Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

УДК 004.852
DOI: https://doi.org/10.30837/ITSSI.2024.27.039

З. Дудар, С. Літвін

МЕТОД ОНТОЛОГІЧНОГО ОПИСУ В ПОБУДОВІ СЕРВІС-ОРІЄНТОВАНИХ СИСТЕМ РОЗПОДІЛЕНОГО НАВЧАННЯ

Предметом дослідження є аналіз та обґрунтування використання процесів програмної інженерії, основаних на онтологіях доступу до інформації та надання доступу до бази знань і їх повторного використання мовою алгебри скінченних предикатів, що є класом дескриптивних логік. Такий підхід дає змогу застосовувати переваги онтологій і логічного програмування в процесі рефакторингу розподілених систем дистанційного навчання. **Мета роботи** – створення формалізму опису взаємодії сервісів і алгоритмів побудови інтерфейсів вебсервісів для реалізації ефективної *SOA*-системи із застосуванням парадигми основаного на онтологіях доступу до інформації. Для досягнення поставленої мети визначено такі **завдання**: аналіз основних видів структур взаємодії *SOA*-сервісів; огляд реалізацій систем, що працюють із сімейством мов дескриптивної логіки, а саме алгебри скінченних предикатів; опис структури онтології, яка необхідна для роботи сервіс-орієнтованих систем, що має розмітку стандарту *Semantic Web* за допомогою рівнянь алгебри скінченних предикатів. Актуальним завданням є забезпечення організації взаємодії інтернет-ресурсів у розробленні систем розподіленого віртуального навчання, і таку взаємодію уможливлює сервіс-орієнтована архітектура створення програмного забезпечення. Стаття розглядає сучасні технології *Semantic Web* та їх роль у створенні прикладних програмних систем розподілених навчальних ресурсів, побудованих за допомогою розподілених взаємодіючих сервісів. Розглянуті **методи** та матеріали: алгебра скінченних предикатів, теорія алгоритмів, об'єктно-орієнтоване проєктування, теорія уніфікації. **Результати**. Отримано семантику, що дає змогу описувати *SOA*-системи в мовах дескриптивної логіки. Показано ефективність математичного формалізму алгебри скінченних предикатів для завдань логічного аналізу прикладних онтологій, упроваджено методи повторного використання знань і опису сервіс-орієнтованих систем. Доведено необхідність застосування алгоритмів автоматичної побудови вебінтерфейсів. **Висновки**. У роботі висвітлено переваги дослідженого рішення, а саме, алгоритми автоматизованої побудови інтерфейсів вебсервісів для *SOA*-архітектури, що дають змогу ефективно виконувати поставлене завдання. Алгоритми вирізняються більш загальною моделлю та швидкістю роботи й потребують оцінювання складності.

Ключові слова: програмна інженерія; сервіс-орієнтована архітектура; алгебра скінченних предикатів; онтології; алгебраїчне програмування; розподілене віртуальне навчальне середовище.

Вступ

Зростання популярності за останні роки таких технологій, як хмарні обчислення, семантичні мережі та *Semantic Web*, демонструє актуальність завдання організації взаємодії інтернет-ресурсів між собою. Забезпечити таку взаємодію дає змогу сервіс-орієнтована архітектура (*SOA*) розроблення програмного забезпечення. *SOA*-системи легко масштабуються, а їх сервісні структури полегшують повторне використання компонентів. *SOA*-архітектура забезпечує створення прикладних програмних систем, побудованих за допомогою розподілених взаємодіючих сервісів. Нині існує безліч систем для створення вебзастосунків із *SOA*-архітектурою: *Microsoft .NET, Oracle SOA Suite*. Однак відсутні ефективні системи, здатні автоматично будувати інтерфейси та перетворювати популярні сайти минулого покоління в елементи композитних служб або сервісних кластерів.

Як гілка символічного штучного інтелекту, подання знань та логічне виведення спрямовані на проєктування програмного забезпечення систем, що інтерпретують подання світу, схоже на сприйняття людини. Системи, що належать до класу навчальних розподілених віртуальних систем, також основані на знаннях, мають обчислювальну модель предметної галузі, де символи відіграють роль артефактів реального світу, таких як фізичні об'єкти, події та відносини. Предметна галузь може охоплювати будь-яку частину реального світу або гіпотетичну систему, щодо якої є бажання подати та інтерпретувати знання для цілей проєктування відповідних програмних систем.

Названі методи з використанням апарату штучного інтелекту створюють основу для побудови інтелектуальних адаптивних гіпермедійних систем та значно підвищує якість процесу інтенсифікації дистанційного навчання, відчутно збільшуючи його можливості. Ці системи підтримують модель

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)

користувача й застосовують її для адаптації гіпермедійного простору та методів навчання до своїх потреб. Кожен користувач має свої траєкторії навчання та індивідуальні навігаційні можливості для роботи з гіпермедійним простором за допомогою *SOA*-архітектури навчальних ресурсів і розподілених застосунків. Необхідність алгебро-логічного опису таких систем зумовлена проблемами реалізації *EAI* (*Enterprise Application Integration* – інтеграції корпоративних застосунків) – інтеграційної архітектури, що містить набір технологій та методів для інтеграції систем і застосунків у масштабах корпоративних архітектур.

Зазначені застосунки без створення додаткових програмних рішень, як правило, не можуть взаємодіяти один з одним для обміну інформацією або за правилами бізнес-процесів (вироблення індивідуальної траєкторії навчання). Отже, з'являються проблеми з автоматизацією процесів і надлишковою інформацією, що зберігається в декількох місцях. Інтеграція програмних засобів і застосунків навчального призначення – засіб, що забезпечує процес зв'язування таких застосунків у межах однієї парадигми для одночасного їх використання з метою спрощення й автоматизації бізнес-процесів. Процес зв'язування має протікати таким чином, щоб не вносити радикальних змін у наявні застосунки або структури даних. Унаслідок цих факторів побудова *SOA*-системи з композитною структурою зазвичай зводиться до методів інтеграції корпоративних застосунків.

Будувати ефективну *SOA*-систему, позбавлену або частково позбавлену питань реалізації *EAI*-систем, доцільно за допомогою застосування парадигми доступу до інформації, що основано на онтологіях (*OBDA – Ontology-Based Data Access*) – доступ до даних з допомогою високорівневого інтерфейсу онтології. Центральним процесом, що ґрунтується на онтологіях доступу до інформації, є подання бази даних мовою дескриптивної логіки (*DL*). Унаслідок застосування *OBDA*-технологій спростилася робота щодо зміни структури *SOA*-системи порівняно з провідними *SOA*-системами, зазначеними раніше. Для зміни структури *SOA*-системи досить відредагувати онтологію системи (що не складно для людини без фахової освіти) за допомогою редактора онтологій. У цьому разі відкривається можливість побудови інтерфейсів на базі семантичної сервіс-орієнтованої архітектури (*SSOA*). Використання та розвиток таких технологій безпосередньо орієнтовані на розв'язання актуального завдання – створення розподіленого віртуального навчального середовища та компонентів, пов'язаних із повторним застосуванням знань, структурним перетворенням навчальних елементів та описом індивідуальних навчальних траєкторій.

Аналіз останніх досліджень і публікацій

Об'єктом дослідження є *SOA*-системи, а також їх оптимізація за допомогою парадигми, основаної на онтологіях доступу до інформації; перетворення онтологічних описів і моделей інтелектуальних систем за допомогою алгебри скінченних предикатів, моделей, що описують алгебро-логічні перетворення навчальних матеріалів.

Далі наведено три основні структури, що характеризують *SOA*-системи за внутрішнім складом і способами взаємодії з обчислювальним середовищем [2, 3]:

− атомарна служба (*atomic service*) – неподільна й проста за складом;

− композитна служба (*composite service*) – батьківська, зі складною будовою, що охоплює одну або більш прості за складом дочірні служби, сховані від зовнішнього світу. Ця формація може бути подана у вигляді ієрархічної структури;

− сервісний кластер (*service cluster*) – група розподілених за призначенням і технологічно пов'язаних служб, що взаємодіють для розв'язання поставленого завдання.

Атомарні та композитні служби ефективно працюють в умовах однорідного обчислювального середовища (*Homogeneous Computing Environment*). Однорідне обчислювальне середовище належить до категорії прикладних архітектур (*Application Architecture*). Водночас композитні служби й сервісні кластери часто наявні в гетерогенних обчислювальних середовищах (*Heterogeneous Computing Environment*) під впливом завдань корпоративних архітектур (*Enterprise Architecture*). Отже, композитні служби відіграють важливу роль в обох обчислювальних середовищах.

SOA-системи легко масштабуються, а їх сервісні структури полегшують повторне використання компонентів. *SOA*-архітектура забезпечує створення прикладних програмних систем, побудованих за допомогою розподілених взаємодіючих сервісів [4].

Зі зростанням обсягу інформації в мережі Internet потрібно спростити доступ до інформації

ISSN 2522-9818 (print)
Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27) ISSN 2524-2296 (online)

вебсервісів для подальшої автоматизації процесів структурування та подання інформації. Такий інтерфейс можна забезпечити з допомогою подання знань про предметну галузь у формі онтології, доступної як інструмент технологій *Semantic Web* – перспективної галузі у сфері розвитку Internet [5]. Основне завдання *Semantic Web* – подання інформації у вигляді, придатному для машинного оброблення. Програмний агент може безпосередньо отримувати з павутини факти й виводити з них логічні висновки. Основними інструментами семантичної павутини є семантичні мережі, універсальні ідентифікатори ресурсів (*URI – Uniform Resource Identifiers*) і онтології [6]. Структура *Semantic Web* має множину мов опису, схвалених консорціумом *W3C*. Ця множина, крім іншого, містить *URI, XML, RDF, RDF-S, OWL* [7].

URI – універсальний ідентифікатор ресурсів, що зазвичай використовується для посилання на об'єкт за допомогою протоколу HTTP у мережі Internet. Водночас у *Semantic Web* універсальний ідентифікатор ресурсів використовується для позначення об'єктів, зокрема й об'єктів матеріального світу, таких як країни, компанії, люди та інші, або абстрактних об'єктів: назва компанії, ім'я, вік. Унаслідок того, що *URI* глобально унікальні, вони дають змогу ідентифікувати ті самі об'єкти в різних місцях середовища *Semantic Web* [8].

Структура XML-документа визначається за допомогою специфікації *XML Schema*. Завдяки схемі документа, сформованій цією специфікацією, можна проаналізувати будь-який XML-документ на правильність його структури [9].

RDF – стандарт, що описує графи, у яких дуги й вузли містять *URI*. Формат опису RDF-даних має вигляд триплета "суб'єкт – відношення – об'єкт" (рис. 1). Елементами RDF-триплета є *URI* [10, 11].

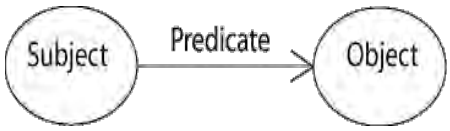

Рис. 1. Схема формату "суб'єкт – відношення – об'єкт"

Консорціум *W3C* створив XML-схему для опису RDF-документів. Водночас для RDF існує аналог *XML Schema – RDF-S.*

RDF-графи можна використовувати для роботи з онтологіями [12], описаними за допомогою *RDF-S* і *OWL*. Цей підхід забезпечить можливість отримувати з RDF-документів логічні висновки. Унаслідок цього можна використовувати стандарти *Semantic Web* для взаємодії баз знань і вебсервісів.

Визначення не вирішених раніше частин загальної проблеми. Мета роботи, завдання

За допомогою онтологій можна створювати загальні та локальні твердження. Розрізняють загальні та прикладні онтології. Загальні, або базисні, онтології містять універсальні для більшості предметних галузей поняття. Такий тип онтологій передбачає наявність базового набору термінів. Ці терміни є правилами для наступного опису більш конкретних об'єктів предметних галузей. Існують практичні реалізації технологій уніфікації онтологій, де використовуються загальні онтології [13]. Прикладні онтології містять знання про об'єкти якої-небудь конкретної предметної галузі. Такі онтології є більш модифіковані щодо загальних онтологій. Прикладні онтології оперують термінологією, властивими для певної царини термінами, що не беруть до уваги омоніми з інших предметних галузей. Прикладом омонімів може бути слово "провідник". У фізиці ця лексема вирізняється за значенням від того самого терміна транспортної галузі.

Мовою опису онтологій використовуються дескриптивні логіки (*Description Logics, DL*) [14]. Вони містять багаті виразні можливості логіки предикатів і прийнятні обчислювальні властивості: здатність розв'язання та невисоку обчислювальну складність завдань логічного аналізу.

Ці фактори забезпечують можливість застосовувати *DL* на практиці для роботи із *SOA*-системами. Дескриптивні логіки є розв'язними, але менш виразними фрагментами логіки предикатів. За синтаксисом *DL* схожі на модальні логіки. Мінімальними вимогами до виразної сили мови для побудови моделі *SOA*-системи є опис концептуальної моделі даних у базах даних і складників їх відношень.

Дескриптивні логіки створювалися для розширення семантичних мереж і фреймових структур механізмами формальної логіки. Історично *DL* називалися "термінологічні системи", але в 1980-х отримали назву "дескриптивні логіки". Вони оперують поняттями онтологій, такими як "ролі" та "концепти". В інших розділах математичної логіки термін "концепт" відповідає поняттю "одномісний предикат". Термін "роль" відповідає поняттю "двомісний предикат", або "бінарне відношення" [15].

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)

У поняттях *DL*-структури, описаних у термінах алгебри скінченних предикатів, мають бути визначені таким чином:

− термінологія, що здатна описати предметну галузь або концептуальну модель даних, наприклад модель "сутність – зв'язок" (*Entity-Relationship, ER*), *ER*-модель даних використовується для проєктування реляційних баз даних або *Tbox* – набору термінів загального виду;

− набір тверджень про індивідів приватного виду, або *Abox*.

Отже, можна зробити взаємо однозначний перехід між аксіоматикою дескриптивних логік і аксіоматикою алгебри скінченних предикатів та її різновидів. Розглянемо кілька типів аксіом на прикладі синтаксису дескриптивної логіки *ALC*.

Формалізм для опису й аналізу концептуальної моделі *SOA*-системи на моделі алгебри скінченних предикатів, який потрібно орієнтувати на онтологічний підхід, дозволяє описувати взаємодію вебсервісів та нові алгоритми автоматизованої побудови інтерфейсів вебсервісів для *SOA*-архітектури. Такий підхід показує можливість автоматизовано будувати інтерфейси вебсервісів та використовувати весь накопичений описовий апарат алгебри скінченних предикатів як клас дескриптивних алгебр.

Отже, припустимо, A і B – деякі концепти або ролі в термінах онтологічного опису, тоді аксіома вкладеності – це вираз виду $A \subseteq B$. Цей вираз буквально означає, що концепт або роль A додані в концепт або роль B. Аксіома еквівалентності – це вираз виду $A \equiv B$, еквівалентність можна подати як дві аксіоми $A \subseteq B$ і $B \subseteq A$. Аксіома кон'юнктивності – це вираз виду $A \cap B$, що відповідає за операцію перетинання з теорії множин та повністю відповідає аксіоматиці алгебро-логічного подання задач побудови розподілених віртуальних навчальних середовищ [16].

Завданнями роботи є проведення аналізу основних видів структур взаємодії *SOA*-сервісів, огляд експериментальних реалізацій систем *OBDA*, що працюють із сімейством мов дескриптивної логіки, та опис структури онтології, що необхідна для роботи *SOA*-системи, яка має розмітку стандарту *Semantic Web* за допомогою рівнянь алгебри скінченних предикатів. Необхідним завданням є проєктування алгоритмів автоматизованої побудови інтерфейсів вебсервісів для *SOA*-архітектури в процесі розроблення та рефакторингу віртуальних розподілених навчальних систем, що дають змогу автоматизовано будувати інтерфейси вебсервісів на основі онтології. Важливо також підтвердити практичне застосування класу дескриптивної логіки та мов вебонтологій з метою надання онтологічного доступу до інформації. Одним із завдань дослідження є обґрунтування теоретичної складності та завершуваності ухвалених рішень.

Мета роботи – створення формалізму опису взаємодії сервісів і алгоритмів побудови інтерфейсів вебсервісів для реалізації ефективної *SOA*-системи із застосуванням парадигми основаного на онтологіях доступу до інформації. Програмна реалізація *SOA*-системи призначена для використання в основі систем автоматизованої побудови інтерфейсів взаємодії вебсервісів.

Матеріали й методи

Перевага *SOA*-технологій полягає в багаторазовому використанні програмного коду для створення складених розподілених програмних систем. Сервіс-орієнтована архітектура забезпечує кросплатформеність і незалежність від засобів розроблення, дозволяючи легко управляти створюваними програмними системами та інтегрувати їх. Наприклад, програмні реалізації компонентів *SOA*-системи, написані мовою програмування *Java*, можуть бути викликані компонентами, написаними мовою *C#* у межах однієї або декількох *SOA*-систем [2]. Основною перевагою *SOA*-архітектури є здатність до масштабованості – керування зростанням корпоративних систем, можливість надання та використання послуг у масштабах мережі Internet, здатність до скорочення витрат на організацію взаємодії систем. Нижче наведений приклад проєкту, для якого підходить *SOA*-архітектура. Великій компанії, що об'єднує менші організації, необхідно визначити, як інтегрувати придбані IT-інфраструктури у свій проєкт. Завдяки здатності до масштабування та інтеграції *SOA*-архітектура дає змогу проєктам адаптуватися до потреб конкретної предметної галузі або процесів, що відбуваються в межах цієї галузі. Інфраструктура *SOA* сприяє також більш гнучкій і оперативній роботі системи, саме тому цілісна система побудована на значній кількості парних інтерфейсів. Отже, *SOA*-архітектура забезпечує міцну основу для систем, що відповідають умовам бізнес-гнучкості й адаптивності.

Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

Центральним поняттям *SOA*-архітектури є "сервіс" (*service* – послуга) [17]. Іменник "послуга" тлумачиться в словниках як "виконання робіт (функцій) один для одного". Однак необхідно пам'ятати, що зазначений термін поєднує такі взаємозалежні ідеї:

- можливість виконувати роботу для клієнтів;
- чітка формалізація робіт, наданих клієнтам;
- пропозиція виконати роботу для клієнта.

Сервіси є центральним поняттям *SOA*-архітектури, оскільки реалізують концепцію масштабування завдяки ефектам прозорості та легкої інтеграції. Прозорість виражається з допомогою опису сервісу, що містить відомості, необхідні для взаємодії із сервісом, і подає це в термінах "вхідні дані служб", "вихідні дані служб" і пов'язаній з ними інформації.

Опис сервісів також передає інформацію про завдання служби та умови для використання сервісу. Ключовим компонентом сервісів є чітко детерміновані інтерфейси, які можуть бути викликані стандартними способами, навіть якщо дані про застосунок, що їх викликає, не відомі сервісу. Водночас об'єктам, що взаємодіють із сервісами, не відома інформація про механізми виконання службами їх цільового завдання. Ці ефекти досягаються з допомогою інкапсуляції деталей реалізації від інших частин системи [2].

Повторне використання знань в інформаційних системах часто описується дескриптивними логічними мовами [18]. Дескриптивний підхід насамперед належить до формалізмів логіки предикатів першого порядку. Його важливість випливає з того факту, що здебільшого всі формалізми символічного подання знань можуть бути зведені до логіки предикатів першого порядку. Логіка першого порядку відтворює сутність мислення людини з допомогою терміна "логічний висновок". Вона також дає змогу вводити поняття "універсальна правильність" у тому значенні, що логічне твердження може бути правильним незалежно від будь-яких передумов. Логічний висновок та універсальна правильність можуть описуватися в термінах моделювання семантики тверджень. Модель для логічної теорії описує умови, за яких теорія правильна. Логічний висновок є твердженням, що правильне для всіх моделей теорії. Далі дедуктивне міркування пояснює, як висновок випливає з теорії. Дедукція дає доступ до знань, які неявно подані в теорії. У термінах предикатних рівнянь [19] це означає, що розв'язання рівняння щодо невідомих значень змінних чи відношень між ними дає нам додаткові знання, не подані в явному вигляді в системі рівнянь бази знань.

Додавання рівнянь, отриманих унаслідок дедуктивного висновку з наявної бази знань, є поповненням бази знань і дає змогу повторно використати знання. Це має багато практичних переваг, зокрема: підвищення ефективності, якості та інноваційної діяльності, зменшення витрат часу й ресурсів, а також сприяння навчанню та розвитку. Проблеми повторного використання знань мають вагоме значення для різних галузей науки та практики.

Система логічних предикатних рівнянь здатна застосовувати знання предметної галузі, отримані з меншої проблеми, до більш складних проблем тієї самої або спорідненої галузі. Але наразі переважній більшості логічних та еволюційних обчислювальних методів не вистачає цієї здатності. Відсутність можливості застосувати вже набуті знання про предметну галузь призводить до споживання більшої кількості ресурсів і часу для розв'язання більш складних проблем предметної галузі. Відповідно до того, як проблема збільшується в розмірах, її стає важко, а іноді навіть непрактично (якщо не неможливо), розв'язати через нестачу ресурсів і часу. Тому необхідна система, яка має здатність повторно використовувати набуті знання про проблемну сферу для масштабування в певній галузі.

Система, основана на знаннях, підтримує базу знань, що зберігає символи обчислювальної моделі у формі тверджень щодо галузі, яка розглядається. У нашому випадку ці твердження записуються як рівняння алгебри скінченних предикатів. Така система здійснює логічні виведення з наявних рівнянь, результати яких можуть поповнювати базу знань, забезпечуючи цим повторне використання знань. Програмні застосунки можуть базувати прийняття рішень на основі запитів до бази знань.

Основна мета розроблення алгебро-логічного опису онтологічних властивостей сервіс-орієнтованих ресурсів полягає в тому, щоб створити систему, здатну автономно масштабувати навчання, починаючи від незначних питань і завершуючи складнішими проблемами тієї самої чи спорідненої сфери, у поведінці, подібній до людської. Для того, щоб автономно масштабувати проблемну сферу, необхідно вміти формувати багаторазове застосування логічних блоків знань. Для вилучення та повторного використання інформації з логічних рівнянь у проблемній сфері потрібне розширене кодування. Простір пошуку може розширитися, наприклад,

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)

у деяких формах генетичного програмування машинного навчання.

Як правило, агент містить еволюційні обчислення та машинне навчання для вирішення певного завдання в невідомому середовищі. Правила мають форму "якщо умова, то дія". Зазвичай умова подана бітовим рядком фіксованої довжини, визначеним у потрійному алфавіті $\{0,1,\#\}$, де # – символ, що означає "невизначено", тобто або 0, або 1, а дія подана числовою константою. Техніка класифікаційної системи навчання може масштабуватися в проблемних сферах, але щоразу її потрібно вивчати із самого початку. Крім того, збільшення розмірності проблеми, що призводить до розширення простору пошуку, вимагає великого простору пам'яті та призводить до значно більшого часу навчання та зрештою обмежує систему розміром проблеми [19]. За допомогою явної передачі знань домену в системі класифікації можна досягти масштабованості, але це додає упередженості та обмежує використання в кількох доменах.

Будь-яка обчислювальна система має такі обмеження: 1) алфавіт літер, з яких будуються слова для будь-якої конкретної обчислювальної системи, завжди скінченний; 2) довжина слів, які здатна сприймати та формувати ця система, обмежена деяким кінцевим, наперед заданим числом букв, що визначається конструкцією системи, її швидкодією та терміном служби; 3) реакції обчислювальної системи чітко детерміновані: повторне подання вхідного слова завжди приводить до формування системою того самого вихідного слова.

Проблема повторного використання має три основні причини. Перша стосується розуміння контексту та застосованого рішення, інша – змісту знань, що документуються, і остання стосується всієї системи бази знань або програмного забезпечення, яке використовується для її підтримки. Коли говорять про обмін і повторне застосування знань, існує припущення, що знання є такими, які можна відтворювати та переміщувати з місця на місце; сутність, що можна отримати від людини-експерта й перенести з однієї комп'ютерної системи чи програми в іншу. Але як можна запропонувати поділитися або повторно використовувати те, що не має таких властивостей, як локальність і стійкість? Адже знання – це абстракція, яку неможливо записати й потримати в руках. Знання – це те, що спостерігач пояснив би розумному агенту, що дає змогу агенту раціонально моделювати свою поведінку для досягнення певних цілей, які сприймаються, відповідно до того, що він дізнався від спостерігача.

Якщо припустити, що все вищезазначене виконано, то проблема IT-підтримки процесу управління знаннями все ще потребує вирішення. Мета поточних баз знань – сприяти обміну знаннями та повторному їх використанню. Для подання знань необхідно зважати на значну кількість факторів. Термінології, онтології та методи вирішення проблем – це декілька з них [14]. Питання полягає в тому, щоб мати змогу поділитися знаннями, які містяться в різних базах знань, оскільки всі ці фактори відрізняються від однієї бази до іншої. Несумісність систем і форматів також унеможливлює об'єднання двох баз знань.

Можна виокремити чотири основні причини.

1. Неоднорідність подання. Існує кілька підходів до подання знань, але один формалізм подання не може бути безпосередньо включений в інший та не існує універсального формалізму подання знань, який би ідеально відповідав усім вимогам, і тому обмін знаннями передбачає переклад змісту однієї бази в іншу.

2. Мовні діалекти. Обмін знаннями між системами може бути дуже складним, якщо знання були закодовані різними діалектами. Є ймовірність, що це повністю змінить зміст повідомлення або його інтерпретацію.

3. Відсутність визначених правил спілкування. Теоретично окремі системи можуть комунікувати одна з одною і в такий спосіб мати користь від знань одна одної, не маючи спільної бази. Але зазвичай це неможливо, оскільки бракує узгодженого протоколу, що дозволяв би системам взаємодіяти та запитувати одна одну.

4. Невідповідності моделі на рівні знань. Навіть у разі усунення всіх перешкод проблеми з термінологією також стануть перешкодою для ефективного спілкування між різними базами. Відсутність спільного словникового запасу не дає змоги співвідносити знання однієї бази з іншою.

Отже, знання розглядається як здатність реагувати певним чином, а не як матеріальна субстанція. Навіть інформацію, що використовується для подання знань, не можна вважати такою; правила, символи та фрейми не здатні генерувати розумну поведінку. Щоб математично описати функції інтелекту, необхідно створити формальну мову,

Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

якою можна було б здійснювати такий опис. Формальна мова має обиратися таким чином, щоб нею можна було в зручній формі записати будь-який скінченний алфавітний оператор. Такою мовою є алгебра скінченних предикатів [8].

У роботі розглядаються кон'юнктивні перетворення та відповідні формули й розширення алгебри скінченних предикатів, що відповідає кінцевій меті – побудові адекватної навчальної індивідуальної траєкторії суб'єкта навчання. Відповідь на кон'юнктивний запит $CQA(A,T,q_c)$, де q_c – кон'юнктивний запит до інформації з *Abox A* деякої КВ, що залежить від *Tbox T,* зводиться до завдання виконати запит до бази даних: $QE(A,q_0)$, де q_0 – запит у межах логіки предикатів, що не залежить від *Abox* і здатний бути реалізованим у мові реляційних баз даних, наприклад *SQL*. Проблема цього підходу полягає в тому, що запит може суттєво розширитися. З огляду на наведений приклад подамо *ER*-діаграму взаємодії сервісів в *SOA*-системі (рис. 2).

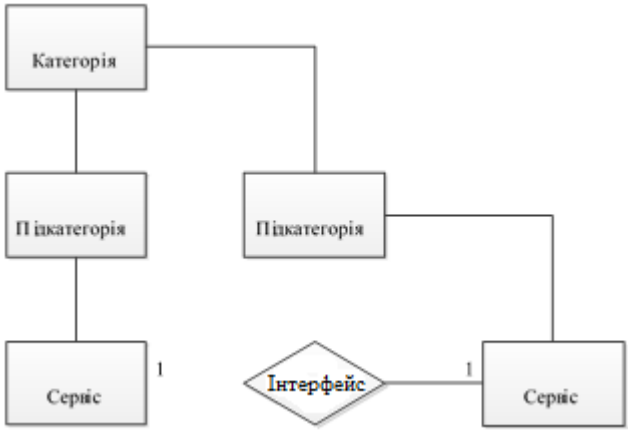

Рис. 2. Приклад *ER*-діаграми взаємодії вебсервісів

Розглянемо автоматизацію побудови вебінтерфейсів на основі вхідних параметрів для *SOA*-архітектури. Побудуємо для цього математичну формалізацію елементів сервіс-орієнтованої архітектури побудови вебінтерфейсів.

Сервіси всередині *SOA*-системи можуть належати до кількох категорій [2], певним розроблювачам, що, наприклад, розподіляють сервіси за територіальними й функціональними ознаками. Категорії зі свого боку можуть міститися в інших категоріях. Кожний сервіс може мати інтерфейс взаємодії з іншими сервісами. Отже, існує кілька визначень.

Визначення 1. Категоризацію композитних сервісів і сервісних кластерів можна подати у вигляді формалізації в межах теорії множин.

Якщо сервіси b_s, b_{s1}, b_{s2} є елементами множини B_s, певні категорії $B_s, B_{s_1}, B_{s_2}, B_{s_3}$ є множинами, що належать до універсальної множини U, інтерфейси належать множині відношень R_s, то система множин, що описує *SOA*-архітектуру, підпадає під певні обмеження.

Сервіс b_s має міститися в певній категорії $(b_s \in B_s)$.

Необхідно, щоб вкладення підкатегорії в категорію відповідало вимозі $B_{s_1} \subseteq B_{s_2}$.

Вкладення підкатегорії в кілька категорій має відповідати вимозі $Bs_1 \subseteq Bs_2 \cap Bs_3$.

Вкладення декількох підкатегорій в одну категорію має відповідати вимозі $Bs_1 \cup Bs_2 \subseteq Bs_3$.

Категорії мають відповідати вимозі $B_{s_1} \cup B_{s_2} = \varnothing$.

Інтерфейс має відповідати вимозі $(b_{s_1},\ b_{s_2}) \in R_s$.

Тоді автоматизована побудова інтерфейсів має вигляд такої відповідності – відповідність φ є відображенням області визначення $Servis(C_s, Q_s, S_s)$ в область значень:

$$\cup ServI(c_i, q_i, s_i),$$

$$s_{SOA} \in I_{SOA},$$

де C_s, Q_s, S_s – параметри для множинного створення сервісів; I_{SOA} – елемент множини *ISOA*; *ISOA* – множина інтерфейсів, які відповідають зазначеним параметрам.

Визначення 2. Значення для області визначення відповідності називаються вхідними параметрами із залежними умовами, якщо $x \in x'_C$, $x \in x'_s$, де x – деякий елемент умови запиту. В іншому разі значення для відповідності називаються вхідними параметрами без залежних умов.

Наведемо приклад для параметрів без залежних умов. Потрібно побудувати інтерфейси (Клієнт *Cs*, Дані *Qs*, Сервер *Ss*), де *Cs* – сайт навчального порталу (*Educationportal*) міста Харків (*Khcity*). У вебсервісі C_s із вебсервісів S_s будуть відправлятися дані Q_s, що містять інформацію про номери телефонів установ (*Phone*). Вебсервіси S_s зі свого боку –

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)

це утворювальні сайти навчальних закладів (*Education*) міста (*Khcity*).

$$C_s(x_1, x_2) \leftarrow Educationportal(x_1), Khcity(x_2);$$

$$Q_s(x_3) \leftarrow Phone(x_3);$$

$$S_s(x_4, x_5) \leftarrow Education(x_4), Area(x_5).$$

Далі наведений приклад для параметрів із залежними умовами. Запит адреси сервера та запити адреси клієнта можуть містити однакові елементи умови, такі параметри називаємо параметрами із залежними умовами.

Потрібно побудувати інтерфейси (Клієнт C_s, Дані Q_s, Сервер S_s), де на всіх сайтах навчальних порталів (*Educationportal*) кожного міста (*City*) для адміністративного регіону (*Region*) подаватиметься інформація про персонал (*Personal*) з вебсервісу S_s. Вебсервіси S_s – це навчальні сайти міста (*City*), зазначеного в C_s. Залежні умови позначаються однаковими змінними елементів умови запиту. У цьому прикладі така змінна – x_3.

$$C_s(x_1, x_2, x_3) \leftarrow Educationportal(x_1), Region(x_2), City(x_3);$$

$$Q_s(x_4) \leftarrow Personal(x_4);$$

$$S_s(x_5, x_3) \leftarrow Education(x_5), City(x_3).$$

Далі подана оптимізація множини *ANS* для відповідності *Servirend*. Ролі в онтологічній *SOA*-моделі використовуються для реєстрації інформації про інтерфейси, що додаються. Визначимо безліч аксіом і тверджень бази знань *K*, що містять ролі, як *Rbox*. У процесі оброблення вхідних параметрів відповідності *Servirend* інформація про ролі не потрібна. Отже, немає необхідності брати до уваги *Rbox* для реалізації подання *Servirend*.

Розглянемо інтерпретацію відповіді на запит до бази даних без урахування *Rbox*.

Нехай запит q – кон'юнктивний запит або об'єднання кон'юнктивних запитів, а база знань $ДО_c = K/Rbox$, тоді відповіддю на q до $ДО_c$ є множина $servans(q, K_c)$ кортежів $\vec{a}$ із констант, що містяться в K_c, таких, що $\vec{a}^{M_c} \in q^{M_c}$ для кожної моделі M_c у K_c.

Відсутність *Rbox* зменшує розмір *Tbox* і *Abox,* що частково вирішує проблему розширення запиту під час відповіді на *OBDA*-запит.

Результати досліджень та їх обговорення

Далі наведемо приклад опису *ER*-моделі обмеження доступу користувачів до тем або інших видів послуг вебсервісів засобами логіки (рис. 3).

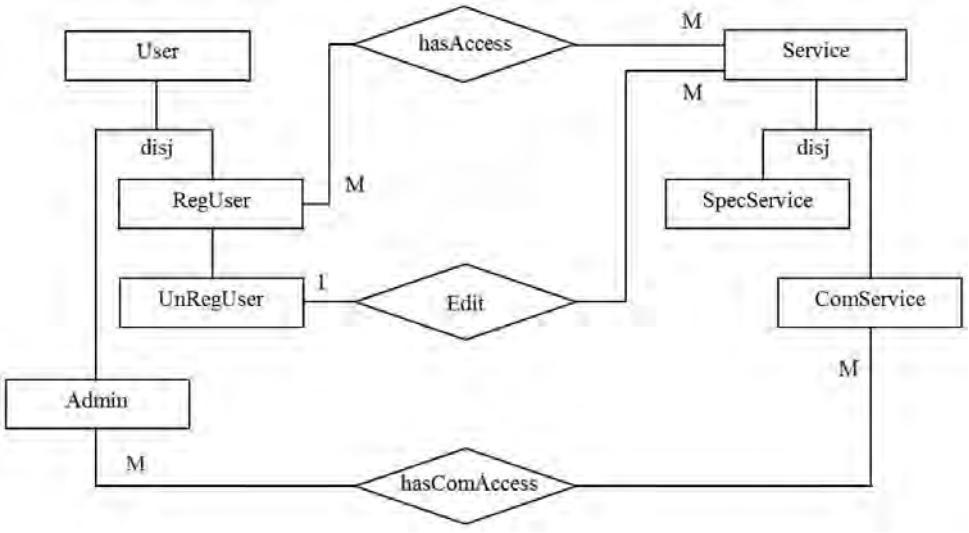

Рис. 3. *ER*-діаграма доступу до послуг вебсервісу

ISSN 2522-9818 (print)
Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27) ISSN 2524-2296 (online)

У розглянутій *ER*-моделі користувачі поділяються на зареєстрованих (*Reguser*) і незареєстрованих (*Unreguser*). Розрізняють послуги (*Service*) спеціальні (*Specservise*) і загальні (*Comservice*). Незареєстровані користувачі мають доступ до загальних послуг (*hascomaccess*). Зареєстровані користувачі мають доступ до всіх послуг (*hasaccess*), зокрема й до спеціальних. Зареєстрований користувач може бути адміністратором (*Admin*), що редагує (*Edit*) послугу.

Підсутності *ER*-моделі описуються за допомогою включення концептів.

$Reguser \in User$, $Unreguser \subseteq User$, $Admin \subseteq Reguser$,

$Comservice \subseteq Service$, $Specservice \subseteq Service$.

Аналогічно описуються підзв'язки за допомогою включення ролей:

$\exists hasComAcces \subseteq \exists hasAccess$.

Якщо використовувати кон'юнктивний складник модифікацій алгебро-логічного опису, то немає можливості повністю відтворити диз'юнкцію. Проте можна описати диз'юнкцію концептів і ролей з допомогою негативного включення:

$RegUser \subseteq \neg UnRegUser$,

$ComService \subseteq \neg SpecService$.

Для зазначення зв'язку між двома сутностями, що беруть участь у відношенні, використовуються еквівалентність ролі з концептом, що виражає першу сутність, і еквівалентність інверсії цієї ролі з концептом, що виражає другу сутність. Еквівалентність реалізується за допомогою взаємної імплікації двох сигнатур.

$UnregUser \subseteq \exists hasComAccess$,

$RegUser \subseteq \exists hasAccess$,

$Admin \subseteq \exists edit$,

$\exists hasComAccess \subseteq UnRegAccess$,

$\exists hasAccess \subseteq RegUser$,

$\exists edit \subseteq Admin$,

$Service \subseteq \exists has\_Access-$,

$\exists has\_Access- \subseteq Service$, $Service \subseteq \exists edit-$,

$\exists edit- \subseteq Service$,

$ComServise \subseteq \exists hasComAcess-$,

$\exists hasComAccess- \subseteq ComService$.

Для створення алгоритмів логічного аналізу онтологічної *SOA*-системи потрібно розглянути основні алгоритмічні завдання, пов'язані з категорійним аналізом алгебри скінченних предикатів як дескриптивної логіки та парадигмою *OBDA*.

Перелічимо визначення цих завдань.

Крок 1 – визначення несуперечності бази знань – перевірка на існування хоча б однієї моделі в KB $K =< Tbox, Abox >$.

Крок 2 – перевірка на існування $B^{I \subseteq CI}$ у довільній інтерпретації I для заданого *Tbox*, де C, B – концепти або ролі, тобто з'ясування властивості "вкладеність концептів або ролей".

Крок 3 – "перевірка екземпляра класу" – перевірка на існування $a^I \in A^I$ у довільній моделі I бази знань K, де A – концепт; a – індивід. Перевірка екземпляра ролі визначається відповідно.

Крок 4 – завершальний – повернення відповіді на запити до KB.

Щоб уникнути негативних результатів роботи алгоритмів логічного аналізу, необхідно використовувати явні критерії вибору мови дескриптивної логіки для розв'язання завдань конкретної предметної галузі. Такими критеріями можуть слугувати обчислювальна складність і завершеність алгоритмів – за наявності фактора завершеності алгоритм видає результат своєї роботи через кінцевий час.

Обчислювальна складність алгоритму визначає залежність необхідного обсягу пам'яті та часу для виконання цього алгоритму від розміру вхідних даних. Це одна з основних проблем у розвитку онтологічного опису.

Дослідження оптимізації алгоритмів починається з витягу класу обчислювальної складності завдання, яке алгоритм має вирішувати. Проаналізовано класи складності, поширені у сфері розроблення алгоритмів для дескриптивної логіки:

$Logspace \subseteq Nlogspace \subseteq P \subseteq NP \subseteq exptime \subseteq Nexptime$.

Клас *Logspace* містить завдання, для виконання яких детермінованій машині Тюрінга буде потрібно обсяг пам'яті, що перебуває в логарифмічній залежності від вхідних значень.

Префікс *N* у назвах класів *Nlogspace, NP* і *Nexptime* означає, що зазначені класи містять завдання, які можуть бути розв'язані за умови витрати ресурсів, що відповідають їх класу складності з назвою без префікса на недетермінованій машині Тюрінга. Оскільки недетермінована машина Тюрінга – це лише абстрактна модель, то алгоритм цього класу навряд чи зможе бути розв'язаним на реальному комп'ютері, відповідному до детермінованої машини, за такий самий час.

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)

Клас *P* містить всі ті проблеми, рішення яких детермінованою машиною Тюрінга поліноміально залежить від розміру входу.

Exptime – клас, що містить завдання, які детермінована машина Тюрінга виконує за кількість часу, що перебуває в експонентній залежності від вхідних значень.

Спочатку дослідження складності висновку в дескриптивній логіці були більш зосереджені на класі *P* без розгляду нерозв'язаних *NP* проблем. Це давало гарантовану вірогідність того, що алгоритм працюватиме швидко. Однак бази знань реального розміру можуть бути оброблені за розумний термін, незважаючи на високий клас складності – *Exptime* або вище.

Розуміння принципів зменшення складності в дескриптивних логіках дозволить не тільки обрати адекватну мову для опису вебонтології, але й використовувати ці знання в проєктуванні КВ.

Визначимо КВ *K* як двійку: $K =< Tbox, Abox >$, де *Tbox* – множина аксіом; *Abox* – множина тверджень. Вхідні параметри, що впливають на обчислювальну складність завдань логічного висновку: розмір *Tbox*, розмір *Abox*, розмір запиту – число параметрів запиту q.

Для завдань логічного висновку розрізняють три види обчислювальної складності:

– складність щодо даних: складність оцінюється як функція від розміру *Abox*, що використовується для мов, які спеціалізуються на більших розмірах *Abox,* а саме для завдань онтологічного доступу до БД;

– складність щодо онтології та даних: складність оцінюється як функція від розміру *Tbox* і *Abox*. Це застосовується в тому разі, коли *Tbox* може бути дуже великим і його розміром не можна зневажати;

– комбінована складність: складність оцінюється як функція від розміру q, *Tbox* і *Abox*. Це зазвичай і застосовується для алгебро-логічного подання.

Зниження обчислювальної складності алгоритмів зазвичай досягається через обмеження виразності мови. Також складність можна зменшити з допомогою винаходу алгоритмів для конкретної предметної галузі.

Швидкість виконання алгоритмів для завдань логічного висновку в технологіях дескриптивної логіки вища, ніж у структурованих алгоритмів.

Далі обґрунтовується важливість застосування алгоритмів автопобудови інтерфейсів вебсервісів. Необхідне розроблення алгоритмів для реалізації відповідності нотаціям *Serviren* унаслідок того, що таких алгоритмів на базі дескриптивної логіки наразі не існує. Водночас структуровані алгоритми, як показано вище, не мають переваг технологій алгебраїчного програмування. Необхідно зменшити кількість породжених запитів порівняно з іншими алгоритмами логічного висновку. На відміну від стандартних алгоритмів логічного висновку, існує можливість створення алгоритму запиту до бази даних, оптимізованого під *SOA*-завдання. Цей алгоритм буде вирізнятися від інших меншим розміром породжуваних запитів до бази даних.

Для онтологічної *SOA*-системи необхідні специфічні алгоритми *CRUD* на базі дескриптивної логіки. Принцип роботи алгоритму передбачає певні етапи.

Крок 1 – перевірити зміст у запиті q інформації про ролі.

Крок 2 – переформулювати атоми кожного кон'юнктивного запиту $q \in PR'$ і отримати новий запит для кожного атома переформульованої формули. Для кожної пари атомів g_1 і g_2, яка уніфікується та перебуває в тілі запиту q, обчислити кон'юнктивний запит $q' = reduce(q, g_1, g_2)$.

Загальна змінна – це змінна, що з'являється мінімум удвічі в тілі запиту, або константа. Водночас аргумент атома запиту називається незалежним, якщо він є нерозпізнаваною незагальною змінною. Незалежний аргумент атома позначається символом "_".

Функція *findr* видає результат "1", якщо в запиті перебувають атоми з двома аргументами, інакше результат дорівнює "0". Тут $q[g/g']$ означає кон'юнктивний запит, отриманий від q атома, що замінює g новим атомом g'. Функція τ ухвалює як вхідне значення кон'юнктивний запит q і повертає новий кон'юнктивний запит, отриманий заміною кожного входження незалежної змінної в q символом "_". Аргумент атома в запиті є залежним, якщо він відповідає розпізнаваній змінній або загальній змінній.

Функція *reduce* приймає як вхідні параметри кон'юнктивний запит q і два атоми g_1 і g_2, що перебувають у тілі запиту q, і повертає кон'юнктивний запит q' за допомогою застосування до q уніфікатора *MGU (Most General Unifier)* для g_1 і g_2. У процесі уніфікації g_1 і g_2 кожне входження

ISSN 2522-9818 (print)
Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27) ISSN 2524-2296 (online)

символу "_" має бути розглянуте як присутність незалежної змінної. Уніфікатор *MGU* замінює кожний символ "_" в g_1 на відповідний аргумент в g_2 і кожний символ "_" в g_2 на відповідний аргумент в g_1. Завдяки уніфікації, виконуваної за допомогою функції *reduce*, зв'язані змінні в q можуть стати незв'язаними в q'. Отже, позитивні включення, незастосовні до атомів q, пізніше можуть бути застосовані до атомів q'.

Доведення завершеності алгоритму. Нехай T – *Tbox* бази знань та q – кон'юнктивний запит до T. Тоді алгоритм $Rewwr(q,T)$ завершується.

Процес доведення описано далі. Максимальне число атомів у тілі кон'юнктивного запиту, що генерується за допомогою поданого алгоритму, дорівнює довжині початкового запиту q. Довжина запиту менша або дорівнює n, де n – розмір запиту, тобто n пропорційний числу атомів і числу компонентів запиту.

Кількість термінів, що містять кон'юнктивні запити, які генеруються за допомогою поданого алгоритму, дорівнює кількості змінних і констант, що входять в q, плюс символ "_". Отже, кардинальність такого набору менша або дорівнює $n+1$, де n – розмір запиту.

Число різних атомів, які можуть виникнути в кон'юнктивних запитах, що генеруються за алгоритмом, менше або дорівнює $m\cdot(n+1)^2$, $m = m_1 + m_2$, де m_1 – число концептів чи ролей у запиті. Водночас $m2$ – число концептів чи ролей у *Tbox* або, у разі відсутності інформації про ролі в запиті, у *Tbox/RBox*.

Алгоритм бере до уваги запити, які він згенерував. Із цього випливає, що число різних кон'юнктивних запитів, згенероване алгоритмом, скінченне. Також алгоритм не генерує запит понад один раз. Отже, алгоритм завжди завершується.

Як приклад описано процес реалізації в *SOA*-системі можливості використання *CRUD (Create, Read, Update, Delete). CRUD* – абревіатура стандартних операцій під час роботи з інформацією: створення, перегляд, оновлення та вилучення.

Реалізацією операції "створення" є алгоритм *Sircreate*. У цьому алгоритмі, користуючись раніше розглянутими алгоритмами, необхідно отримати інформацію з *Abox* і обробити її, формуючи в цьому разі конструкції для побудови інтерфейсів. Цей алгоритм може бути використано для операції оновлення, тому що за повторного виконання алгоритму на вже використаних параметрах необхідні дані будуть оновлені без ушкодження структури системи.

Нехай $ДО =< T, A >$ – база знань логіки, 3, Q, S – кон'юнктивні запити. Алгоритм *Sircreate* як вхідні параметри має: $3,S,Q,T,A$. В алгоритмі використано функцію *Consistent,* що реалізує перевірку бази знань на несуперечність. Якщо база даних має неправильний стан (порушена умова несуперечності), тоді немає сенсу виконувати подальші дії, й алгоритм завершується.

Процедура *addrole* додає інформацію про нові інтерфейси в *Tbox T*. Процедура *addinabox* додає інформацію про нові інтерфейси в *Abox A*. Функція *dba* виконує запити в триплеті над *Abox A*, що є базою даних. Процедура *sendservices* відповідає за формування коду із запитами до бази даних для вебсервісів та його відправлення на адреси клієнтів і сервера, базуючись на відповіді *Abox SOA*-системи.

Прикладом використання алгоритмів, що розроблено в роботі, застосовано інструментальні засоби, а саме *Java* – мова програмування, яка найбільше підходить для розроблення інтелектуальної сервіс-орієнтованої системи, базуючись на мовах дескриптивної логіки (*Sir*-системи). Історично склалося так, що більшість алгоритмів логічного виведення для дескриптивних логік і інструменти для роботи із *Semantic Web* реалізовані мовою *Java*. Отже, якщо обрати мову програмування *Java* для розроблення *Sir*-системи, можна інтегрувати сторонні алгоритми, реалізовані цією мовою.

Рекомендовано редактор онтологій *Protégé*, що має великий вибір інструментів для роботи з онтологіями. Водночас *Protégé* підтримує плагіни, написані мовою *Java*. Спираючись на попередній пункт, плагінів може бути реалізовано багато. Крім того, можна забезпечити інтеграцію *Sir*-системи та редактора *Protégé* за допомогою розроблення плагіна.

Висновки й перспективи подальшого розвитку

Ефективність алгебри скінченних предикатів полягає в тому, що її мовою зручно записуються закони істинності та хибності, які відіграють особливу роль, оскільки задають вимоги, необхідні й достатні для коректного введення змінних ознак

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2024. No. 1 (27)

на скінченних множинах. Закон істинності задає область зміни змінної, а закон хибності забезпечує попарну різницю всіх елементів множини, на якій задана змінна. Алгебра скінченних предикатів дає змогу інтерпретувати знання в строгій математичній формі, де різні ознаки та їх значення пов'язані між собою за допомогою булевих і предикатних операцій.

Для ефективного розв'язання рівнянь із невідомими скінченними предикатами доцільно спочатку дослідити рівняння, записані за допомогою операцій булевої алгебри, оскільки саме ці операції найчастіше трапляються під час запису довільних предикатних рівнянь.

Як показано в роботі, архітектура *OBDA* забезпечує ефективну роботу за умови невеликого розміру *Tbox* і значного розміру *Abox*. Водночас *Tbox* в *OBDA* є більш динамічною та масштабованою структурою порівняно з *Abox*. У процесі розроблення онтології необхідно зважати на ці фактори, розрізняючи статичні конструкції та конструкції, які будуть змінюватися та масштабуватися під час роботи системи, на *Abox* і *Tbox* системи.

У роботі алгоритму створено нові запити до баз даних. Ці запити зберігаються в текстовому форматі. Для обчислення пам'яті, яку займають результати роботи алгоритму, необхідно кількість створених запитів обчислити за формулою, ідентичною формулі розрахунку часу роботи алгоритму, з різницею в тому, що, замість функції розрахунку часу *t(x)*, використовувалася функція розрахунку кількості породжених запитів.

Отримано оригінальні алгоритми автоматизованої побудови інтерфейсів вебсервісів для *SOA*-архітектури, що дають змогу автоматизовано будувати інтерфейси вебсервісів. Алгоритми різняться більш загальною моделлю та швидкістю роботи, вищою за аналоги на 24–41%, залежно від вхідних параметрів алгоритмів.

Запропоновано метод опису онтологічних запитів, що вирізняється від сторонніх алгоритмів трансформації запитів більш швидкою роботою завдяки ігноруванню несуттєвих елементів вхідних показників. Різниця у швидкості роботи між отриманим алгоритмом трансформації запитів і аналогами залежить від кількості несуттєвих елементів в онтології. Доведено коректність і оцінки складності побудованих алгоритмів.

Розроблено ефективну *SOA*-систему за допомогою парадигми, основаної на онтологіях доступу до інформації. Для реалізації *SOA*-системи створено формалізм для опису процесів системи за допомогою онтології, запропоновано ефективні алгоритми та вдосконалено наявні алгоритми на основі алгебри скінченних предикатів і категорійного аналізу для роботи із *SOA*-системами.

У подальших дослідженнях необхідно докладно проаналізувати алгоритми відповідей на запит у межах технологій *OBDA* для алгебро-логічного опису й обрати ефективний алгоритм перед тим, як створювати *Sir*-системи. Для реалізації алгоритму рекомендується застосовувати мови вебонтологій *OWL* і *OWL 2*, зокрема профіль *OWL 2 QL*. Також перспективним є подання вихідних показників у вигляді запитів розроблення і використання *Sql*-подібних мов запитів. Доцільно спрощувати запити, застосовуючи методи мінімізації формул алгебри скінченних предикатів та методи подання не повністю визначених предикатів для реляційних запитів.

Для безпеки зберігання бази знань рекомендується використовувати реплікацію інформації.

Список літератури

1. Sarker U., Deraman A.B., Hasan R. Descriptive Logic for Software Engineering Ontology: Aspect Software Quality Control. *4th International Conference on Computer and Information Sciences (ICCOINS)*, Kuala Lumpur, Malaysia, 2018, P. 1–5. DOI: http://doi.org/10.1109/ICCOINS.2018.8510585
2. Rawal R., Goel K., Gupta, C. COVID-19: Disease Pattern Study based on Semantic-Web Approach using Description Logic, *IEEE International Conference for Innovation in Technology (INOCON)*, Bangluru, India, 2020, P. 1–5. DOI: http://doi.org/10.1109/INOCON50539.2020.9298278
3. Kamide N., Sequential Fuzzy Description Logic: Reasoning for Fuzzy Knowledge Bases with Sequential Information. 2020 *IEEE 50th International Symposium on Multiple-Valued Logic (ISMVL)*, Miyazaki, Japan, 2020, P. 218–223. DOI: http://doi.org/10.1109/ISMVL49045.2020.000-2
4. Shubin I. Development of conjunctive decomposition tools. 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021). CEUR Workshop Proceedings, 2021, Vol. 2870, P. 890–900. URL: https://ceur-ws.org/Vol-2870/paper67.pdf

Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

5. Sapra D., Pimentel A. D. Deep Learning Model Reuse and Composition in Knowledge Centric Networking. *29th International Conference on Computer Communications and Networks (ICCCN)*, Honolulu, HI, USA, 2020, P. 1–11. DOI: http://doi.org/10.1109/ICCCN49398.2020.9209668
6. Shubin I., Snisar S., Litvin S., Formalization and Application of Algebraic Methods in Automated Intelligent Systems. *IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T)*, Kharkiv, Ukraine, 2021, P. 67–70. DOI: http://doi.org/10.1109/PICST54195.2021.9772174
7. Каратаєв О., Ситніков Д. Метод повторного використання знань у формі логічних рівнянь. *Сучасний стан наукових досліджень та технологій в промисловості,* No. 3(25). 2023. С. 15–26. DOI: http://doi.org/10.30837/ITSSI.2023.25.015
8. Karataiev O., Sitnikov D., Sharonova N. A Method for Investigating Links between Discrete Data Features in Knowledge Bases in the Form of Predicate Equations, 7th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2023). CEUR Workshop Proceedings, 2023, P. 224–235. URL: https://ceur-ws.org/Vol-3387/paper17.pdf
9. Sharonova N., Doroshenko A., Cherednichenko O. Issues of Fact-based Information Analysis. 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021) CEUR Workshop Proceedings, 2021, Vol. 2870. URL: https://ceur-ws.org/Vol-2136/10000011.pdf
10. Zhou B., Bao J., Liu Y., Song D., BA-IKG: BiLSTM Embedded ALBERT for Industrial Knowledge Graph Generation and Reuse. *IEEE 18th International Conference on Industrial Informatics (INDIN)*, Warwick, United Kingdom, 2020, P. 63–69. DOI: http://doi.org/10.1109/INDIN45582.2020.9442198
11. He L., P. Jiang, P-SaaS: knowledge service-oriented manufacturing workflow model for knowledge collaboration and reuse. *IEEE 16th International Conference on Automation Science and Engineering (CASE)*, Hong Kong, China, 2020, P. 570–575. DOI: http://doi.org/10.1109/CASE48305.2020.9216974
12. Tereshchenko G., Gruzdo I. Overview and Analysis of Existing Decisions of Determining the Meaning of Text Documents, *International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*, Vol. 8632014, 2019, P. 645–653. DOI: http://doi.org/10.1109/INFOCOMMST.2018.8632014
13. Sharonova N., Kyrychenko I., Tereshchenko G. Application of big data methods in E-learning systems, 2021 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021), 2021. CEUR-WS, Vol. 2870. 2021, P. 1302–1311. URL: https://ceur-ws.org/Vol-2870/paper96.pdf
14. Каратаєв О., Шубін І. Проблеми повторного використання знань у процесі проєктування програмних систем. *Сучасний стан наукових досліджень та технологій в промисловості*. 2023. No 2 (24). С. 62–71. DOI: http://doi.org/10.30837/itssi.2023.24.062
15. Козирєв А., Шубін І. Метод планування завдань оброблення даних у розподілених системах з обмеженою інформацією про доступні ресурси. *Сучасний стан наукових досліджень та технологій в промисловості.* 2023. No. 3 (25). С. 27–39. DOI: https://doi.org/10.30837/ITSSI.2023.25.027
16. Karataiev O., Shubin I., Formal Model of Multi-Agent Architecture of a Software System Based on Knowledge Interpretation. *Radioelectronic and Computer Systems*. No 4 (108). 2023 P. 53–64. DOI: http://doi.org/ 10.32620/reks.2023.4.05
17. Backer P., Siemens G. Educational data mining and learning analytics. The Cambridge handbook of the learning sciences, 2019. 274 p. URL: https://www.researchgate.net/publication/316628053_Educational_data_mining_and_learning_analytics
18. Dudar Z., Shubin I., Kozyriev A. Individual Training Technology in Distributed Virtual University. *Lecture Notes in Networks and Systems*. 2021, 212 LNNS, P. 379–399. DOI: http://doi.org/10.1007/978-3-030-76343-5_20
19. Gruzdo I., Kyrychenko I., Tereshchenko G., Shanidze O. Analysis of Models Usability Methods Used on Design Stage to Increase Site Optimization. 7th International Conference on Computational Linguistics and Intelligent Systems. (COLINS-2023), Volume III: Intelligent Systems Workshop, 2023. CEUR Workshop Proceedings, Vol. 3403, P. 387–4093. URL: https://ceur-ws.org/Vol-3403/paper31.pdf

References

1. Sarker, U., Deraman, A. B., Hasan, R. (2018), "Descriptive Logic for Software Engineering Ontology: Aspect Software Quality Control", *4th International Conference on Computer and Information Sciences (ICCOINS)*, Kuala Lumpur, Malaysia, 2018, P. 1–5. DOI: http://doi.org/10.1109/ICCOINS.2018.8510585
2. Rawal, R., Goel, K., Gupta, C. (2020), "COVID-19: Disease Pattern Study based on Semantic-Web Approach using Description Logic", *2020 IEEE International Conference for Innovation in Technology (INOCON)*, Bangluru, India, 2020, P. 1–5. DOI: http://doi.org/10.1109/INOCON50539.2020.9298278

3. Kamide, N. (2020), "Sequential Fuzzy Description Logic: Reasoning for Fuzzy Knowledge Bases with Sequential Information", *2020 IEEE 50th International Symposium on Multiple-Valued Logic (ISMVL)*, Miyazaki, Japan, 2020, P. 218–223. DOI: http://doi.org/10.1109/ISMVL49045.2020.000-2
4. Shubin, I. (2023), "Development of conjunctive decomposition tools", 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021), CEUR Workshop Proceedings, Vol. 2870, P. 890–900. available at: https://ceur-ws.org/Vol-2870/paper67.pdf
5. Sapra, D., Pimentel, A. D. (2020), "Deep Learning Model Reuse and Composition in Knowledge Centric Networking", *29th International Conference on Computer Communications and Networks (ICCCN)*, Honolulu, HI, USA, P. 1–11. DOI: 10.1109/ICCCN49398.2020.9209668
6. Shubin, I. Snisar, S., Litvin, S. (2021), "Formalization and Application of Algebraic Methods in Automated Intelligent Systems", *IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T)*, Kharkiv, Ukraine, P. 67–70. DOI: http://doi.org/10.1109/PICST54195.2021.9772174
7. Karataiev, O., Sitnikov, D. (2023), "The method of reuse of knowledge in the form of logical equations", *Innovative technologies and scientific solutions for industries*, No. 3(25), P. 15–26. DOI: http://doi.org/10.30837/ITSSI.2023.25.015
8. Karataiev, O., Sitnikov, D., Sharonova, N. (2023), "A Method for Investigating Links between Discrete Data Features in Knowledge Bases in the Form of Predicate Equations", 7th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2023), CEUR Workshop Proceedings, 2023, P. 224–235. available at: https://ceur-ws.org/Vol-3387/paper17.pdf
9. Sharonova, N., Doroshenko, A., Cherednichenko, O. (2021), "Issues of Fact-based Information Analysis", *5th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021)*. CEUR Workshop Proceedings, Vol. 2870. available at: https://ceur-ws.org/Vol-2136/10000011.pdf
10. Zhou, B., Bao, J., Liu, Y., Song, D. (2020), "BA-IKG: BiLSTM Embedded ALBERT for Industrial Knowledge Graph Generation and Reuse", *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*, Warwick, United Kingdom, 2020, P. 63–69. DOI: http://doi.org/10.1109/INDIN45582.2020.9442198
11. He, L., Jiang, P. (2020), "P-SaaS: knowledge service-oriented manufacturing workflow model for knowledge collaboration and reuse", *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, Hong Kong, China, 2020, P. 570–575. DOI: http://doi.org/10.1109/CASE48305.2020.9216974
12. Tereshchenko, G., Gruzdo, I. (2019), "Overview and Analysis of Existing Decisions of Determining the Meaning of Text Documents", *International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)* 2019, P. 645–653. DOI: 10.1109/INFOCOMMST.2018.8632014
13. Sharonova, N., Kyrychenko, I., Tereshchenko, G. (2021), "Application of big data methods in E-learning systems", 2021 5th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2021), CEUR Workshop Proceedings, Vol. 2870, P. 1302–1311. available at: https://ceur-ws.org/Vol-2870/paper96.pdf
14. Shubin, I., Karataiev, O. (2023), "Reuse of Information Based on The Interpretation of Knowledge", *Innovative technologies and scientific solutions for industries.* No. 2 (24), P. 62–71. DOI: http://doi.org/10.30837/itssi.2023.24.062
15. Kozyriev, A., Shubin, I. (2023), "Method of Planning Data Processing Tasks in Distributed Systems with Limited Information About Available Resources", *Innovative technologies and scientific solutions for industries*. No. 3 (25), P. 27–39. DOI: http://doi.org/10.30837/ITSSI.2023.25.027
16. Karataiev, O. Shubin, I. (2023), "Formal Model of Multi-Agent Architecture of a Software System Based on Knowledge Interpretation", *Radioelectronic and Computer Systems*. No. 4 (108), P. 53–64. DOI: http://doi.org/10.32620/reks.2023.4.05
17. Backer, P., Siemens, G. (2019), "Educational data mining and learning analytics", The Cambridge handbook of the learning sciences, 274 p. available at: https://www.researchgate.net/publication/316628053_Educational_data_mining_and_learning_analytics
18. Dudar, Z., Shubin, I., Kozyriev, A. (2021), "Individual Training Technology in Distributed Virtual University". *Lecture Notes in Networks and Systems*. 212 LNNS, P. 379–399. DOI: http://doi.org/10.1007/978-3-030-76343-5_20
19. Gruzdo, I., Kyrychenko, I., Tereshchenko, G., Shanidze, O. (2023), "Analysis of Models Usability Methods Used on Design Stage to Increase Site Optimization", 7th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2023), Volume III: Intelligent Systems Workshop, CEUR Workshop Proceedings, Vol. 3403, P. 387–4093. available at: https://ceur-ws.org/Vol-3403/paper31.pdf

Received 10.03.2024

Сучасний стан наукових досліджень та технологій в промисловості. 2024. №1 (27)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

Відомості про авторів / About the Authors

Дудар Зоя Володимирівна – Харківський національний університет радіоелектроніки, професорка кафедри програмної інженерії, Харків, Україна; e-mail: zoia.dudar@nure.ua, ORCID ID: https://orcid.org/0000-0001-5728-9253

Літвін Світлана Геннадіївна – Харківський національний університет радіоелектроніки, аспірантка кафедри програмної інженерії, Харків, Україна; e-mail: svitlana.litvin@nure.ua; ORCID ID: https://orcid.org/0000-0002-7183-6345

Dudar Zoia – Kharkiv National University of Radio Electronics, Professor at the Department of Software Engineering, Kharkiv, Ukraine.

Litvin Svitlana – Kharkiv National University of Radio Electronics, Postgraduate Student at the Department of Software Engineering, Kharkiv, Ukraine.

ONTOLOGICAL DESCRIPTION METHOD FOR BUILDING SERVICE-ORIENTED DISTRIBUTED LEARNING SYSTEMS

The **subject** of the research is the analysis and justification of the use of software engineering processes based on ontologies of data access and representation of access to knowledge bases and knowledge reuse in the language of algebra of finite predicates, which is a class of descriptive logics. This approach allows you to use the advantages of ontologies and logic programming in the refactoring of distributed distance learning systems. The **purpose** of the work is to create a formalism for describing the interaction of services and algorithms for building web service interfaces for the implementation of an effective SOA system using the paradigm of access to data based on ontologies. To achieve the **goal**, the following tasks were solved: an analysis of the main types of interaction structures of SOA services, an overview of implementations of systems working with a family of descriptive logic languages, namely, algebra of finite predicates, and a description of the structure of the ontology necessary for the operation of service-oriented systems, which has the markup of the Semantic Web standard using finite predicate algebra equations. The **task** is an urgent task of ensuring the organization of interaction of Internet resources in the development of distributed virtual learning systems, and such interaction is enabled by the service-oriented architecture of software development. The article examines modern technologies of the Semantic Web and their role in ensuring the creation of applied software systems of distributed learning resources built using distributed interactive services. The considered **methods** and materials include methods of algebra of finite predicates, theory of algorithms, object-oriented design, theory of unification. **The results.** The semantics of the SOA system is obtained, which makes it possible to describe SOA systems in the languages of descriptive logic. The effectiveness of the mathematical formalism of the algebra of finite predicates for the tasks of logical analysis of applied ontologies, the use of knowledge reuse methods, and the description of service-oriented systems is shown. The necessity of using algorithms for automatic construction of web interfaces is shown. **The conclusions** of the work highlight the advantages of the researched solution, namely, algorithms for the automated construction of web service interfaces for SOA architecture, which make it possible to automatically build web service interfaces. Algorithms differ in a more general model and speed of work and require an assessment of complexity.

Keywords: software engineering; service-oriented architecture; algebra of finite predicates; ontologies; algebraic programming; distributed virtual learning environment.

Бібліографічні описи / Bibliographic descriptions

Дудар З. В., Літвін С. Г. Метод онтологічного опису в побудові сервіс-орієнтованих систем розподіленого навчання. *Сучасний стан наукових досліджень та технологій в промисловості*. 2024. № 1 (27). С. 39–53. DOI: https://doi.org/10.30837/ITSSI.2024.27.039

Dudar, Z., Litvin, S. (2024), "Ontological description method for building service-oriented distributed learning systems", *Innovative Technologies and Scientific Solutions for Industries*, No. 1 (27), P. 39–53. DOI: https://doi.org/10.30837/ITSSI.2024.27.039