# Applying Description Logics with Concrete Domains to Solve the Problem of Semantic Web Services Discovery and Composition

Olha Zakharova[1]

[1] *Institute of Software Systems of the National Academy of Sciences of Ukraine, Glushkova ave. 40, Kiyv, 03187, Ukraine*

**Abstract**

This research is one of the branches of a general scientific and applied problem, namely, the problem of using and developing ontological approaches with descriptive logics (DL) apparatus to obtain the effective automated solution of complex business task based on service-oriented architecture. The base idea of the proposed method is the maximal ontologization, that is, all aspects of the task are determined by means of DL ontologies. The proposed ontology system is based on five basic types of ontologies, namely: (1) an ontology of general entities, (2) an ontology of the application domain, (3) the top-level ontology of web services, (4) the ontologies of the web services tasks and (5) the ontologies of specific web services. All listed ontologies are related. In previous studies, prototypes of the ontologies of types 1-3 and 5 have already been proposed based on the basic descriptive logic $\mathcal{ALC}$. It is the effective tool for knowledge representation and ensures the execution of standard deductions. But, DL-specifications of the web services tasks require the modelling «abstract» objects (parameter, condition, service etc.) and its «specific» characteristics, in particular, time and space characteristics. These specific values can't be represented in the model as elements of the area $\triangle$. This is because when moving one interpretation to another, these "specific" elements and the links between them will change, while there is a requirement that they should be unchanged. When using descriptive logics with concrete domains, it is allocated a separate area with a fixed set of predicates. So, the interpretation "in the new sense" consists of the interpretation "in the old sense" («abstract») extended by the fixed «concrete» interpretation (with the set of predicates). Obviously, this also requires using the special roles to link these "abstract" and "concrete" elements of constructors to build concepts based on such roles and "concrete" predicates. So, the focus of this study is the applying descriptive logics with concrete domains to the formalization of basic semantic web services tasks. In this article, the ontologies of the web services tasks are defined as an extension of the general top-level web service ontology with special complex concepts. Their elements are pairs of "web service-goal (request)"/"web service-web service", which are compared according to the task being solved. Also, on their basis, special complex concepts are determined, which specify a set of found similarities between existing services and a goal in the discovery task, or "predecessor-follower" type responsibilities of web services to build a composite service. The proposed TBox statements are constructed using descriptive logic with concrete domains $\mathcal{A}(\mathcal{D})$. For this purpose, a concrete domain $\mathcal{D}_{match}$={$\boldsymbol{\mathcal{S}}$, $\boldsymbol{\Phi}$} is defined as a pair of sets: (1) $\boldsymbol{\mathcal{S}} = \mathcal{S}_P \sqcup \mathcal{S}_C$ is the union of both sets: $\mathcal{S}_P$ – the set of the possible subsets of instances of the concept Parameter and possible subsets of the set of individuals of the Context concept, and (2) $\boldsymbol{\Phi}$ is the set of the predicates defined on $\boldsymbol{\mathcal{S}}$. For now, $\boldsymbol{\Phi}$ was considered with unary (P1) and binary (P2) predicate symbols: P1 = {$\top$}, P2= {$\subseteq$, $\supseteq$, $\cong$, $\not\cong$, $\not\subseteq$, $\not\supseteq$}. It should be noted that the set theory operators was used. This is due to the features of the chosen domain: elements of the domain are the sets of the instances. Such definition of the concrete domain provides possibility of using only functional roles to construct the DL statements of complex concepts DiscoveryServices and CompositionObject for determining corresponding web services tasks.

**Keywords**

Semantic web services problems, a discovery task, a composition task, a discovery goal, composition goal, description logics with concrete domains, concrete domain, web services matching, a «predecessor-follower» chain, a functional model of web services, a process model of web services, concrete domain solvability, web services contexts matching, ontological approaches for solving problems, system of the ontologies, top level general web service ontology, a domain ontology, an ontology of the web services tasks, named entities taxonomy, ontological structure of general entities.

13th International Scientific and Practical Conference from Programming UkrPROGP'2022, October 11-12, 2022, Kyiv, Ukraine
EMAIL: ozakharova68@gmail.com (A. 1)
ORCID: 0000-0002-9579-2973 (A. 1)

## Introduction

Ontologies increasingly cover all areas related to the data and knowledge representation and, also, to the definition of objects, processes, states, tasks, and algorithms. This is due to their effectiveness as a tool for the formal description of information elements and to their reasoners that allow deriving new knowledge and making certain conclusions from existing assertions and rules.

Today, ontologies are the widespread tool for formally describing application areas or domains. As for semantic web services, there are many studies which are related to applying ontological formalisms, in particular, descriptive logics, to solve the problems of web services' presentation and semantization. The use of an integrated domain ontology provides an unambiguous understanding of the semantics of significant elements of the application area. This ensures the accuracy of the obtained results. And the specification of the top-level general ontology of semantic web services, as objects of research, provides an unified formal description of the main characteristics of the web service, which are significant for solving problems of web services on all stages of web service life cycle. Also, such formal representation allows to prevent ambiguity in the understanding of semantics during the analysis of existing web services and provides opportunities for automated solving the web services tasks, namely: search, discovery, verification, composition, etc.

The use of descriptive logic apparatus to formalize the top-level general ontology of the web service allows to build ontological definitions of the web services tasks. These definitions are general, too. That is, they do not depend on the specific applied problem and the specific web services that used to solve it. Similarly, descriptions of specific services are also based on this top-level web service ontology, but are specific to the business task they solve. In fact, the definition of specific web services links to the ontology of the applied domain and to the top-level web service ontology.

Taking into account the great formalization opportunities and available DL reasoners, we can assert that the proposed approach provides the possibility of automated and effective solving the specified tasks. Note, the defined tasks and web services ontologies cover only simplified model of the representation of web services, in particular, functional Input/Output model extended by contexts descriptions. They provide the basis for further improvement by expanding this model and involving additional characteristics of both functional and process models of representation to the specified task ontologies. This will certainly increase the accuracy of the obtained result, but is the subject of further research.

Below, the section "Web Service Discovery and Composition Tasks" presents the general definition of web services problems such as discovery and composition problems. The section "Ontological approaches for solving the web services tasks" specifies the system of the main five types of ontologies that create a general formal information environment to solve the main problem. The section "Description logics with concrete domains. Main definitions" provides the theoretical basis necessary to construct the web services tasks ontologies, namely: the basic elements of descriptive logic with concrete domains are determined.

And finally, in the section "Defining the $\boldsymbol{\mathcal{A}(\mathcal{D})}$ ontologies of the tasks of the web services discovery and composition" the descriptive logic with the concrete domain $\boldsymbol{\mathcal{ALC}}(\mathcal{D}_{match})$ is constructed. Also, based on $\boldsymbol{\mathcal{A}}(\mathcal{D}_{match})$ they are determined with it the ontologies of the web services discovery and composition problems as an extensions of the top-level web service ontology.

## 1. Web service discovery and composition tasks

Not formally, the discovery problem is the task of finding web services that match the search request with the maximum degree of similarity. The search request is a hypothetical virtual web service that satisfies the requester's requirements. At the same time, similarity should be considered and evaluated as a complex multidimensional concept, because it is very rarely possible to achieve a full match. The adequacy of the obtained result largely depends on the accuracy and completeness of the formal representation of both the purpose of the search (expressed by the target request) and the available web services. Semantic web services are defined on two levels: functional - with its main characteristics, and procedural, which contains the order, conditions and results of operations within the web service environment.

Depending on the set of web service characteristics, there are several types of the functional models, the most common of which is the IOPE model. It determines the web service by four sets: Input, Output parameters, Preconditions and Effects of the web service. Clearly, IOs (Inputs-Outputs) provide syntactic information about the web service, while PEs (Preconditions-Effects) reflect their semantics. So, this web service model is chosen as the basis for further solving problems.

Considering the fact that the web service specification includes the possibility of defining its short textual description (for example, in the Documentation tag of the web service's WSDL description). It may contain information about web service's purpose and functionality that is the web service's context. It should be noted that this information may be considered in the process of matching the web service with the discovery (or composition) goal. At least, a meaningful structured context can allow to significantly reduce the set of web services that will be subjects to further processing at the initial stage of solving the problem. It may be realized by excluding services that have completely different purpose and functionality.

So, we can extend the IOPE model of the web service [1, 2] with the context description [3] and define the web service as:

$$S = \{In_s, Out_s, Pre_s, Eff_s, Doc_s\}, \quad (1)$$

where $In_s$- the finite set of input parameters of the web service,

$Out_s$ – the finite set of output parameters of the web service,

$Pre_s$- the finite set of preconditions of the web service,

$Eff_s$ – the finite set of effects of the web service,

$Doc_s$ – the finite set of semantic items of the web service's context description.

The discovery goal, or the target request, is the definition of the virtual web service which expresses the requester's needs and may be defined as:

$$Q = \{In_Q, Out_Q, Pre_Q, Eff_Q, Doc_Q\}, \quad (2)$$

where $In_Q$- the finite set of input parameters of the target web service,

$Out_Q$ – the finite set of output parameters of the target web service,

$Pre_Q$- the finite set of preconditions of the target web service,

$Eff_Q$ – the finite set of effects of the target web service,

$Doc_Q$ – the finite set of semantic items of the target web service's context description.

According to the given model, the discovery task at the functional level can be defined as a semantic search and discovery of web services based on a step-by-step matching descriptions of the available web services with the target request, which is the description of the abstract wanted web service. The purpose of this stage is to find a web service that implements the business task, or to form a set of the web services-candidates for the further construction of the composite web service, namely:

- determining the discovery goal as target request that is the representation of the virtual wanted web service;
- analyzing the web service context and defining the categories of the request and declared web services;
- matching context descriptions of the target request and the available web services to reduce the set of analyzed web services;
- syntactic matching the web services and the discovery goal by their input and output parameters;
- semantic matching the web services and the discovery goal by their preconditions;
- semantic matching the web services and the discovery goal by their effects;
- verification of the web service, matching the web service and its specification using DL resoners;
- verification of non-functional characteristics of the web services [4] such as the cost, the authority, the degree of trust in the web service, the reputation of the web service etc. Note, taking into account the non-functional web service characteristics increases the quality of the result of the web service selection, but it essentially complicates the task;
- determining the degree of a semantic suitability. Evaluating quantitative indicators of compliance of the web services and the target request and ranking the web services according to these degree values.

According to the given model, the declared web service responds to the target request if:

- the web service context is not less than the target request context;
- the preconditions of the declared web service are simpler than the preconditions of the target request, or at least they are satisfied so easy as the conditions of the target request;
- the effects of the declared web service is not less than the effects of the target request. In other words, the expected effects (defined by the target request) must be satisfied first of all;
- the web service produces output parameters, the set of which is not less than the set of output parameters of the target request;
- the declared web service requires input parameters, the set of which is not greater than the set of input parameters of the target request.

Formally, for the web service and target request represented by the extended IOPE model, the task of establishing structural correspondence can be defined as follows.

*Definition 1*. Let's the web service $S=\{s|\ s = (I; O; P; E; D)\}$ from the repository $\mathcal{R}$ is represented based on the acyclic T-BOX $\mathcal{T}$, where $P$ is the finite set of the preconditions of the web service, $I$ is the finite set of its input parameters, $E$ is the finite set of the effects of the web service, $O$ is the finite set of the output parameters of the web service, and the target request $Q$ is defined as the wanted virtual web service $Q= (P'; I'; E'; O'; D')$, where $P'$ is the finite set of its preconditions, $I'$ is the finite set of its input parameters, $E'$ is the finite set of its effects, $O'$ is the finite set of its output parameters. The web service $S$ matches the target request $Q$ if $D' \sqsubseteq D$, $I \sqsubseteq I'$, $O' \sqsubseteq O$ and it exists such interpretation $\mathcal{I}$[5] of T-BOX $\mathcal{T}$ as $P'^{\mathcal{I}} \subseteq P^{\mathcal{I}}$, $E^{\mathcal{I}} \subseteq E'^{\mathcal{I}}$[6].

Figure 1 bellow shows the graphic representation of the discovery task.

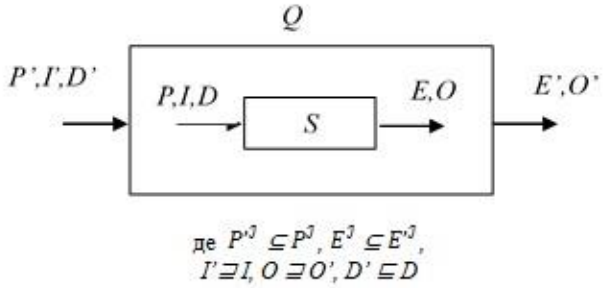


**Figure 1:** The graphic representation of the discovery task

At the process level the discovery problem solving is complicated by the need to take into account the behavioral aspects of web services and the internal semantics of the process itself. However, it should be noted that the success of the analysis of the process model depends entirely on the quality of semantic annotations of the web services. The main features of the process of the semantic annotations of web services are:

- semantic annotations have to be based on a single formal tools that may be easy automated (for example, ontologies, descriptive logics etc.), and on the unified system of concepts of the application domain;
- it has to provide the annotations of all elements and steps of the process, which are significant for the web service execution and express its meaning (i.e. the completeness of the annotation system)
- it should not define redundant annotations that overload the description of the web service and complicate its processing.

The process of the web service may be considered as a sequence of operations that are significant for the web service performance. Each of them has own characteristics (input and output parameters, execution conditions) and produces certain effects. Actually, these are atomic web services that may or may not depend on each other, so they can be executed sequentially or in parallel. These relations determine the dynamics of the web service and make it necessary to compare the main components of the process as atomic web services, and also to take into account their dependencies in time. To solve tasks this requires using dynamic formalisms, such as, for example, temporal [7, 8] or dynamic [9] descriptive logics.

The target request, or the wanted virtual web service, is the goal of the web service task solving. Informally, it is the business-task description that should been solved with existing web services. In real life, it is very difficult to find one web service which satisfies the goal. So, it is necessary to build the composited web service from the existing web services that will solve the given business task, that is, it will be as similar as possible to the target request. In this case, the discovery task is reduced to finding the web services - candidates for composition, and the composition task is to build a complex web service as a chain of the web services – candidates that meets the goal. It should be constructed based on dependencies of the web services and correspondences of their characteristics. In this case, the problem of defining correspondences of the web services is critical, in particular, correspondences of "predecessor-follower" type for ordering services into the chain.

*Definition 2*. There is a set of the web services – candidates $\mathcal{S}$ for composition and the target request (the goal) $Q =\{P_Q; I_Q; E_Q; O_Q; D_Q\}$ that are represented by the model (1). The web services $S_1= \{P_{S1}; I_{S1}; E_{S1}; O_{S1}; D_{S1}\}$ and $S_2= \{P_{S2}; I_{S2}; E_{S2}; O_{S2}; D_{S2}\}$ may be considered as sequential $S_1 \rightarrow S_2$ in the result composited web service, if:

- the contexts of the both web services are included in the context of the goal: $D_{S1} \sqsubseteq D_Q$ та $D_{S2} \sqsubseteq D_Q$,

- the output parameters of the first web service provide the values for the input parameters of the second web service: $I_{S2} \sqsubseteq O_{S1}$,
- the effects produced by the first web service provide the preconditions of the second web service will be truth: ${E_{S1}}^{\mathcal{J}} \rightarrow {P_{S2}}^{\mathcal{J}}$.

Note, this definition is reduced as it does not consider complex branching, when the web service follower, for example, has several predecessors.

## 2. Ontological approaches for solving the web services tasks

Ontological approaches for solving tasks provide the maximum involvement of ontologies at all stages of solution building. It allows achieving a unified strict formalization of all aspects and ensures the possibility of an automated solution with the involvement of reasoners. So, to solve the complex of the web services tasks on all stages of their life cycle [16] it is proposed to use a system of DL ontologies that contains next ontology types:

- *The named entities taxonomy* to categorize the target request and available web services while analyze their contexts [3]. This taxonomy is in some ways similar to data bases used for matching texts, for example, to the system of classes of named entities proposed by Microsoft [10], or to the special dictionaries of concepts (for example, Wordnet [11]). But, taking into account the formalization of the web service representation, this taxonomy should contribute to defining the similarity of the goal and existing declarations, and, first of all, it should allow the categorization of existing web services: by application area, by types of implemented functions, etc. So, it should include all important semantic aspects of the web services contexts. Its main goal is definition of the most complete generalized system of concepts providing the possibility of a structured semantic description of the context of any web service, regardless of its usage area or business purposes. In [3] it was proposed the prototype of such DL ontology (*DocumentationOntology*) defined with the description logic ALC. This ontology provides the web services categorization by their purposes, functionalities, knowledge areas etc.
- *The integrated ontology of the application domain*. It describes the application area of the business task which have to be resolved. It should be noted, the business task, in general, can rely on several ontologies in the application domain. In this case, it is efficient to integrate them into a general unified ontology of the domain and use a single T-BOX for all web services that must be annotated in the domain. T-BOX, at the same time, has to include all concepts that must be represented in the application domain.
- *The top-level web service general ontology*. It gives the general web service definition, i.e. it specifies the basic entities of the web service that present its main characteristics. The web service ontology must reflect the definition of the model of the web service. So, its T-BOX depends on the selected representation model. DL ontology of the web service represented by extended IOPE model on functional level will be present bellow.
- *The web services tasks ontologies*. Actually, they are extensions of the top-level general ontology of the web service and supplement its T-BOX with statements and logical rules that formally describe the task itself and enable automated decision-making (for example, on the correspondence of the web service declaration and the goal or on the correspondence of services to each other in the composition chain) through the DL reasoners usage.
- *The web services ontologies* allow translating the web services declarations to the formal DL language. These ontologies rely on the top-level general web service ontology and integrated ontology of application area. Also, they can use the entities from the taxonomy of the general structure of concepts (the special concepts dictionary).

To specify this ontology system previous researches were based on the basic DL $\mathcal{ALC}$. It is the effective tool for knowledge representation and ensures the execution of standard deductions. But, DL-specifications of the web services tasks require the modelling «abstract» objects (parameter, condition, service etc.) and its «specific» characteristics, in particular, time and space characteristics. These specific values can't be represented in the model as elements of the area $\triangle$. This is because when moving one interpretation to another, these "specific" elements and the links between them will change, while there is a requirement that they should be unchanged. When using descriptive logics with concrete domains, it is allocated a separate area with a fixed set of predicates. So, the interpretation "in the new sense" consists of the interpretation "in the old sense" («abstract») extended by the fixed «concrete» interpretation (with the set of predicates). Obviously, this also requires using the special roles to link these "abstract" and "concrete" elements of constructors to build concepts based on such roles and "concrete" predicates.

So, the focus of this study is the DL formalization of web services tasks, in particular, the tasks of the web services discovery and composition, using the apparatus of the descriptive logics with concrete domains.

At first, we will present the main theoretical aspects and syntax of DL with concrete domains, which we will rely on further when defining the specified tasks ontologies.

## 3. Description logics with concrete domains. Main definitions

Essentially, a concrete domain is a structure of some predicate signature, i.e. the set with relations specified on it. Formally, it can be defined as follows.

*Definition 3* [12, 13]. The concrete domain is the pair $\mathcal{D}=(D, \Phi)$, where D is a non-empty set, $\Phi$ is a set of predicates defined on *D*. Assume that the given set of the predicate symbols is *PN*, each symbol $P \in PN$ has valence *n*, $\Phi$ matches it n-ary relation $P^{\mathcal{D}} \subseteq D^n$. In addition, we will take into account the following agreements:

- $\mathcal{D}$ contains unary predicate, i.e. the symbol $\top$ is always included in *PN* and interpreted as $\top^{\mathcal{D}} = D$.
- $\mathcal{D}$ is always closed with respect to the addition, i.e. the n –ary predicate symbol $\neg P$ exists for each n-ary predicate symbol *P* in the signature *PN* and it is interpreted as $\triangle^n / P^{\mathcal{D}}$.

It is clearly that any domain can be converted into the area satisfying this agreement.

Based on this definition in [14] it was introduced the description logic with the concrete domain $\mathcal{A}(\mathcal{D})$.

*Definition 4*. Let's some concrete domain $\mathcal{D}$ is given. It includes the set of the predicate symbols PN and following finite sets of the predicate symbols: CN is the set of atomic concepts, RN is the set of atomic roles, AF ⊆ RN is the set of atomic abstract attributes, CF is the set of atomic concrete attributes. Composed concrete attribute is the chain $f_1 \ldots f_k h$ with k≥1 of atomic abstract attributes $f_i \in AF$ and one concrete attribute h ∈ CF.

*Definition 5* (Syntax of the $\mathcal{A}(\mathcal{D})$ logic). Concepts of the $\mathcal{A}(\mathcal{D})$ logic are defined by grammar:

$$\top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C \mid \exists [u_1, \ldots, u_n].P \ ,$$

where $A \in CN$, $R \in RN$, $u_1, \ldots, u_n$ – arbitrary (composed) attributes, $P \in PN$ – *n*-ary concrete predicate. So, it should be noted that only new concept type $\exists [u_1, \ldots, u_n].P$ is added to syntax of DL $\mathcal{ALC}$.

*Definition 6* (Semantic of $\mathcal{A}(\mathcal{D})$) [14]. Interpretation $\mathcal{I} = (\triangle, .^{\mathcal{I}})$ is defined similarly as for $\mathcal{ALC}$ logic but there are next additions:

- the sets $\triangle$ and *D* do not intersect each other;
- the function $f^{\mathcal{I}}: \triangle \to \triangle$ is matched to each atomic attribute $f \in AF$;
- the function $h^{\mathcal{I}}: \triangle \to D$ is matched to each atomic concrete attribute h ∈ CF;
- the composed concrete attribute $u = f_1 \ldots f_k h$ is interpreted as the composition of the partial functions: $u^{\mathcal{I}}(x) = h^{\mathcal{I}}(f_k^{\mathcal{I}}(\ldots f_1^{\mathcal{I}} \ldots))$, its result is the partial function $u^{\mathcal{I}}: \triangle \to D$;
- the new concept type $\exists [u_1, \ldots, u_n].P$ is interpreted as follow:

$$(\exists [u_1, \ldots, u_n].P)^{\mathcal{I}} = \{e \in \triangle \mid \exists x_1, \ldots, x_n \in D: u_1^{\mathcal{I}}(e) = x_1 \wedge \ldots u_n^{\mathcal{I}}(e) = x_n \wedge \langle x_1, \ldots, x_n \rangle \in P^{\mathcal{D}}\}.$$

If all functions $u_i^{\mathcal{I}}$ are defined in the point e then the formula above may be reduced:

$$(\exists [u_1, \ldots, u_n].P)^{\mathcal{I}} = \{e \in \triangle \mid \langle u_1^{\mathcal{I}}(e), \ldots, u_n^{\mathcal{I}}(e) \rangle \in P^{\mathcal{D}}\}$$

The concept $\exists u.\top$ expresses a set of all points where the attribute u is defined. $\top$ is not concept here, it is a concrete predicate. So, the concepts of this logic are closed under the negation operator:

$$\neg \exists [u_1, \ldots, u_n].P \equiv \neg \exists u_1.\top \sqcup \ldots \sqcup \neg \exists u_n.\top \sqcup [u_1, \ldots, u_n].\neg P \ .$$

Authors in [13] consider examples of the concrete domains and, also, issues of resolvability of DLs with concrete domains in general and for given domains in particular. It is proved that [15]:

- if the domain $\mathcal{D}$ is resolvable then the logic $\mathcal{ALC}(\mathcal{D})$ is resolvable too;
- if the domain $\mathcal{D}$ belongs to the class Pspace by complexity then the logic $\mathcal{ALC}(\mathcal{D})$ is *PSpace-full* [17].

The ontology system and the specification of DL $\mathcal{A}(\mathcal{D})$ presented above are led in the basis of DL-definition of the web services discovery and composition tasks.

## 4. Defining the $\mathcal{A}(\mathcal{D})$ ontologies of the tasks of the web services discovery and composition

As mentioned above, the ontology of the web service task, in fact, is a set of DL statements, which is an extension of the top-level general ontology of the web service and it depends on the model of the web service

representation. Based on the extended *IOPE* functional model, we define the *T-BOX* of the top-level general ontology of the web service *ServiceOntology* as follows (the prototype of the *DocumentationOntology* is given in [3]).

**T-BOX**

*Service, Condition, Effect, Parameter, Name, Value, Type, Context, DocItem;*

*Service* $\sqsubseteq$ *has.DocItem;*

*Service* $\sqsubseteq$ *has.Context;*

*DocItem* $\sqsubseteq$ *Context;*

*DocItem* $\sqsubseteq$ *has.ItemName;*

*DocItem* $\sqsubseteq$ *has.ItemValue;*

*DocItem* $\sqsubseteq$ *BusinessDomenCategory* / *link to *DocumentationOntology* (defines the business domain category)

*DocItem* $\sqsubseteq$ *BFunctionType* /* link to *DocumentationOntology* (defines the type of the function realized by the web service)

*DocItem* $\sqsubseteq$ *MainEntitiesTypes* / *link to *DocumentationOntology* (defines types of the entities from the named entity taxonomy)

*ItemName* $\sqsubseteq$ *hasType.{String};*

*ItemValue* $\sqsubseteq$ *hasType.Type;*

*Parameter* $\sqsubseteq$ *has.Name;*

*Parameter* $\sqsubseteq$ *has.Value;*

*Name* $\sqsubseteq$ *hasType.{String};*

*Value* $\sqsubseteq$ *hasType.Type;*

*Condition* $\sqsubseteq$ *hasValue.Boolean;*

*Service* $\sqsubseteq$ *hasInputParameters.Parameter; /* *I* – component of the web service

*Service* $\sqsubseteq$ *hasOutputParameters.Parameter /* *O* – component of the web service

*Service* $\sqsubseteq$ *hasPreCondition.Condition; /* *P* – component of the web service

*Service* $\sqsubseteq$ *hasEffect.Effect; /* *E* – component of the web service

*Service* $\sqsubseteq$ *hasServiceContext.Context; /** extension of IOPE web service model by context

*Type = {String, Numeric, Boolean, …}.*

This study focuses on the first stage of the specification of the discovery task by means of $\mathcal{A}(\mathcal{D})$, namely, for the extended IO model of the web service, and foresees further development to the extended functional IOPE model of the web service given above, as well as to the process model of the web service.

Now let's define the concrete domain $\mathcal{D}_{match}$={$\mathcal{S}$, $\Phi$}, where $\mathcal{S} = \mathcal{S}_P \sqcup \mathcal{S}_C$ is the union of both sets: $\mathcal{S}_P$ – the set of the possible subsets of instances of the concept Parameter and $\mathcal{S}_C$ is the set of the possible subsets of instances of the concept Context, $\Phi$ is the set of the predicates defined on $\mathcal{S}$. For now, we will consider $\Phi$ with unary (P1) and binary (P2) predicate symbols: $P1 = \{T\}$, $P2 = \{\subseteq, \supseteq, \cong, \ncong, \nsubseteq, \nsupseteq\}$. Note that we use set theory operators. This is due to the features of the chosen domain: elements of the domain are the sets of the instances.

## 4.1. The discovery task ontology MatchingOntology.

Let's define T-BOX of the discovery task ontology *MatchingOntology* based on $\mathcal{A}(\mathcal{D}_{match})$ logic as follow.

*S* $\sqsubseteq$ *Service*; /*link to the top-level general web services ontology *ServiceOntology*;

*Q* $\sqsubseteq$ *Service*; /* link to the top-level general web services ontology *ServiceOntology*.

Let's introduce the complex concept *DiscoveryObject* that will separate abstract services (target requests) from the web services available in the registry. Its instances are pairs "web service - target request" that correspond or do not correspond to each other.

*DiscoveryObject* $\sqsubseteq$ *hasService.Service* /*link with *ServiceOntology*;

*DiscoveryObject* $\sqsubseteq$ *hasQuery.Service*; /* link with *ServiceOntology*;

Let's *hasAvailableValues* is the concrete attribute in $\mathcal{A}(\mathcal{D}_{match})$ description logic with values in $\mathcal{S}$. In this case, the concepts *InputMatching* and *OutputMatching* define the set of objects that describe the pairs "web service – target request" that match each other by the input/output parameters. Immediately, it should be noted that all the given chains use only functional roles, where:

*InputMatching* $\equiv$ *DiscoveryObject* $\sqcap$ *(hasService hasInputParameters hasAvailableValues* $\subseteq$ *hasQuery hasInputParameters hasAvailableValues),*

*OutputMatching* ≡ *DiscoveryObject* ⊓ (*hasQuery hasOutputParameters hasAvailableValues* ⊆ *hasService hasOutputParameters hasAvailableValues) and ContextMatching* ≡ *DiscoveryObject* ⊓ (*hasQuery hasContext hasAvailableValues* ⊆ *hasService hasContext hasAvailableValues).*

So, the concept *DiscoveryServices* ≡ *InputMatching* ⊓ *OutputMatching* ⊓ *ContextMatching* specifies the set of objects that describe the pairs "web service – target request" match each other by input, output parameters and contexts, together.

### 4.2. The ontology of the task of the composition of the web services *CompositionOntology*.

Let's define the T-BOX of the composition task ontology *CompositionOntology* based on $\mathcal{A}(\mathcal{D}_{match})$ as follow.

*S1* ⊑ *Service*; /*link with *ServiceOntology*

*S2* ⊑ *Service*; /*link with *ServiceOntology*

*Q* ⊑ *Service*; /* link with *ServiceOntology*

Let's introduce the special complex concept *CompositionObject*. Its instances are pairs "web service – web service" match (or not) each other.

*CompositionObject* ⊑ *hasService1.Service* /*link with *ServiceOntology*;

*CompositionObject* ⊑ *hasService2.Service*; /*link with *ServiceOntology*;

*CompositionObject* ⊑ *hasQuery.Service*; /*link with *ServiceOntology*.

Let's *hasAvailableValues* is the concrete attribute in $\mathcal{A}(\mathcal{D}_{match})$ description logic with values in $\boldsymbol{S}$. In this case, the concept *ServiceMatching* ≡ *CompositionObject* ⊓ (*hasService2 hasInputParameters hasAvailableValues* ⊆ *hasService1 hasOutputParameters hasAvailableValues)*⊓ *(hasService2 hasContext hasAvailableValues* ⊆ *hasQuery hasContext hasAvailableValues)* ⊓ *(hasService1 hasContext hasAvailableValues* ⊆ *hasQuery hasContext hasAvailableValues)* specifies the set of objects describing the pairs "web service – web service" that match each other by input/output parameters. I.e. we can construct the composited web service from the web services *S1* and *S2* as their sequence chain *S1*→*S2*. Note that contexts of the both web services must be included into the context of the target request. This condition may seem too strict. Really, it may be sufficient to have a non-empty intersection of the value sets of the web service contexts and the target request.

The further development of the given tasks ontologies is related to extending them with concepts that will define the pairs "web service- target request" that are similar by preconditions and effects, respectively. In the future they should expand the definition of complex concepts DiscoveryServices and CompositionObject, improving tasks definitions by additional characteristics of the web services suitability. In addition, it should be noted that the composition chain, in reality, is much wider than a sequence of web services compatible by input/output. It may contain branches, the definition of which also involves the further development of the CompositionObject ontology.

## 5. Conclusions

The purpose of this study is creating effective solutions for matching web services within the framework of solving the main web services tasks, in particularly, the discovery and composition problems. Proposed approach is based on the idea of maximal usage of ontologies for resolving the tasks. This implies using DL ontologies for formally describing the web services and the goal (as the virtual web service) and to represent the tasks, data, processes and algorithms. Five main types of ontologies were defined: (1) the general taxonomy of the named entities, (2) the top-level general web service ontology, (3) the unified ontology of the application domain, (4) the web services tasks ontologies and (5) the specific web services ontologies. The set of these ontologies forms the unified information system, all ontologies of which are related. Thus, the ontologies of the specific web services include relations with the ontologies of all other four types. To represent all these ontologies the descriptive logic tools were applied. In this research solving the problems was based on using DLs with concrete domains. In particularly, the logic $\mathcal{A}(\mathcal{D}_{match})$ was proposed. In this logic the concrete domain is specified by the set of all subsets of the instances of the concept Parameter. It includes all possible parameters of available and wanted web services with set theoretic operations on it. Applying $\mathcal{A}(\mathcal{D}_{match})$, defined in this way, allows to represent the main features of the web service by functional roles of the description logic. Also, it permits to build concepts which instances give solutions of the specified tasks. The

obtained solutions are complex concepts constructed from chains of the functional roles of the ontology, concrete attributes of the concepts and set theoretic operations of descriptive logics. Thus, the concept *DiscoveryServices* defines the pairs of the web services ("web service – target request") that match one other, and the concept *CompositionObject* determines the pairs of web services ("web service-web service") related as «predecessor-follower». Taking into account the great formalization opportunities and available DL reasoners, this approach provides the possibility of automated and effective solving the specified tasks.

But, it should be noted that proposed tasks ontologies are reduced because:

1. They propose the solutions based on the simplified model of the web services representation which only includes matching input, output parameters and contexts.
2. Only the full compliance is considered for the discovery task. To form sets of web services – candidate that cover he target request it is necessary ease compliance conditions and consider target decomposition options.
3. For the web service composition task the different variants of complex branching and the cases of several predecessors that ensure the execution of the follower service are not included.

However this approach is open, and the proposed tasks ontologies provide the extension possibilities to create solutions of the tasks with step-by-step reducing the limits indicated above. It is the subject of further investigations.

Another important aspect is the evaluation of solvability of the $\mathcal{A}(\mathcal{D}_{match})$ logic which actually leads to defining the solvability of the $\mathcal{D}_{match}$ domain. It is also the subject of further investigations.

## Acknowledgements

The author would like to thank the Institute of Software Systems of the National Academy of Sciences of Ukraine for supporting this research.

## References

[1] F. Baader, C. Lutz, M. Milieie, U. Sattler, F. Wolter, A Description Logic Based Approach to Reasoning about Web Services, in: Proceedings of the WWW 2005 Workshop on Web Service Semantics (WSS2005), 2005.

[2] F. Baader, C. Lutz, M. Milieie, U. Sattler, F. Wolter, Integrating Description Logics and Action Formalisms for Reasoning about Web-services. Technical Report, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, volume LTCS-05-02, LTCS-Report 05-02, 2005.

[3] O. Zakharova, Context web services matching in the discovery task resolving. Ontological approaches. Problems in programming, № 2-3 (2020) 39-49.

[4] O. Zakharova, Defining and resolving Web-services discovery problems using description logics formalism. Problems in programming, № 4 (2017) 66–78.

[5] S. Staab, R. Studer, Handbook on Ontologies. International Handbooks on Information Systems (INFOSYS). Second edition, 2009.

[6] C. Lutz, U. Sattler, A Proposal for Describing Services with DLs, in: Int. Workshop on Description Logics, 2002.

[7] E. Emerson, Temporal and Modal Logic J. van Leeuwen, editor, Handbook of Theoretical Computer Science, volume B: Formal Models and Semantics, pp. 995–1072, Elsevier, 1990.

[8] Hao, S. & Zhang, L. (2010) Dynamic Web Services Composition Based on Linear Temporal Logic. Conference: Information Science and Management Engineering (ISME), Volume: 1

[9] L. Chang, F. Lin, Z. Shi, A dynamic description logic for representation and reasoning about actions. KSEM'07: Proceedings of the 2nd international conference on Knowledge science, engineering and management, pp. 115–127, 2007. doi:10.1007/978-3-540-76719-0_15.

[10] Supported Categories for Named Entity Recognition - Azure Cognitive Services. Microsoft Docs

[11] Y. Peng, Two levels semantic web service discovery, in: Proceedings of Seventh International Conference on Fuzzy Systems and Knowledge Discovery, aug. 2010. doi:10.1109/FSKD.2010.5569776.

[12] C. Lutz, Description Logics with Concrete Domains - A Survey, in: Proceedings of Forth Conference on Advances in Modal Logics, volume 4 in King's College Publications, 2003.

[13] E. Zolin, Description logic, Special course lectures. URL:logic.math.msu.ru/staff/zolin/dl.

[14] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider, The Description Logic Handbook. Cambridge University Press, 2003.
[15] F. Baader, P. Hanskle, A schema for integrating concrete domains into concept languages, in: Proceeding of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91), pp. 452-457, Sydney, 1991.
[16] O. Zakharova, The technique of using Description Logics in the process of constructing a composite service at the functional level. Problems in programming. № 1 (2018) 77-91.
[17] S. Arora, B. Barak, Computational Complexity: A Modern Approach, Cambridge University Press, 2009. doi:10.1017/CBO9780511804090.