**ҒТАХР 27.41.15**

**10.51889/2959-5894.2025.91.3.015**

**С.Г. Алиева[1*], А.К. Альжанов[1], С.А. Жамалова[2]. Ж.Л. Мырзаева[2].**

[1]Л.Н. Гумилев атындағы Еуразия ұлттық университеті, Астана, Қазақстан
[2]Астана халықаралық университеті Астана, Қазақстан
[*]*e-mail: gulim_alieva@bk.ru*

# ОНТОЛОГИЯДАҒЫ ПӘНДІК БІЛІМНІҢ ҚҰРЫЛЫМЫ

*Аңдатпа*

Бұл мақалада Бұл мақалада информатика саласындағы алгоритмдік білімді формалды түрде көрсету және оны онтология түрінде модельдеу мәселесі қарастырылады. Зерттеудің негізгі мақсаты – алгоритмдерді құрылымдық түрде сипаттап, оларды ақпараттық жүйелерде, жасанды интеллект саласында және білім беру процесінде қолдануға бейімдеу. Зерттеу барысында әдіснамалық негіз ретінде онтологияларды жобалауда кеңінен танылған Protégé бағдарламасы және OWL (Web Ontology Language) тілі пайдаланылды. Онтологияны құру кезінде «Algorithm» негізгі класы алынып, оның ішкі кластар жүйесі анықталды: SortingAlgorithm, SearchingAlgorithm, GraphAlgorithm, OptimizationAlgorithm, CryptographicAlgorithm және MachineLearningAlgorithm. Сонымен қатар, hasComplexity, hasDataStructure, hasApplication, usesAlgorithmType, hasStep сияқты объектілік қасиеттер және hasName, hasTimeComplexity, hasSpaceComplexity, hasAuthor, hasYearIntroduced секілді мәліметтік қасиеттер енгізілді. Бұл қасиеттер алгоритмдердің тиімділігін, қолдану саласын және тарихи контекстін формалды түрде сипаттауға мүмкіндік береді. Зерттеу нәтижелері алгоритмдік білімді онтология негізінде жүйелеу олардың автоматты түрде өңделуін қамтамасыз ететінін көрсетті. Мұндай тәсіл оқу процесінде алгоритмдерді меңгеруді жеңілдетіп қана қоймай, сонымен қатар жасанды интеллект жүйелерінде, семантикалық вебте және интеллектуалды ақпараттық жүйелерде қолданудың тиімділігін арттырады. Осылайша, ұсынылған онтология информатикадағы білімді формалды сипаттау мен автоматтандырудың жаңа деңгейін көрсетіп, әрі қарай кеңейтуге және нақты пәндік салаларға бейімдеуге негіз қалайды.

**Түйін сөздер:** онтология, алгоритм, Protégé, OWL, білімді құрылымдау, формализация, ақпараттық жүйе, жасанды интеллект.

С.Г. Алиева[1], А.К. Альжанов[1], С.А. Жамалова[2], Ж.Л. Мырзаева[2]
[1] Евразийский национальный университет имени Л.Н. Гумилева, г. Астана, Казахстан
[2] Международный университет Астаны, г. Астана, Казахстан

# СТРУКТУРА ПРЕДМЕТНЫХ ЗНАНИЙ В ОНТОЛОГИИ

*Аннотация*

В статье рассматривается проблема формального представления алгоритмических знаний в области информатики посредством моделирования алгоритмов в форме онтологии. Основная цель исследования заключается в структурировании алгоритмических понятий и их адаптации для использования в информационных системах, в приложениях искусственного интеллекта и в образовательном процессе. В качестве методологической основы были использованы редактор онтологий Protégé и язык OWL (Web Ontology Language). При построении онтологии в качестве базового был выбран класс «Algorithm», для которого выделены подклассы: SortingAlgorithm, SearchingAlgorithm, GraphAlgorithm, OptimizationAlgorithm, CryptographicAlgorithm и MachineLearningAlgorithm. Определены объектные свойства (hasComplexity, hasDataStructure, hasApplication, usesAlgorithmType, hasStep) и датированные свойства (hasName, hasTimeComplexity, hasSpaceComplexity, hasAuthor, hasYearIntroduced). Эти свойства позволяют формально описывать эффективность алгоритмов, области их применения и исторический контекст. Результаты исследования показали, что структурирование алгоритмических знаний на основе онтологии обеспечивает их автоматизированную обработку и повышает эффективность использования в учебном процессе, в системах искусственного интеллекта, в технологиях семантической паутины и в интеллектуальных информационных системах. Разработанная онтология демонстрирует новый

уровень формального представления знаний в информатике и может служить основой для дальнейшего расширения и интеграции в прикладные области.

**Ключевые слова:** алгоритм, онтология, формализация знаний, Protégé, OWL, семантическая паутина, искусственный интеллект.

S.G. Aliyeva[1], A.K. Alzhanov[1], S.A. Zhamalova[2], Zh.L. Myrzayeva[2]
[1] L.N. Gumilyov Eurasian National University, Astana, Kazakhstan
[2] Astana International University, Astana, Kazakhstan

**THE STRUCTURE OF DOMAIN KNOWLEDGE IN ONTOLOGY**

*Abstract*

This article addresses the issue of formally representing algorithmic knowledge in computer science through the modeling of algorithms in the form of ontology. The main goal of the study is to structure algorithmic concepts and adapt them for use in information systems, artificial intelligence, and the educational process. The methodological framework relies on the widely recognized Protégé ontology editor and the OWL (Web Ontology Language). In constructing the ontology, the "Algorithm" class was chosen as the core, with subclasses identified as SortingAlgorithm, SearchingAlgorithm, GraphAlgorithm, OptimizationAlgorithm, CryptographicAlgorithm, and MachineLearningAlgorithm. Object properties such as hasComplexity, hasDataStructure, hasApplication, usesAlgorithmType, and hasStep, as well as data properties including hasName, hasTimeComplexity, hasSpaceComplexity, hasAuthor, and hasYearIntroduced were defined. These properties make it possible to describe algorithms in terms of efficiency, application domains, and historical context in a formalized manner. The results demonstrated that structuring algorithmic knowledge within an ontology not only enables automated processing but also improves its practical application in education, artificial intelligence, semantic web technologies, and intelligent information systems. The developed ontology provides a new level of formal knowledge representation in informatics and serves as a foundation for further expansion and adaptation to specific applied domains.

**Keywords:** algorithm, ontology, knowledge formalization, Protégé, OWL, semantic web, artificial intelligence.

## Кіріспе

Қазіргі ақпараттық қоғамда білім мен деректер көлемі қарқынды өсіп келеді. Бұл жағдай оларды жүйелеу, құрылымдау және өңдеу тәсілдерін жетілдіруді талап етеді. Информатика саласында білімді ұсыну мен модельдеу мәселесі ерекше маңызға ие, себебі кез келген есепті тиімді шешу үшін ұғымдарды формалды сипаттау қажет. R. Studer, V. R. Benjamins және D. Fensel [1] білім инженериясының негізгі қағидаларын талдап, білімді құрылымдаудың теориялық негізін көрсеткен. Ал, T.R.Gruber [2] онтологияны «тасымалданатын білім сипаттамасының формалды спецификациясы»- деп анықтап, оны білімді ұсынудың әмбебап тәсілі ретінде қарастырады.

Онтология – ұғымдар жиынтығы мен олардың арасындағы қатынастарды сипаттайтын формалды модель. N. F. Noy мен D. L. McGuinness [3] онтология жасаудың алғашқы қадамдары мен әдістемесін ұсынса, N. Samaridi, E. Papakitsos және N. Karanikolas [4] онтологияны интеллектуалды жүйелерде білімді модельдеудің тиімді құралы ретінде сипаттайды. Сонымен қатар, C. Feilmayr және W. Wöß [5] онтология құруда қолданылатын құралдар мен әдістерді жіктеп, олардың артықшылықтары мен шектеулерін айқындаған. Қазіргі таңда кеңінен қолданылатын құралдардың бірі – Стэнфорд университеті әзірлеген Protégé редакторы. Бұл бағдарлама OWL (Web Ontology Language) тіліне негізделіп, визуалды түрде класстарды, қасиеттерді және индивидтерді құруға мүмкіндік береді [6]. Соңғы жылдары жасанды интеллект саласында онтологиялық модельдерді терең нейрондық желілермен біріктіру мәселесі де қарастырылып келеді. P. Hohenecker және T. Lukasiewicz [7] онтология негізіндегі пайымдауды нейрондық желілер көмегімен жүзеге асырудың мүмкіндіктерін көрсетіп, білімді өңдеудің жаңа тәсілдерін ұсынды. Бұдан бөлек, R. S. Kurmangali [8] семантикалық веб пен онтология интеграциясын қарастырып, олардың заманауи ақпараттық жүйелерде қолданылу жолдарын айқындаған.

Алгоритмдер информатикадағы негізгі ұғымдардың бірі болып табылады. Олар есептерді шешуге арналған қадамдық нұсқаулар жүйесін сипаттайды. Gao,W., Chen,Y [9] онтологияларды жобалаудағы алгоритмдік тәсілдерді талдап, олардың теориялық негізін ашқан. T. C. Jepsen [10] онтология негізінде білімді модельдеу мүмкіндіктерін қарастырса, Oliveira, de Rodrigues C. M., және басқалар [11] онтологияны білім беру жүйесіне енгізудің жолдарын сипаттап, практикалық қолдану салаларын айқындаған. Бұл зерттеулер алгоритмдерге арналған онтология құрудың ғылыми әрі практикалық тұрғыдан өзектілігін дәлелдейді.

Осыған байланысты зерттеу жұмысының мақсаты – алгоритмдік білімді онтология түрінде формализациялау және оны ақпараттық жүйелерде қолдануға бейімдеу болып табылады. Зерттеу болжамы – алгоритмдік білім онтология негізінде ұсынылған жағдайда, оны формалды түрде сипаттап, автоматты өңдеуге болады. Бұл білім беру жүйесінде де, жасанды интеллект саласында да тиімділікті арттырады.

**Зерттеу әдіснамасы**

Зерттеу 2024–2025 жылдары информатика саласындағы алгоритмдік білімді формалды түрде құрылымдау мәселесін талдау негізінде жүргізілді. Жұмыс зертханалық жағдайда орындалып, онтологияны құруға арналған бағдарламалық құралдар қолданылды. Онтологияны жобалау кезеңдері N. F. Noy мен D. L. McGuinness ұсынған әдістемеге және C. Feilmayr мен W. Wöß сипаттаған құралдар кешеніне сүйене отырып жүзеге асырылды.Онтологияны құру үдерісінде ең алғашқы және маңызды кезеңдердің бірі – пәндік саланың тезаурусын әзірлеу болып табылады. Тезаурус – нақты бір ғылыми немесе қолданбалы саладағы ұғымдар мен терминдердің жүйеленген жиынтығы, олардың арасындағы семантикалық (мағыналық) байланыстармен бірге сипатталады. Кез келген пәндік салада жиі қолданылатын терминдер нақты бір түсінікті немесе құбылысты білдіреді. Алайда әртүрлі ғалымдар мен зерттеушілер бір ұғымды әртүрлі атаулармен атайды, ал кейбір терминдер бір сала ішінде бірнеше мағынаға ие болуы мүмкін. Мұндай көпмағыналылық пен терминологиялық әркелкілік онтология құруда біртұтас тілдік модельді қажет етеді. Бұл қажеттілік тезаурус арқылы шешіледі.

Тезаурустағы семантикалық қатынастар (мысалы: *«бөлігі болып табылады»*, *«түрі болып табылады»*, *«қолданады»*, *«себепші болады»* т.б.) ұғымдар арасындағы мағыналық байланыстарды дәл әрі құрылымдық түрде көрсетуге мүмкіндік береді. Бұл онтологияның дәлдігін, қайта пайдалануға жарамдылығын және тұтастығын қамтамасыз етеді. Сонымен қатар, тезаурус тек терминдер тізімі ғана емес, өз алдына білім базасы ретінде де қолданылуы мүмкін. Осыған орай информатика саласындағы негізгі ұғымдар жүйеленіп, олардың мағыналық байланыстары анықталды. Нәтижесінде тезаурус көпмағыналылықты қысқартып, біртұтас тілдік модельді қамтамасыз етті.

Онтология – білім беру немесе ақпараттық жүйелер саласындағы ұғымдарды, олардың қасиеттерін және өзара байланыстарын сипаттайтын формалды модель. Онтология негізінде нақты саладағы терминология мен ұғымдар жүйесі қалыптасады. Онтологиялар OWL (Web Ontology Language) тілі негізінде құрылып, RDF, RDFS және SPARQL секілді стандарттарды қолданады. Protégé бағдарламасы. Protégé – Стэнфорд университеті әзірлеген ашық кодты онтология редакторы. Ол OWL тілін қолдап, визуалды түрде класс, қасиет, индивидтерді құруға мүмкіндік береді. Сонымен қатар, плагиндер арқылы логикалық қорытындылар жасауға, графтар тұрғызуға және нәтижелерді экспорттауға болады. Protégé қазіргі таңда білімді жүйелеудің ең кеңінен танылған құралдарының бірі болып саналады.

Зерттеу барысында онтологияның негізгі ұғымы ретінде «Algorithm» класы алынды. Algorithm – кез келген алгоритмдік процедураны сипаттайтын жалпы түсінік. Ол нақты есептерді шешуге арналған қадамдық нұсқаулар жиынтығын бейнелейді және барлық нақты алгоритмдер осы кластың ішкі түрлері ретінде модельденді. Келесі кезеңде «Algorithm» класының ішкі кластар жүйесі жасалды.

1. SortingAlgorithm (Сұрыптау алгоритмдері). Деректерді белгілі бір тәртіпке келтіруге арналған алгоритмдер: BubbleSort, QuickSort, MergeSort, HeapSort. Бұл кластың қасиеттері: hasTimeComplexity, hasDataStructure, usesAlgorithmType.

2. SearchingAlgorithm (Іздеу алгоритмдері). Құрылымнан немесе массивтен нақты мәнді табуға арналған алгоритмдер: BinarySearch, LinearSearch, DepthFirstSearch, BreadthFirstSearch. Бұл класта деректер құрылымы (Tree, Graph, Array) негізгі рөл атқарады.

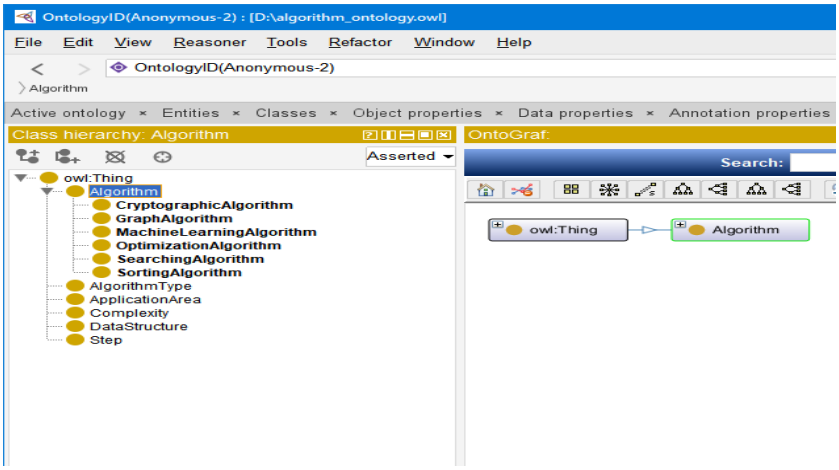
3. GraphAlgorithm (Графтық алгоритмдер). Граф құрылымында орындалатын операцияларға арналған алгоритмдер: DijkstraAlgorithm, PrimAlgorithm, KruskalAlgorithm, A*. Жол табу, байланыстылықты тексеру және т.б.

4. OptimizationAlgorithm (Оптимизация алгоритмдері). Мақсат функциясын ең төменгі немесе ең жоғары мәнге жеткізуге арналған алгоритмдер: GeneticAlgorithm, GradientDescent, SimulatedAnnealing.

5. CryptographicAlgorithm (Криптографиялық алгоритмдер). Мәліметтерді шифрлау, дешифрлау және қорғауға арналған алгоритмдер: RSA, AES, SHA256, DiffieHellman. Қасиеттеріне hasKeySize, isSymmetric, hasEncryptionType жатады.

6. MachineLearningAlgorithm (Машиналық оқыту алгоритмдері). Мәліметтерден үлгілерді үйренуге және болжауға бағытталған алгоритмдер: DecisionTree, RandomForest, KNN, SupportVectorMachine, NeuralNetwork. Бұл класс күрделі гиперпараметрлермен ерекшеленеді.

Осылайша, алты ішкі класс анықталып, олар белгілі бір есептерді шешуге бағытталған алгоритмдер тобын қамтиды (Сурет 1).




*Сурет 1. Алгоритмдер онтологиясының негізгі кластар иерархиясы (Protégé ортасы)*

Одан кейін онтологиядағы объектілік қасиеттер анықталды. hasComplexity, hasDataStructure, hasApplication, usesAlgorithmType және hasStep сияқты қасиеттер енгізіліп, алгоритмдер мен басқа ұғымдар арасындағы семантикалық байланыстар сипатталды.

1. hasComplexity (Күрделілігі бар). Бұл қасиет алгоритм мен оның уақыттық немесе кеңістік күрделілігі арасындағы байланысты сипаттайды (*Algorithm → Complexity*). Әр алгоритмнің тиімділігін анықтайтын негізгі көрсеткіш – күрделілік (complexity). Бұл қасиет арқылы O(n), O(n log n), O(1) сияқты нотациялармен берілетін уақыт/жад күрделілігі модельденеді. Мысалы: *MergeSort hasComplexity O(n log n)*.

2. Алгоритм қандай деректер құрылымымен жұмыс істейтінін сипаттайды (*Algorithm → DataStructure*). Кейбір алгоритмдер тиімді жұмыс істеу үшін арнайы деректер құрылымдарын қажет етеді (графтар, ағаштар, массивтер, стек).

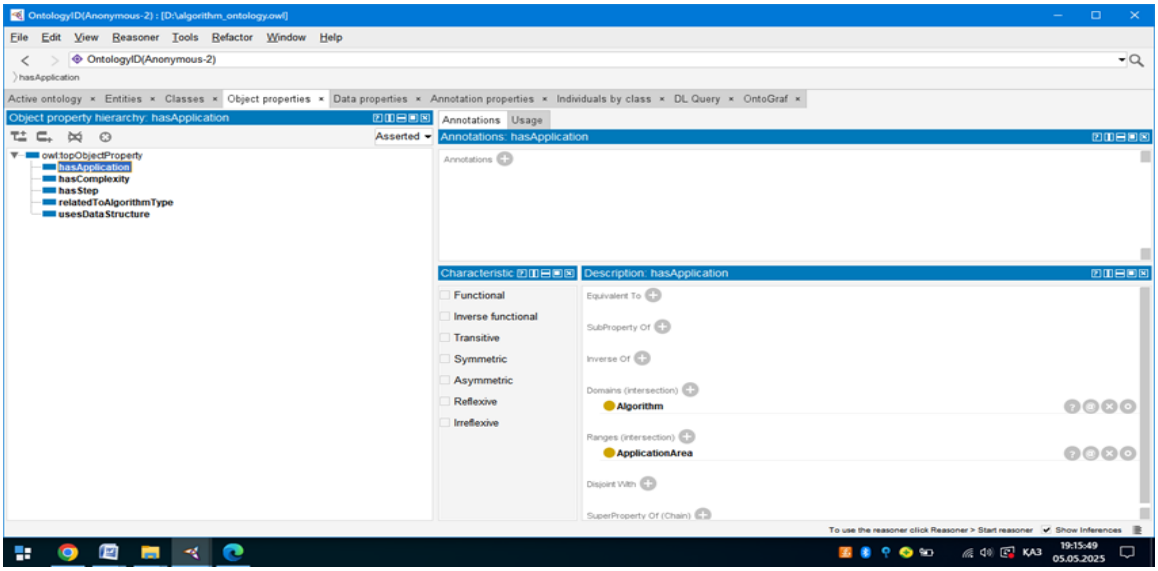
3. hasApplication (Қолдану саласы бар). Алгоритм нақты қандай есептерді немесе қолдану салаларын шешуге арналғанын сипаттайды (*Algorithm → ApplicationArea*). Бұл байланыс

алгоритмнің теориялық емес, нақты өмірлік қолданысын модельдейді (Сурет 2). Мысалы, деректерді сұрыптау, жол табу, шифрлау, медициналық диагностика.

4. usesAlgorithmType (Алгоритм түрін қолданады). Бұл қасиет алгоритмнің қандай стратегияға немесе парадигмаға жататынын көрсетеді (*Algorithm → AlgorithmType*). Алгоритмнің ішкі жұмыс механизмі мен логикалық құрылымына негізделген Divide and Conquer, Greedy, Dynamic Programming, Backtracking сияқты түрлерді модельдеуге мүмкіндік береді.

5. hasStep (Қадамдары бар). Бұл қасиет алгоритмнің орындалу процесіндегі жекелеген қадамдармен байланысын көрсетеді (*Algorithm → Step*). Күрделі алгоритмдерді формалды сипаттағанда оның әрбір қадамы жеке ұғым ретінде қарастырылып, білім жүйесін талдау мен визуализациялау үшін қолданылады.

Аталған объектілік қасиеттер алгоритм ұғымын онтологияда семантикалық тұрғыдан жан-жақты сипаттауға мүмкіндік береді. Олар арқылы кластар арасындағы мағыналық қатынастар көрнекі түрде көрсетіліп, жүйе формалды сұраныстарға (SPARQL) дайын болады. Мұндай байланыстарды формализациялау семантикалық вебтегі интеграциямен де үндеседі [10]. Бұл онтологияны білім беру жүйесінде, іздеу механизмдерінде және жасанды интеллект саласында қолдануға толық негіз қалайды (Сурет 2):




*Сурет 2. hasApplication объектілік қасиетінің сипаттамасы*

Сонымен бірге онтологияда мәліметтік қасиеттер (data properties) енгізілді. Олар алгоритмдердің мәтіндік және сандық сипаттамаларын айқындауға мүмкіндік берді: hasName, hasTimeComplexity, hasSpaceComplexity, hasAuthor және hasYearIntroduced.

1. hasName (Атауы бар). Алгоритмнің нақты атауын мәтін (string) түрінде көрсетеді (*Algorithm → string*). Әрбір алгоритмнің атауы міндетті түрде болуы тиіс, бұл қасиет арқылы алгоритмді басқа алгоритмдерден ажыратуға және сәйкестендіруге болады.

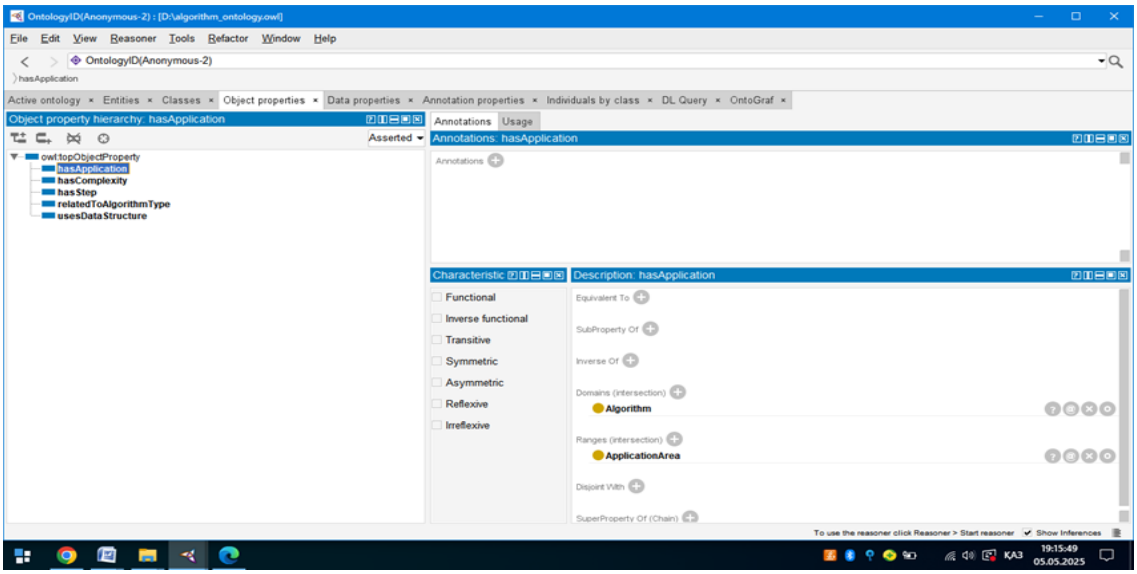
2. hasTimeComplexity (Уақыт күрделілігі бар). Алгоритмнің орындалу уақытына байланысты күрделілігін сипаттайды (*Algorithm → string*). Бұл көрсеткіш алгоритмнің тиімділігін бағалауда негізгі параметр болып табылады және теориялық немесе тәжірибелік талдауда жиі қолданылады.

3. hasSpaceComplexity (Жад күрделілігі бар). Алгоритмнің жұмыс істеуі үшін қажет болатын жад көлемін сипаттайды (*Algorithm → string*). Уақыт күрделілігімен қатар жадының тиімді пайдаланылуы да маңызды: кейбір алгоритмдер қосымша жадты қажет етпесе, басқалары үлкен көлемдегі жады қолданады.

4.hasAuthor (Авторы бар). Алгоритмді алғаш ұсынған немесе сипаттаған автордың есімін анықтайды (*Algorithm → string*). Бұл қасиет алгоритмнің ғылыми және тарихи маңызын ашып көрсетеді және оны академиялық тұрғыда сілтеме жасауға мүмкіндік береді.

5. hasYearIntroduced (Ұсынылған жылы бар**).** Алгоритм алғаш жарияланған немесе сипатталған жылды көрсетеді (*Algorithm → integer*, мысалы: 1960, 1978). Бұл мәлімет алгоритмнің даму тарихын қадағалауға және ғылыми хронологияны құруға мүмкіндік береді.

Аталған мәліметтік қасиеттер алгоритмді толық әрі дәл сипаттауға жағдай жасайды. Олар онтологияның семантикалық толықтығын арттырып, жүйені автоматты түрде сұрыптауға, іздеуге және өңдеуге ыңғайлы етеді. Сонымен қатар, бұл қасиеттер OWL форматында нақты түрде жазылып, Protégé интерфейсінде визуалды түрде бейнеленді. Қосымша кластар ретінде DataStructure, Complexity, AlgorithmType, ApplicationArea, Step ұғымдары енгізілді. Олар негізгі «Algorithm» класына семантикалық жағынан байланыстырылып, онтологияның тұтастығын қамтамасыз етті. (Сурет 3).




*Сурет 3. hasAuthor мәліметтік қасиетінің сипаттамасы*

Алгоритмдік онтологияны құру барысында 6 ішкі класс, 5 объектілік қасиет және 5 мәліметтік қасиет анықталды. Бұл құрылым OWL (Web Ontology Language) стандарты талаптарына толық сәйкес келеді және Protégé онтология редакторы арқылы визуалды түрде өңделіп, тексерілді. Онтология құрамына енгізілген ішкі кластар: SortingAlgorithm, SearchingAlgorithm, GraphAlgorithm, OptimizationAlgorithm, CryptographicAlgorithm, MachineLearningAlgorithm. Object properties арқылы алгоритмдер мен басқа ұғымдар арасындағы семантикалық байланыстар көрсетілді, мысалы: hasComplexity, hasDataStructure, hasApplication, usesAlgorithmType, hasStep. Data properties алгоритмдердің нақты сипаттамаларын көрсетуге мүмкіндік берді: hasName, hasTimeComplexity, hasSpaceComplexity, hasAuthor, hasYearIntroduced.

Қосымша кластар ретінде DataStructure, Complexity, AlgorithmType, ApplicationArea, Step ұғымдары енгізілді. Олар негізгі «Algorithm» класына семантикалық тұрғыдан байланыстырылып, онтологияның тұтастығын қамтамасыз етті.

Мысал ретінде онтологияға енгізілген MergeSort алгоритмі SortingAlgorithm класына жатады және келесі сипаттамалармен толықты:

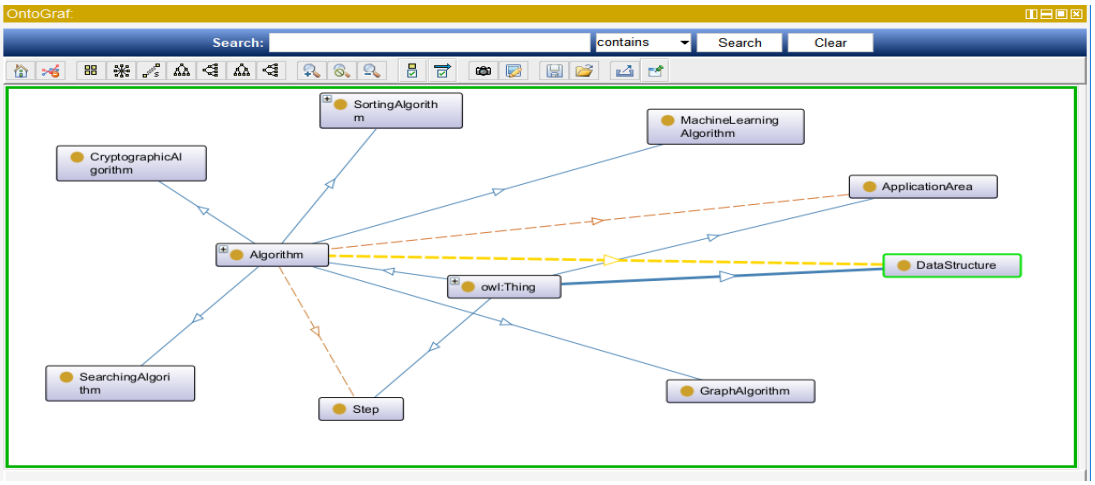
- hasName: *"MergeSort"*
- hasTimeComplexity: *"O(n log n)"*
- hasSpaceComplexity: *"O(n)"*
- usesAlgorithmType: *"DivideAndConquer"*

- hasDataStructure: *"Array"*

MergeSort алгоритмінің уақыт күрделілігі $T(n) = O(n \log n)$ (1)

Сонымен қатар, әрбір ішкі класқа нақты индивидтер қосылды: мысалы, QuickSort – SortingAlgorithm класына, BinarySearch – SearchingAlgorithm класына енгізілді.

Соңғы кезеңде онтология RDF және SPARQL құралдары арқылы тексеріліп, Protégé интерфейсінде визуалды түрде бейнеленді. Верификация нәтижесінде логикалық қателіктер анықталмай, онтологияның дұрыстығы расталды (Сурет 4).




*Сурет 4. Algorithm класы, объектілік және мәліметтік қасиеттер арасындағы байланыс*

**Зерттеу нәтижелері**

Зерттеу барысында алгоритмдік білімді онтология түрінде модельдеу нәтижесінде «Algorithm» атты негізгі класс құрылды. Оның негізінде алты ішкі класс анықталды: *SortingAlgorithm, SearchingAlgorithm, GraphAlgorithm, OptimizationAlgorithm, CryptographicAlgorithm, MachineLearningAlgorithm*. Бұл ішкі кластар нақты есептерді шешуге бағытталған алгоритмдер топтарын қамтып, пәндік саланы толық сипаттауға мүмкіндік берді.

Сонымен қатар, онтологияда бес объектілік қасиет (*hasComplexity, hasDataStructure, hasApplication, usesAlgorithmType, hasStep*) және бес мәліметтік қасиет (*hasName, hasTimeComplexity, hasSpaceComplexity, hasAuthor, hasYearIntroduced*) енгізілді. Бұл қасиеттер алгоритмдердің құрылымдық сипаттамаларын, қолданылу салаларын және күрделілік деңгейін формалды түрде көрсетуге жағдай жасады.

Мысал ретінде MergeSort алгоритмі *SortingAlgorithm* класына енгізіліп, оның негізгі сипаттамалары айқындалды: атауы (*MergeSort*), уақыт күрделілігі (*O(n log n)*), жад күрделілігі (*O(n)*), қолданған деректер құрылымы (*Array*) және қолданған әдіснамалық түрі (*Divide and Conquer*). Мұндай сипаттамалар әрбір алгоритмді нақты әрі жүйелі түрде формализациялауға мүмкіндік берді. Онтология RDF және SPARQL құралдары арқылы тексеріліп, Protégé редакторының визуалды интерфейсінде бейнеленді. Верификация нәтижесінде логикалық қателер анықталмай, онтологияның дұрыстығы расталды. Бұл нәтиже жасалған модельдің теориялық тұрғыдан дұрыс және практикалық тұрғыдан қолдануға дайын екенін көрсетті.

**Дискуссия**

Зерттеу нәтижелері алгоритмдік білімді онтология негізінде құрылымдаудың тиімділігін көрсетті. Құрылған онтология OWL стандартына сәйкес келіп, алгоритмдердің түрлері, қасиеттері мен қолдану салаларын формалды түрде сипаттауға мүмкіндік берді.

Бұл тұрғыда біздің нәтижелерімізді басқа ғалымдардың еңбектерімен салыстыруға болады. T. R. Gruber онтологияны білімді ұсынудың әмбебап тәсілі ретінде сипаттаса, N. F. Noy мен D. L. McGuinness онтология жасаудың жүйелі әдістемесін ұсынды. Алынған модель осы қағидаттардың тәжірибелік қолданылуын растайды. Сонымен қатар, онтологияларды құрудың табысты факторларын талдаған C. Feilmayr және W. Wöß еңбегі біздің құрылымдық шешімдерімізбен сәйкес келеді.

Жасанды интеллект саласындағы пайымдау әдістерін зерттеген P. Hohenecker және T. Lukasiewicz жұмыстары онтологияны терең нейрондық желілермен ықпалдастыру мүмкіндігін көрсетті. Бұл біздің нәтижелердің заманауи ғылыми үрдістермен үйлесетінін дәлелдейді. Сонымен бірге, білімді формалды сипаттау мен оны автоматтандырудың қажеттілігін көрсеткен C. M. de Rodrigues Oliveira, F. L. және әріптестері еңбектері онтологияны практикалық тұрғыдан қолданудың маңыздылығын қуаттайды.

**Қорытынды**

Құрылып отырған онтологияда информатикада кең қолданылатын және қолданыстан тәуелсіз сипатта болатын *«алгоритм»* термині негізгі ұғым ретінде алынды. Бұл ұғым онтологияда кластар иерархиясы арқылы бейнеленіп, әрбір ішкі класс белгілі бір деңгейдегі ұғымды немесе объектілер тобын сипаттады. Әр класс басқа кластармен семантикалық байланыстар арқылы жалғасып, бағытталған граф құрылымында, яғни семантикалық желі ретінде ұсынылды.

Зерттеу нәтижесінде алгоритмдік білімді жүйелеу мен формалдаудың тиімді тәсілі ретінде онтология құрудың мүмкіндігі дәлелденді. Ұсынылған онтология арқылы алгоритмдердің түрлері, қасиеттері мен қолдану салалары нақты анықталып, OWL форматына сай құрылды. Бұл ғылыми жаңалық алгоритмдік білімді формалды сипаттаудың жаңа деңгейін көрсетіп отыр. Практикалық тұрғыдан алғанда, онтология білім беру процесінде студенттердің алгоритмдік ұғымдарды меңгеруін жеңілдетуге, сондай-ақ жасанды интеллект пен семантикалық вебке негізделген ақпараттық жүйелерде тиімділікті арттыруға мүмкіндік береді. Сонымен бірге онтологияны құру барысында барлық алгоритмдерді толық қамту мүмкін болмағаны және модельдеудің күрделі тестілеуді талап ететіні шектеу ретінде көрінді. Алдағы уақытта онтологияны кеңейтіп, оны нақты пәндік салаларға (деректер ғылымы, машиналық оқыту, киберқауіпсіздік) интеграциялау зерттеудің келесі қадамы болмақ.

*Пайдаланылған дереккөздердің тізімі*

*[1] Studer, R., Benjamins, V. R., Fensel, D. (1998). Knowledge engineering: Principles and methods.* Data *& Knowledge Engineering, 25(1–2), 161–197. https://doi.org/10.1016/S0169-023X(97)00056-6*

*[2] Gruber, T. R. (1993). A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2), 199–220. https://doi.org/10.1006/knac.1993.1008*

*[3] Noy, N. F., McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory. https://protege.stanford.edu/publications/ontology_development/ontology101.pdfGol'dshtein, S. L.,*

*[4] Samaridi, N., Papakitsos, E., Karanikolas, N. (2024). Ontological Representation of the Structure and Vocabulary of Modern Greek on the Protégé Platform. Computation, 12(12), 249. https://doi.org/10.3390/computation12120249*

*[5] Feilmayr, C., Wöß, W. (2016). An analysis of ontologies and their success factors for application to business. Data & Knowledge Engineering, 101, 1–23. https://doi.org/10.1016/j.datak.2015.11.003*

*[6] Protege Documentation. Stanford University. https://protege.stanford.edu*

*[7] Hohenecker, P., Lukasiewicz, T. (2018). Ontology reasoning with deep neural networks. Journal of Artificial Intelligence Research, 62, 569–627. https://doi.org/10.1613/jair.1.11232*

*[8] Құрманғали, Р. С. Семантикалық веб және онтологияның интеграциясы // Қазақ ғылыми-зерттеу журналы. – 2023. – №1(5). – Б. 60–66.*

[9] Gao, W., Chen, Y. Approximation analysis of ontology learning algorithm in linear combination setting // Journal of Cloud Computing: Advances, Systems and Applications. – 2020. – DOI: https://doi.org/10.1186/s13677-020-00173-y

[10] Jepsen, T. C. (2009). Just What Is an Ontology, Anyway? IT Professional, 11(5), 22–27. https://doi.org/10.1109/MITP.2009.123

[11] Oliveira, de Rodrigues C. M., Gonçalves de Freitas F. L., Spósito Barreiros E. F., et al. (2019). Legal ontologies over time: A systematic mapping study. Expert Systems with Applications, 130, 12–30. https://doi.org/10.1016/j.eswa.2019.05.005

References

[1] Studer, R., Benjamins, V. R., Fensel, D. (1998). Knowledge engineering: Principles and methods. Data & Knowledge Engineering, 25(1–2), 161–197. https://doi.org/10.1016/S0169-023X(97)00056-6

[2] Gruber, T. R. (1993). A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2), 199–220. https://doi.org/10.1006/knac.1993.1008

[3] Noy, N. F., McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory. https://protege.stanford.edu/publications/ontology_development/ontology101.pdfGol'dshtein, S. L.,

[4] Samaridi, N., Papakitsos, E., Karanikolas, N. (2024). Ontological Representation of the Structure and Vocabulary of Modern Greek on the Protégé Platform. Computation, 12(12), 249. https://doi.org/10.3390/computation12120249

[5] Feilmayr, C., Wöß, W. (2016). An analysis of ontologies and their success factors for application to business. Data & Knowledge Engineering, 101, 1–23. https://doi.org/10.1016/j.datak.2015.11.003

[6] Protege Documentation. Stanford University. https://protege.stanford.edu

[7] Hohenecker, P., Lukasiewicz, T. (2018). Ontology reasoning with deep neural networks. Journal of Artificial Intelligence Research, 62, 569–627. https://doi.org/10.1613/jair.1.11232

[8] Kurmangali, R. S. (2023). Semanticalyq veb zhane ontologiyanin integratsiyasy [Semantic web and ontology integration]. Kazak gylymi-zertteu zhurnaly, 1(5), 60–66. (In Russian)

[9] Gao, W., Chen, Y. Approximation analysis of ontology learning algorithm in linear combination setting // Journal of Cloud Computing: Advances, Systems and Applications. – 2020. – DOI: https://doi.org/10.1186/s13677-020-00173-y

[10] Jepsen, T. C. (2009). Just What Is an Ontology, Anyway? IT Professional, 11(5), 22–27. https://doi.org/10.1109/MITP.2009.123

[11] Oliveira, de Rodrigues C. M., Gonçalves de Freitas F. L., Spósito Barreiros E. F., et al. (2019). Legal ontologies over time: A systematic mapping study. Expert Systems with Applications, 130, 12–30. https://doi.org/10.1016/j.eswa.2019.05.005