCHTYPES(D_G, D_W, $Conf$)
  (Val_{Cons}, Ann_{Cons}) := MATCHCONSTRAINTS(D_G, D_W, $Conf$)
  (Val_{Rel}, Ann_{Rel}) := MATCHRELATIONS(D_G, D_W, $Conf$)

  Val := (Val_{Type}, Val_{Cons}, Val_{Rel})
  Ann := $Ann_{Type} + Ann_{Cons} + Ann_{Rel}$

  return (Val, Ann)
end function

Algorithm 5 *matchRelations*

function MATCHRELATIONS(D_G, D_W, $Conf$)
  ▷ $Rels_G$ - relations defined only for D_G
  ▷ $Rels_W$ - relations defined only for D_W
  ▷ $Rels_{G,W}$ - relations defined for both
  ($Rels_G$, $Rels_W$, $Rels_{G,W}$) := GETRELATIONS(D_G, D_W)

  ▷ for all relations defined in D_G *and* D_W
  ▷ match the derivatives in their range
  for all $R \in Rels_{G,W}$ **do**
    $Range_G$:= GETRELRANGE(D_G)
    $Range_W$:= GETRELRANGE(D_W)
    (Val_{Range}, Ann_{Range}) := MATCHDERIVATIVES($Range_G$, $Range_W$)
    Val += Val_{Range}
    Ann += Ann_{Range}
  end for
  ▷ valuate relations defined only for D_G
  for all $R \in Rels_G$ **do**
    MS_R := GETMISSINGSTRATEGY(D_G, R, $Conf$)
    Val += VALUATEMISSREL($webservice$, MS_R, $Conf$)
    Ann += (D_G, D_W, rel, (R, $missing$, $webservice$, MS_R))
  end for
  ▷ valuate relations defined only for D_W
  for all $R \in Rels_W$ **do**
    Val += VALUATEMISSREL($goal$, $null$, $Conf$)
    Ann += (D_G, D_W, rel, (R, $missing$, $goal$))
  end for

  Val /= $|Rels_G| + |Rels_W| + |Rels_{G,W}|$
end function

takes the results and does the rest, that is relation matching, syntactic matching, aggregation of valuation vectors, state matching including the computation of maximum weighted bipartite graph matching. The OntoBroker loads the matchmaker ontology from a given set of F-Logic files that contain the types, derivatives (including goals and services), instances, and constraints, as well as the rules for type and constraint matching, unfolding and some auxiliary tasks. In an upcoming version of WSMO-MX, the goals will be passed by the matching engine to the ontology manager only at the time of the respective request to the matchmaker.

4 Example

Goal, service, ontology. Suppose the user defines a goal derivative *Ticket_D4* as shown in figure 4. That is, she is looking for any ticket for a trip between two arbitrary towns, but if it starts in Berlin, then it must not end in Bremen. Please note, that the user may specify matching relaxations for any object of the goal as exemplified, but also different weights for the matching filters to be applied. In this example, we assume the filters to be equally weighted.

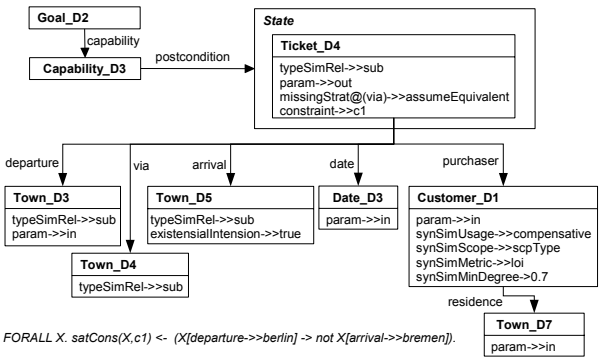
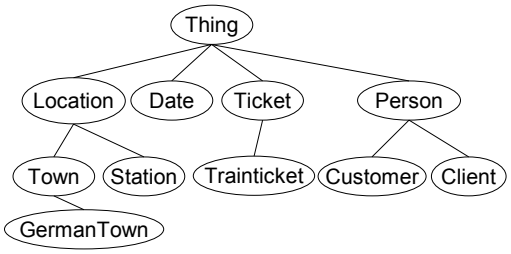


Figure 4. Example goal in WSML-MX

The part of the type hierarchy in the matchmaker ontology and all instances used in this example are shown in figure 5.

For reasons of efficiency and data privacy, mediation between service providers and requesters by means of an autonomous matchmaker is not appropriate for constraint matching over different instance bases. Alternatively, an autonomous WSMO-MX matchmaker could perform constraint matching without instance sets by polynomial means of theta-subsumption reasoning for restricted set of Horn clauses like in LARKS [16]. This is part of our future work on WSMO-MX.

In this example, the service derivative *Ticket_D5* given in section 2 will be matched against the goal derivative *Ticket_D4*. Please note, that the service offers tickets for any trip between any two German towns, but if the user




```
t1:Ticket_D4[departure->>berlin;
             arrival->>leipzig; ...].
t2:Ticket_D4[departure->>berlin;
             arrival->>kiel; ...].
t3:Ticket_D5[departure->>hamburg;
             arrival->>bremen; ...].
t4:Ticket_D5[departure->>hamburg;
             arrival->>hannover; ...].
t5:Ticket_D5[departure->>berlin;
             arrival->>hamburg; ...].
t6:Ticket_D6[departure->>berlin;
             arrival->>bremen; ...].
```

Figure 5. Example ontology (type hierarchy and instances)

departs from Berlin, her destination must be Hamburg.

Matching. Since the capabilities of both goal and service derivatives do not include any precondition, the hybrid semantic matching of them is restricted to the matching of their postcondition states as follows.

1. **match types**: the types of Ticket_D4 and Ticket_D5 are equal. Hence the valuation is $v_1 = (1, 0, 0, 0, 0, 0, 0, 0)$.

2. **match parameters**: both are output parameters, no annotation necessary

3. **match relations**

   (a) *departure*: the types of Town_D3 and GermanTown_D1 are not equivalent, but Town_D3 allows subtypes. Since GermanTown is a subconcept of Town, the valuation is $v_2 = (0, 1, 0, 0, 0, 0, 0, 0)$.

   (b) *via*: this relation is not defined for Ticket_D3, but the missingStrategy for this relation is *assumeEquivalent* yielding a valuation $v_3 = (1, 0, 0, 0, 0, 0, 0, 0)$.

   (c) *arrival*: analogous to *departure* types of the ranges of *arrival* are subtypes and yield the valuation $v_4 = (0, 1, 0, 0, 0, 0, 0, 0)$.

   (d) *date*: is equal in goal and service, hence valuated as $v_5 = (1, 0, 0, 0, 0, 0, 0, 0)$

   (e) *purchaser*: type matching fails for Customer_D1 and Client_D1, but compensative syntactic matching is allowed using loss of information (LOI) metric. For the unfolding only the types of the derivatives should be used (*scpType*), yielding the term vectors $(Customer : 1, Town : 1, Person : 1, Location : 1, Town : 1)$ and $(Client : 1, Town : 1, Person : 1, Location : 1, Town : 1)$ for Customer_D1 and Client_D1, respectively. The similarity degree is 0.75, and therefore greater than the declared minimum of 0.7. The resulting valuation vector is $v_6 = (0, 0, 0, 0, 0, 0, 1, 0)$.

   The aggregated relation valuation is $v_7 = \frac{v_2+...+v_6}{5} = (0.4, 0.4, 0, 0, 0, 0.2, 0)$

4. **match constraints**: Ticket_D4 has the constraint c1. This is satisfied by the instances $t1, \ldots, t5$. The constraint c2, which is imposed on Ticket_D5 is satisfied by the instances $t3, \ldots, t5$. That means the instances for Ticket_D5 are a subset of those of Ticket_D4 and hence the valuation is $v_8 = (0, 1, 0, 0, 0, 0, 0, 0)$

Finally, the aggregated valuation for the derivative matching of Ticket_D4 and Ticket_D5 is

$$v_9 = \frac{v_1+v_7+v_8}{3} = (\frac{7}{15}, \frac{7}{15}, 0, 0, 0, \frac{1}{15}, 0).$$

5 Related work

To the best of our knowledge, WSMO-MX is the first implemented full-fledged matchmaker for WSMO-oriented services. It borrows the approach to recursive object-oriented structure matching from [9], the notion of intentional matching from [6], and the hybrid semantic matching from [10]. The mediator based discovery approaches presented in [7, 17] do not allow for a general goal-service matching, but require problem specific mapping, or construction rules. Besides, like in [15], they define their notions of match on the assumption that an advertisement postcondition has to subsume the goal's postcondition for a full match, which is diametrically opposed to our approach and to the original idea of how to match program capabilities initially proposed in [4, 18].

Other relevant approaches to automated selection of semantic web services include those for retrieving relevant OWL-S services [11, 15]. Most of them rely on DL based subsumption reasoning. However, OWL still lacks the support of rules and subsumption reasoning in the underlying description logic $\mathcal{SHOIN}(\mathcal{D})$ is NEXPTIME. Besides, unlike WSMO, there is no way in OWL-S to link I/O parameters in the signature with preconditions and effects as shared variables. Thus, most OWL-S matchmakers perform

signature matching only. OWLS-MX [10] complements the logic based semantic matching of OWL-S service signatures with syntactic matching, which is also rudimentary performed in LARKS [16]. For WSMO-MX, we did improve on this idea of OWLS-MX by allowing for a more fine-grained parametrisation, and integrated interleaving of syntactic and semantic matching.

6 Conclusions

In this paper we presented the general purpose matchmaker WSMO-MX for services described in WSML-MX which is a LP based variant of WSML-Rule that facilitates matching of pre- and postconditions of object-oriented descriptions of goals and services. WSMO-MX applies different matching filters to retrieve WSMO services that are semantically relevant to a given goal with respect to seven degrees of hybrid matching. Each of these degrees are recursively computed by aggregated valuations of ontology based type matching, logical constraint and relation matching, and syntactic similarity of goal and service derivatives. It integrates signature matching with state matching, and returns not only the final aggregated valuation vector for the hybrid matching degrees but an annotation of the matching results for interactive goal refinement by the user. Currently, relation cardinalities are not considered by WSMO-MX but will be integrated as soon as they become standardised [3] and supported by an F-Logic reasoner. Though the matchmaker has been fully implemented, the evaluation of its performance is ongoing work with generating the required WSMO service retrieval test collection first. Like with OWLS-MX, we intend to make WSMO-MX (without OntoBroker[4]) available to the semantic community under GPL-like license at the semwebcentral.org portal.

References

[1] J. Angele and G. Lausen. Ontologies in f-logic. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, pages 29–50. Springer, 2004.

[2] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, September 2001.

[3] G. Gottlob and A. Leitsch. On the efficiency of subsumption algorithms. *Journal of the ACM (JACM)*, 32(2):280 – 295, April 1985.

[4] C. Hoare. An axiomatic basis for computer programming. *Communications of the ACM (CACM)*, 12(10):576–580, 10 1969.

[5] F. Kaufer. *WSMO-MX: A Logic programming based hybrid semantic web service matchmaker*. Computer Science Dept., University of the Saarland, Saarbruecken, Germany, 2006.

[6] U. Keller, R. Lara, H. Lausen, A. Polleres, and D. Fensel. Automatic location of services. In *Proceedings of the 2nd European Semantic Web Symposium (ESWS2005)*, Heraklion, Crete, June 2005.

[7] M. Kifer, R. Lara, A. Polleres, C. Zhao, U. Keller, H. Lausen, and D. Fensel. A logical framework for web service discovery. In *Proceedings of the ISWC 2004 Workshop on Semantic Web Services: Preparing to Meet the World of Business Applications*, volume 119, Hiroshima, Japan, November 2004. CEUR Workshop Proceedings.

[8] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42, 1995.

[9] M. Klein and B. König-Ries. Coupled signature and specification matching for automatic service binding. In *Proceedings of European Conference on Web Services (ECOWS 2004)*, LNCS 3250, page 183, Erfurt, Germany, September 2004. Springer.

[10] M. Klusch, B. Fries, M. Khalid, and K. Sycara. Owls-mx: Hybrid owl-s service matchmaking. In *Proceedings of 1st Intl. AAAI Fall Symposium on Agents and the Semantic Web*, Arlington VA, USA, November 2005.

[11] L. Li and I. Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the Twelfth International Conference on World Wide Web*, pages 331–339. ACM Press, 2003.

[12] S. Muggleton and L. D. Raedt. Inductive logic programming: Theory and applications. *Logic Programming*, 19(20):629–679, 1994.

[13] J. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1), 1965.

[14] T. Scheffer, R. Herbrich, and F. Wysotzki. Efficient algorithms for theta-subsumption. *JAIR*, 1997.

[15] K. Sycara, M. Paolucci, A. Ankolekar, and N. Srinivasan. Automated discovery, interaction and composition of semantic web services. *Journal of Web Semantics*, 1(1):28, 2003.

[16] K. Sycara, S. Widoff, M. Klusch, and J. Lu. Larks: Dynamic matchmaking among heterogeneous software agents in cyberspace. *Autonomous Agents and Multi-Agent Systems*, 5:173–2003, June 2002.

[17] E. D. Valle and D. Cerizza. Cocoon glue: a prototype of wsmo discovery engine for the healthcare field. In *Proceedings of the WIW 2005 Workshop on WSMO Implementations*, volume 134, Innsbruck, Austria, June 2005. CEUR Workshop Proceedings.

[18] A. M. Zaremski and J. M. Wing. Specification matching of software components. In *3rd ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 10 1995.

[3] For more information see `http://forum.projects.semwebcentral.org/`

[4] research licences can be obtained from Ontoprise