

and others are violated. These include relevance logics and connexive logics which find out to justify causal implicative properties. Analyzing the properties of these logics involves clarifying the similarities and analogies of the schemes of these logics with other logics and models such as argumentation logics [2]. One of the prospects for further research is to study the connection of non-classical logics of this kind with the fuzzy models considered in this paper in the framework of causal and spatio-temporal relations of the semantic space.

### III. Conclusions

Approaches and models to the interpretation of fuzzy logics are proposed. The proposed models can be used in the interpretation of fuzzy logic formulas on the basis of metric meaning space for finite structures in order to analyze or synthesize schemes of fuzzy logic inference systems relevant to the structures of ontologies of subject areas.

### References

- [1] J.M. Lee. Introduction to Topological Manifolds, Springer, 2011, 452 p.
- [2] Finn V. K. Intel'ktual'nye sistemy i obshchestvo: Sbornik statei [Intelligent systems and society: Collection of articles], Moscow, KomKniga, 2006. 352 p.
- [3] D.E. Pal'chunov, G.E. Yakh"yaeva Nechetkie logiki i teoriya nechetkikh modelei [Fuzzy logic and theory of fuzzy models], Algebra i logika [Algebra and Logic], vol. 54 no.1, 2015, p. 109–118; Algebra and Logic, vol. 54, no. 1, 2015, p. 74–80.
- [4] Plesnevich, G.S. Binarne modeli znaniy [Binary knowledge models], Trudy Mezhdunarodnykh nauchno-tekhnicheskikh konferentsii «Intel'ktual'nye sistemy» (AIS'08) i «Intel'ktual'nye SAPR» (CAD-2008) [Proceedings of the International Scientific and Technical Conferences "Intelligent Systems" (AIS'08) and "Intelligent CAD" (CAD-2008)]. Nauchnoe izdanie v 4-kh tomakh [4 volumes], Moscow, Fizmatlit, 2008, vol.2. p. 424, 135–146 pp.
- [5] V.N. Vagin Deduktivnaya i obobshchenie v sistemakh prinyatiya reshenii [Deduction and generalization in decision-making systems], Moscow, Nauka. Gl. red. fiz.-mat. lit. 1988. 384 p.
- [6] V.P. Ivashenko Ontologicheskie struktury i parametrizovannye mnogoznachnye logiki [Ontological structures and parameterized multivalued logics], Information Technologies and Systems, 2023, Minsk, BGUIR, pp. 57–58.
- [7] V. Ivashenko Semantic space integration of logical knowledge representation and knowledge processing models Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh sistem [Open semantic technologies for intelligent systems], Minsk, BGUIR. Minsk, 2023, vol. 7. pp. 95–114.
- [8] Ivashenko, V. General-purpose semantic representation language and semantic space Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh sistem [Open semantic technologies for intelligent systems] Minsk, BGUIR, 2022. vol. 6. pp. 41–64.
- [9] G. Malinowski, Kleene logic and inference. Bulletin of the Section of Logic, 2014, vol. 1, no. 43.
- [10] D. Maximov, Logika N.A. Vasil'eva i mnogoznachnye logiki [Vasil'ev Logic and Multi-Valued Logics.] Logical Investigations, 2016, vol. 22. 82–107 pp.
- [11] V.P. Ivashenko. Modeli resheniya zadach v intellektual'nykh sistemakh. V 2 ch. Ch. 1 : Formal'nye modeli obrabotki informatsii i parallel'nye modeli resheniya zadach : ucheb.-metod. posobie [Models for solving problems in intelligent systems. In 2 parts, Part 1: Formal models of information processing and parallel models for solving problems: a tutorial] Minsk, BGUIR, 2020. 79 p.
- [12] A. Hussain, K. Ullah, M. Khan, T. Senapati, S. Moslem, Complex T-Spherical Fuzzy Frank Aggregation Operators With Application in the Assessment of Soil Fertility / IEEE Access, 2023, vol. 11.
- [13] E.P. Klement, M. Navara, Propositional Fuzzy Logics Based on Frank T-Norms: A Comparison // Fuzzy Sets, Logics and Reasoning about Knowledge 2000, vol. 15. 17–38 pp.
- [14] F. Giannini, M. Diligenti, M. Maggini, M. Gori, G. Marra, T-norms driven loss functions for machine learning. Applied Intelligence, 2023, vol. 53. 1–15 pp.
- [15] G. Heald Issues with reliability of fuzzy logic. Int. J. Trend Sci. Res. Develop, 2018, vol. 2 no. 6, 829-834 pp. 8, 2018.
- [16] Zh. Wang, J. K. George Fuzzy Measure Theory, Plenum Press, New York, 1991.
- [17] V.V. Golenkov. Otkrytyi proekt, napravlenyi na sozdanie tekhnologii komponentnogo proektirovaniya intellektual'nykh sistem [An open project aimed at creating a technology for the component design of intelligent systems], Otkrytye semanticheskie tekhnologii proektirovaniya intellektual'nykh sistem [Open semantic technologies for intelligent systems], 2013, pp. 55–78.
- [18] A.S. Narinyani. NE-factory: netochnost' i nedoopredelennost' – razlichie i vzaimosvyaz' [Non-factors: inaccuracy and underdetermination – difference and interrelation]. Izv RAN (RAS). Ser. Teoriya i sistemy upravleniya 5, 2000. pp. 44–56.
- [19] Yu. Manin, M. Marcolli. Semantic spaces. Published, Location, 2016. 32 p. (arXiv)
- [20] D.A. Pospelov. Situatsionnoe upravlenie: teoriya i praktika [Situational management: theory and practice], Moscow, Nauka, 1986. 288 p.

## ИНТЕГРАЦИЯ НЕЧЁТКИХ СИСТЕМ И ИХ ПАРАМЕТРИЧЕСКАЯ ИНТЕРПРЕТАЦИЯ ДЛЯ УНИФИЦИРОВАННОГО ПРЕДСТАВЛЕНИЯ ЗНАНИЙ

Ивашенко В. П.

В статье рассматривается проблема устойчивой интерпретации нечётких логических моделей. Предлагается подход на основе параметризованной нечёткой логики, где каждая логическая формула кроме значений истинности имеет набор одельных параметров. Параметризованные нечёткая логика позволяет объединить различные нечёткие логические системы. Модельные параметры используются для вычисления значений нечёткой истинности, как нечёткой меры. Рассмотрены модели и модельные параметры, связанные с метрическими пространствами, согласуемыми с метрическим смысловыми пространствами и являющиеся основой для интерпретации нечётких логических формул на онтологических моделях.

Received 03.04.2024

# State of ostis-systems Component Design Automation Tools

Maksim Orlov, Anna Makarenko, Ksenija Petrochuk

*Belarusian State University of  
Informatics and Radioelectronics  
Minsk, Belarus*

Email: orlovmaksimkonst@gmail.com, anna.makarenko1517@gmail.com, xenija.petrotschuk@gmail.com

**Abstract**—In the article, an approach to the design of intelligent systems is considered, focused on the use of compatible reusable components, which significantly reduces the complexity of developing such systems. The key means of supporting the component design of intelligent computer systems is the manager of reusable components proposed in the work.

**Keywords**—Component design of intelligent computer systems; reusable semantically compatible components; knowledge-driven systems; semantic networks; OSTIS Technology.

## I. Introduction

The main result of artificial intelligence is not the intelligent systems themselves, but powerful and effective technologies for their development. The analysis of modern technologies for designing intelligent computer systems shows that along with very impressive achievements, the following serious problems occur [1]–[3]:

- high requirements for the initial qualifications of users and developers. Artificial intelligence technologies are not focused on the wide range of developers and users of intelligent systems and, therefore, have not received mass distribution;
- modern information technologies are not oriented to a wide range of developers of applied computer systems;
- there is no general-unified solution to the problem of semantic compatibility of computer systems [4]. There are no approaches that allow integrating scientific and practical results in the field of artificial intelligence, which generates a high degree of duplication of results and a lot of non-unified formats for representation of data, models, methods, tools, and platforms;
- lack of powerful tools for designing intelligent computer systems, including intelligent training subsystems, subsystems for collective design of computer systems and their components, subsystems for verification and analysis of computer systems, subsystems for component design of computer systems;
- long terms of development of intelligent computer systems and high level of complexity of their maintenance and extension;
- the degree of dependence of artificial intelligence technologies on the platforms on which they are implemented is high, which leads to significant changes in technologies when transitioning to new platforms;
- the degree of dependence of artificial intelligence technologies on subject domains in which these technologies are used is high;
- there is a high degree of dependence of intelligent computer systems and their components on each other; the lack of their automatic synchronization. The absence of self-sufficiency of systems and components, their ability to operate separately from each other without loss of expediency of their use;
- increase in the time to solve the problem with the expansion of the functionality of the problem solver and with the expansion of the knowledge base of the system [5];
- lack of methods for designing intelligent computer systems. Updating computer systems often boils down to the development of various kinds of “patches”, which eliminate not causes of the identified disadvantages of updated computer systems but only some consequences of these causes;
- poor adaptability of modern computers to the effective implementation of even existing knowledge representation models and models for solving problems that are difficult to formalize, which requires the development of fundamentally new computers [6];
- there is no single approach to the allocation of reusable components and the formation of libraries of such components, which leads to a high complexity of reuse and integration of previously developed components in new computer systems.
- there is a variety of semantically equivalent implementations of problem-solving models, duplication of knowledge base and user interface components that differ not in the essence of these components but in the form of representation of the processed information;

To solve these problems, it is necessary to implement

a comprehensive technology for designing intelligent computer systems, which includes the following components:

- a model of an intelligent computer system [7];
- *a library of reusable components* and corresponding *tools to support component design of intelligent computer systems*;
- an intelligent integrated automation system for the collective design of intelligent computer systems, including subsystems for editing, debugging, performance evaluation, and visualization of developed components, as well as a simulation subsystem;
- methods of designing intelligent computer systems;
- an intelligent user interface;
- training subsystems for designing intelligent computer systems, including a subsystem for conducting a dialogue with the developer and the user;
- a subsystem for testing and verification of intelligent computer systems, including a subsystem for testing the compatibility of the developed system with other systems;
- an information security support subsystem for the intelligent computer system.

The key component of the technology for intelligent systems design is a component design that is represented as *a library of reusable components* and the corresponding *tools to support component design of intelligent computer systems*. With its help, it is possible to effectively implement the typical subsystems to support the design of intelligent computer systems.

## II. Analysis of existing approaches to solving the problem

The problem is the lack of accessibility and integration in artificial intelligence technologies, which have high initial qualification requirements, lack a unified semantic compatibility solution, and have a high degree of dependency on platforms, subject areas, and components, leading to long development times, high maintenance costs, and difficulties in reusing and integrating previously developed components in new systems.

Existing approaches to solving the problem include component libraries and package managers of programming languages and operating systems, as well as separate systems and platforms with built-in components and means for saving created components.

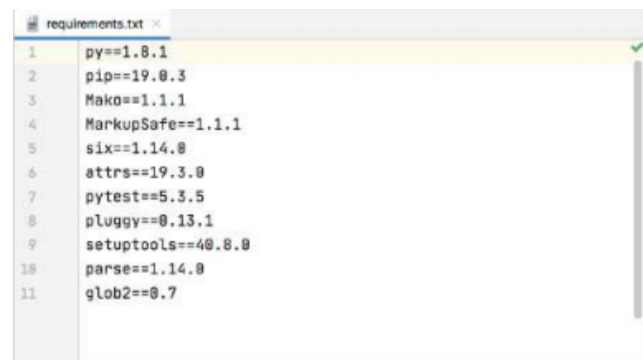
The components of the library may be implemented in different programming languages (which leads to the fact that for each programming language different libraries are developed with their own solutions to various common situations), and may be located in different places, which leads to the fact that the library needs a tool to find components and install them.

Modern package managers such as *npm*, *pip*, *papt*, *maven*, *poetry* and others have the advantage that they are able to resolve conflicts when installing dependent components, but they do not take into account the semantics of components,

but only install components by the [8] identifier. Libraries of such components are only a repository of components, without taking into account the purpose of components, their advantages and disadvantages, areas of application, the hierarchy of components and other information necessary for the intellectualization of component design of computer systems. Searching for components in *component libraries* corresponding to these package managers is reduced to searching by component identifier. Modern package managers are only "installers" without automatic integration of components into the system. Also a significant disadvantage of the modern approach is the platform dependency of components. Modern component libraries are oriented only to a certain programming language, operating system or platform.

The *pip* package manager is a package management system that is used to install packages from the Python Package Index, which is some library of such packages. Pip is often installed with Python. The pip package manager is used only for the Python programming language. It has many functions for working with packages:

- installation of a package;
- installation of a package of a specialized version;
- deletion of a package;
- reinstallation of a package;
- display of installed packages;
- search for packages;
- verification of package dependencies;
- creation of a configuration file with a list of installed packages and their versions;
- installation of a set of packages from a configuration file.



```

1 py==1.8.1
2 pip==19.0.3
3 Mako==1.1.1
4 MarkupSafe==1.1.1
5 six==1.14.0
6 attrs==19.3.0
7 pytest==5.3.5
8 pluggy==0.13.1
9 setuptools==40.8.0
10 parse==1.14.0
11 glob2==0.7

```

Figure 1: pip configuration file

The pip package manager works well with dependencies, displays unsuccessfully installed packages, and also displays information about the required package version in case of conflict with another package. An example of a pip package configuration file is shown in Figure 1. Another example of a package manager is *npm*. npm is a package manager for the JavaScript language. The