ЎЗБЕКИСТОН РЕСПУБЛИКАСИ
ОЛИЙ ВА ЎРТА МАХСУС ТАЪЛИМ ВАЗИРЛИГИ
ЎЗБЕКИСТОН РЕСПУБЛИКАСИ ФАНЛАР АКАДЕМИЯСИ
В.И. РОМАНОВСКИЙ НОМИДАГИ МАТЕМАТИКА ИНСТИТУТИЎЗБЕКИСТОН МИЛЛИЙ УНИВЕРСИТЕТИ
ТОШКЕНТ ДАВЛАТ ТРАНСПОРТ УНИВЕРСИТЕТИ
БУХОРО ДАВЛАТ УНИВЕРСИТЕТИ

Бухоро фарзанди, Беруний номидаги Давлат мукофоти лауреати, кўплабёш изланувчиларнинг ўз йўлини топиб олишида раҳнамолик қилган етук олим, физика-математика фанлари доктори Ғайбулла Назруллаевич Салиховнинг 90 йиллик юбилейларига бағишланади

АМАЛИЙ МАТЕМАТИКА ВА АХБОРОТ ТЕХНОЛОГИЯЛАРИНИНГ ЗАМОНАВИЙ МУАММОЛАРИ

ХАЛҚАРО ИЛМИЙ-АМАЛИЙ АНЖУМАН МАТЕРИАЛЛАРИ

2022 йил, 11-12 май

БУХОРО – 2022

MUNDARIJA

IDEOLOGY OF ONTOLOGY WEB LANGUAGE

Eshonqulov H.I.
Bukhara State University, Bukhara, Uzbekistan

OWL (Web Ontology Language) is a language designed to describe ontologies and developed by the W3 consortium specifically for this purpose. **OWL** is a search-friendly version of DAML+OIL, which is a mixture of DAML and OIL. The DARPA Agent Markup Language (DAML) was developed for DARPA, an agency of the US Department of Defense established to develop new technologies for the military. DAML is a language for "marking up agents" of the WWW, i.e. description language for various Web clients. The OIL (Ontology Inference Layer) language was developed by European researchers to describe Web ontologies. The DAML+OIL language can be considered as the initial version of OWL. OWL is built as an extension of RDF and RDFS. This means that the basic syntax of the language is still XML, and the basic construct is still the RDF triplet. In this context, the OWL language can be considered as an extended version of RDFS, allowing not only to describe classes and properties, but also to set restrictions on their use. In the language of descriptive logic, this means that the logic underlying OWL contains, in addition to describing relations, also axioms that specify the relationships between these relations and various restrictions on the latter.

At the time of this writing, the OWL version 2 standard was being developed. A translation of the preliminary working version of the Brief Introduction to OWL 2 can be found in the differences between versions of OWL lie in the logic behind which these versions are implemented. So, the first version is implemented on the logic **SHOIN(D)**:

1. The DL-Lite profile ensures that the implementation of the question answering procedure can be implemented as a relational database lookup.
2. In the OWL-R profile, the inference procedure is implemented as a logical search by chain rules implemented as triples in the RDF repository. OWL-R has two versions: OWL-R DL and OWL-R Full.

OWL Basics

Classes

The base element of OWL is the class of all classes, defined as owl : Class . The owl : Class class is an instance of the rdf s : Class class discussed above. Therefore, any OWL class must be specified as an instance of owl : Class . For example, if we want to define a class Human (person), then we must define a triple:

Human rdf : type owl : Class which in XML syntax would look like this:

<owl : Class rdf : ID= "Human" />

OWL also has two predefined classes:

- Class owl : Thing, which denotes the set of all individuals.
- Class owl : Nothing (nothing), denoting the empty set.

Each OWL class is a subclass of owl : Thing and a parent of owl : Nothing. Therefore, the inheritance tree in OWL forms a so-called complete lattice. Usually, it is enough to declare an entity as an instance of some class for this individual to automatically become a member of the owl : Thing class. Class inheritance in OWL is defined using the rdfs : subClassOf construct, i.e. just like in the RDF Schema language. As in RDF Schema, the fact that one class is a subclass of another means that all instances of the subclass are instances of the headclass.

OWL Editor Protege

Protege is probably the most popular OWL and RDF editor today. The editor is available for free download from the program website. The program is distributed under the so-called Mozilla Public License (Mozilla Open License), which allows you to use the program for free even in commercial development. The program is implemented in the Java language and therefore requires a preliminary installation of a Java machine. Usually, the Java machine is distributed along with Protege as part of the system for installing it on the user's computer. Protege is a convenient editor for creating OWL ontologies, and it is in this capacity that we will consider it. So, let's start describing the features of this editor. The version of the program considered here is number 3.4 and is based on the first version of the OWL language. There is also a beta version of Protege 4.0 which supports OWL 2.

Ontology store implementations based on OWL and RDF

In this section, we will review the most well-known implementations of ontology repositories that provide an interface in OWL and RDF. It should be noted that in comparison with relational

databases, ontology storage based on RDF triples, so to speak, suffer from low performance. As has been repeatedly mentioned above, a knowledge base based on RDF triples is a graph whose nodes are the subjects and objects of RDF predicates, and the edges of which are the RDF predicates themselves. It is difficult to implement a quick search in this graph.

REFERENCES

1. The Computer Science Ontology: A Comprehensive Automatically-Generated Taxonomy ofResearch Areas. Angelo A. Salatino, Andrea Mannocci, 01.06.2020.
2. Ontology in Computer Science. Silveira, M.S., Paula, M.G. 2003.

MODELING INDIVIDUAL LIFE TRAJECTORIES BY GRAPH

Ibragimov S.

samandar.ibragimov@univ-grenoble-alpes.fr

In various social sciences (sociology, urban planning, ...), a *life trajectory* is an object of study that serves as a support for the analysis of the motivations that determine the choices of an individual all alonghis biographical journey. The work initiated by M. Thériault *et al.* [3, 4] is a reference in Computer Science when dealing with life trajectory modeling of individuals. This work presents three trajectories (professional, family and residential), each of them being modeled as a sequence where *episodes* (period of time covering a stable status) and *events* (facts which alter the status of the episode causing its end and the beginning of a new episode) alternate. In our previous work by D. Noël *et al.* [1, 2], we proposed an extension of [3, 4] based on design patterns, ontologies and