УДК 004.855.91:514.112
DOI: 10.18523/2617-3808.2023.6.17-23

Жежерун О. П., Смиш О. Р., Пруднікова А. О.

ПІДХОДИ ДО ПОБУДОВИ ВИВОДУ В ОНТОЛОГІЧНІЙ БАЗІ ЗНАНЬ

У статті наведено два приклади реалізації виводу в онтологічній базі знань. Один — із використанням SWRL-правил, другий — як систему з елементами обробки природної мови. Наведено опис створених фрагментів таксономічної ієрархії для предметної галузі (планіметрії). Як приклади використано прості задачі відкритого типу зі шкільного підручника з геометрії. Продемонстровані процедури є частиною рекомендаційної навчальної системи, яку розробляють на факультеті інформатики Національного університету «Києво-Могилянська академія».

Ключові слова: обробка природної мови, рекомендаційна система, геометрія, база знань, онтологія, Protégé, OWL, Reasoner, SWRL.

Вступ

Сьогодні дедалі гостріше формується попит на системи, які полегшують процес навчання, автоматизують його та уможливлюють самостійність учня. Інтелектуальні навчальні системи — це один із перспективних напрямів, і створення таких систем сприяє якіснішому навчанню. Ця робота стосується розвитку навчальної рекомендаційної системи, яку реалізують на факультеті інформатики Національного університету «Києво-Могилянська академія». Система орієнтована на допомогу у вивченні геометрії учням середньої школи та має на меті:

– навчити правильно аналізувати умову задачі, будувати послідовність міркувань, які забезпечують розв'язок задачі, правильно будувати графічний малюнок, коректно документувати процес розв'язку;
– сприймати текст задачі, поданий природною українською мовою (зі шкільного підручника), розв'язувати задачу автоматизовано, будувати графічний малюнок, формувати протокол розв'язку природною мовою в спосіб, зрозумілий учневі;
– здійснювати загальну консультативну підтримку процесу навчання, забезпечуючи учня необхідним теоретичним матеріалом.

Зазвичай формування системи відбувається покроково, тому зараз розробки зосереджені на матеріалі 7 класу, але надалі заплановано дійти до рівня допомоги учневі під час підготовки до ЗНО/НМТ.

Необхідно пояснити, що ідеологічно система бере свій початок від ідей В. Глушкова, запропонованих у концепції «Алгоритму Очевидності» [2]. Розвиваючи в Києві в Інституті кібернетики систему автоматичного доведення теорем у математиці, Віктор Михайлович робив акцент не на створенні «ефективних універсальних процедур» доведення теорем, а на розвитку програмних засобів, які співпрацюють із математиком у процесі доведення теореми, забезпечують інформаційну підтримку й можуть бути орієнтовані на програмування рішень навіть «для однієї важкої теореми». Відповідно до планів В. Глушкова [1], у пам'ять комп'ютера заноситься певний фрагмент математичної теорії, наприклад основи теорії груп. Математичні тексти представляються спеціальною мовою для запису таких текстів. Твердження, яке математик вводить у систему, ставало «очевидним», якщо машинний алгоритм встановлював, що воно є логічним наслідком накопиченого в системі матеріалу. Якщо система доводила певне твердження, то вона видавала протокол доведення у форматі, який відповідав науковій публікації.

Проводячи такі аналогії, варто сказати, що теперішня мета — це зробити очевидним для учня процес розв'язку задачі, використовуючи різноманітні процедури, забезпечити інформаційну підтримку. У цій статті є розглянуто дві різні процедури, які використовуються в системі: одна базується на використанні логічних правил, а інша поєднує аналіз української природної мови з онтологічною базою знань.

Онтологічні бази знань

Онтологія — це концептуалізація певної галузі знань і значущий напрям у галузі інтелектуаль-

них систем. Концептуалізацією називають процес створення та встановлення зв'язків між основними поняттями, які виділено на основі класифікації базових термінів у певній предметній галузі. Онтології уможливлюють представлення знань про поняття та відношеннями між ними.

Онтологія формується з таких елементів:

1. Поняття: терміни або поняття, що використовують у певній царині, які можуть мати ієрархічну структуру та впорядковані за принципом «батько–дитина».
2. Відносини: зв'язки між поняттями. Вони можуть бути бінарними (між двома поняттями) або багатозначними. Наприклад: «має атрибут», «є частиною» тощо.
3. Властивості: використовують для опису та характеристики термінів.
4. Аксіоми: певні правила або обмеження, що встановлюють на відносини, властивості або взаємодію між термінами.
5. Логічні правила: правила, які використовують для формалізації логічних відносин і виведення нових знань із наявних.

Онтологію можна представити в графічному вигляді або описати формальною мовою. Для формального опису онтології використано мову OWL (Ontology Web Language). Це мова семантичного вебу, яка призначена для представлення складних знань про предметну галузь та її зв'язків. Знання, виражені в OWL, можна використовувати в комп'ютерних програмах, наприклад, щоб переконатись в узгодженості цих знань [4].

Для створення бази знань спершу потрібно конкретизувати задачу та створити детальний опис кроків. Цей алгоритм допоможе створити структуру бази знань і визначити необхідні елементи, властивості та залежності між ними для моделювання геометричних задач. Здійснено такі етапи:

1. Проаналізовано та відібрано клас задач, що використовуватимуться в цій базі знань. Для початку обрано порівняно прості задачі, як-от задачі на знаходження кутів у трикутнику, задачі на знаходження площі та периметра.
2. Створено концептуальну модель, тобто визначено основні класи, які будуть залучені для розв'язування задач. Визначено властивості для цих класів.
3. Встановлено ієрархію класів, створено підкласи, визначено обмеження та певні правила для властивостей.
4. Додано екземпляри (індивіди). Встановлено зв'язки між індивідами.
5. Зроблено перевірку та тестування. Це значущий крок, адже на ньому відбувається перевірка правильності моделі через тестування різних шляхів.
6. Визначено запити, що можуть бути виконані до бази знань (наприклад, запит на отримання всіх кутів конкретного трикутника та їхньої градусні міри). Реалізовано логіку для виконання цих запитів.
7. Використано створену базу знань для геометричних задач, для надання необхідної інформації. Також застосовано систему для отримання нової інформації на основі встановлених властивостей та правил.

Першим кроком для створення онтології є формування ієрархії класів. Необхідно визначити основні класи, що використовуватимуться в базі знань.

Надкласом є «Пряма», підкласами якої є «Відрізок» і «Промінь». «Геометрична_фігура», що належить до класу «Відрізок», має один підклас.

Своєю чергою, «Трикутник» має такі шість підкласів: «Гострокутний_трикутник», «Прямокутний_трикутник», «Рівнобедрений_трикутник», «Рівносторонній_трикутник», «Тупокутний_трикутник», «Різносторонній_трикутник».

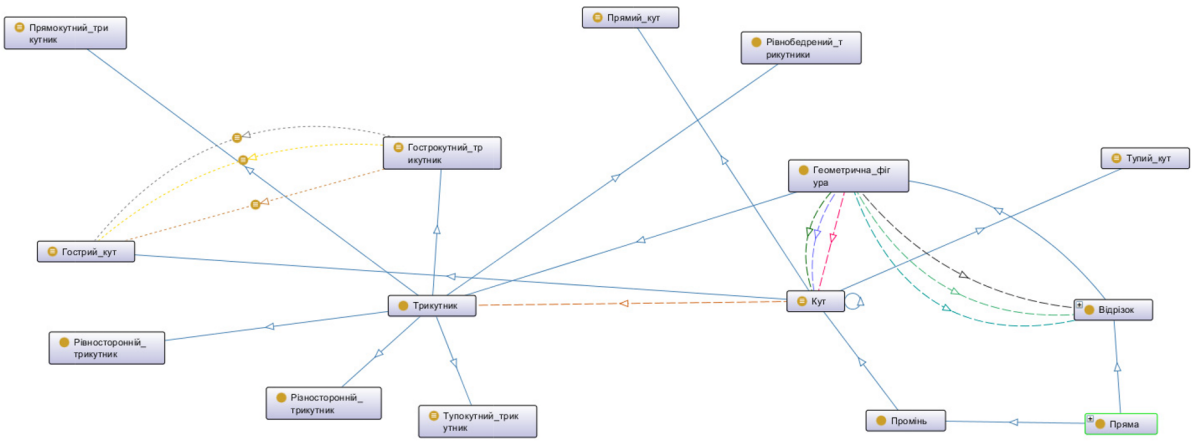

Рис. 1. Ієрархія класів і їхніх зв'язків

До класу «Промінь» належить підклас «Кут», який має три підкласи: «Гострий_кут», «Прямий_кут», «Тупий_кут».

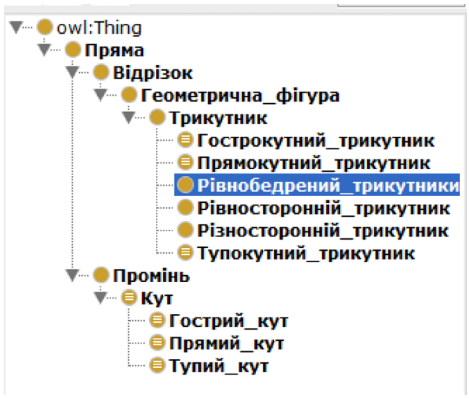

Рис. 2. Ієрархія класів

На другому кроці варто визначити та додати властивості для кожного класу, щоб описати їхні характеристики. У Protégé властивості бувають двох типів: «Object Properties» і «Data Properties». До першого типу належать властивості, які використовують для встановлення взаємозв'язків між класами.

Після створення властивостей треба встановити обмеження та визначення для властивостей і класів. Прикладом може бути визначення для класу «Тупий_кут».

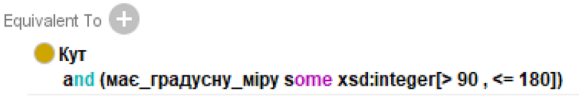

Рис. 3. Клас «Тупий_кут»

Щоб зазначити обмеження, що в кута може бути тільки одне значення градусної міри, потрібно обрати Functional для властивості «має_градусну_міру».

Для реалізації наступного кроку треба створити індивіди.

Для встановлення властивості «сторона_a» потрібно створити відповідний відрізок, аналогічно зробити й для кутів. Тобто, щоб створити повноцінний трикутник, необхідно додати до індивідів три відрізки та три кути.

У Protégé існують вбудовані інструменти для валідації. «OWL DL reasoner» — це механізм для перевірки бази знань на відповідність стандарту OWL. Серед них є Pellet, HermiT та FaCT++ [5].

SWRL (Semantic Web Rule Language) [7] є мовою правил, яку використовують для виконання логічних виводів за допомогою наявних знань, що зберігаються в онтології. Правило визначається двома основними елементами: антецедентом (тілом) і консеквентом (головою). Тіло визначає умови, що мають виконатись, а голова відповідає за дії, що буде зроблено, якщо умова виконається.

Приклади розв'язування задач у Protégé системі

Приклад задачі № 1. Для розв'язання задачі на знаходження периметра правило матиме такий вигляд:

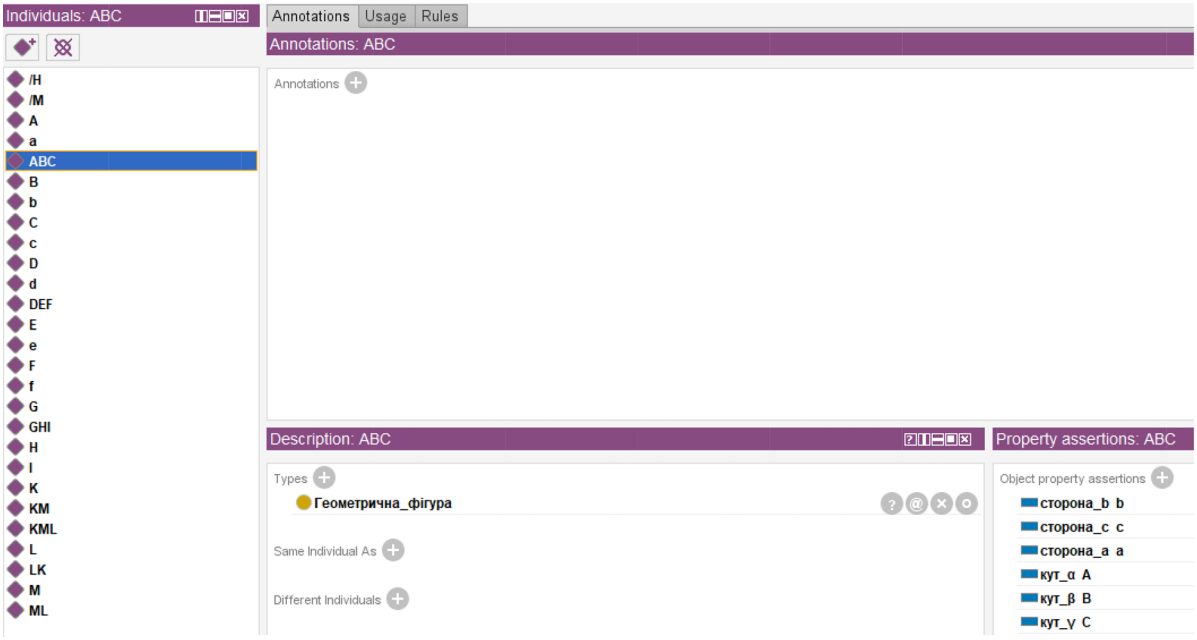

Рис. 4. Індивіди

```
Rule: Трикутник(?t), сторона_a(?t, ?a), сторона_b(?t, ?b), сторона_c(?t, ?c),
має_довжину(?a, ?l1), має_довжину(?b, ?l2), має_довжину(?c, ?l3), add(?res, ?l1, ?l2, ?
l3) -> периметр(?t, ?res)
```

Рис. 5. Правило для знаходження периметра

У цьому SWRL правилі визначено, що якщо об'єкт «?t» є трикутником і має сторони «?a», «?b», «?c», що своєю чергою мають відповідні довжини: «?l1», «?l2», «?l3», то тоді цей трикутник має периметр «?res», який обраховано за допомогою вбудованої математичної функції «add».

SPARQL є мовою запитів, яка уможливлює виконання пошуку інформації в RDF-графах. Її використовують для створення та формулювання складних запитів, щоб отримати дані. У SPARQL, як і в SQL, запити побудовані з певних ключових слів, як-от SELECT, WHERE, FILTER, ORDER BY. Їх застосовують для визначення шаблонів запитів та умов фільтрації вихідних даних.

У Protege є декілька табів правил SPARQL: «SPARQL Query» та «SNAP SPARQL Query». Перше використовують, якщо потрібно отримати результати, що наявні в базі знань, а SNAP тоді, коли потрібно доступитись до висновків, що зробив Reasoner. Важливо пам'ятати, що при використанні цього табу Reasoner має бути активним. У цій роботі використано SNAP, адже важливим є саме результат виконання Reasoner.

Наприклад:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX :<http://www.semanticweb.org/nasty/ontologies/new.owl#>

SELECT ?s ?o
        WHERE { ?s :периметр ?o }
```

Рис. 6. Приклад «SNAP SPARQL Query»

Результатом виконання цього запиту буде перелік трикутників та їхній периметр:

Execute

?s	?o
:ABC	12
:DEF	26
:KML	18

Рис. 7. Результат виконання для прикладу

Приклад задачі № 2. Текст задачі: «Периметр рівнобедреного трикутника = 40 см, а основа = 10 см. Знайдіть бічні сторони трикутника.».

Для розв'язання задачі на знаходження бічних сторін трикутника знадобиться SWRL-правило:

```
Трикутник(?t) ^ має_основу(?t, ?o) ^ має_периметр(?t, ?p) ^
swrlb:subtract(?p, ?o, ?) ^ swrlb:divide(?, 2, ?res) ->
результат(?res)
```

Рис. 8. SWRL-правило

Це правило виходить із того, що бічну сторону трикутника може бути знайдено через різницю між периметром та основою, поділену удвічі.

Наступним кроком є створення відповідних індивідів. Додано та створено трикутник ABC. Також додано його сторони та кути. У результаті отримано такі взаємозв'язки:

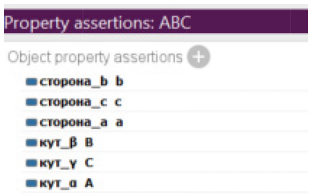

Рис. 9. Додавання трикутника

З умови задачі відомо, що периметр 40, а основа 10:

Рис. 10. Встановлення відомих даних

Після запуску Reasoner, виведено такі висновки:

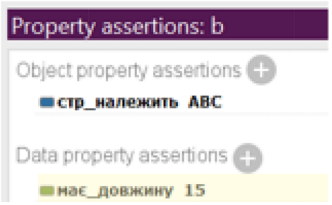

Рис. 11. Вивід відповіді

Для сторони b Reasoner визначив, що довжина дорівнюватиме 15, таку саму довжину матиме й сторона c.

Отже, у такому разі, після введення інформації про трикутник, його основу та периметр, можна використати SWRL-правило для автоматизованого знаходження бічних сторін трикутника на основі заданих умов.

```
# text = Периметр рівнобедреного трикутника дорівнює 40 см, а основа дорівнює 10 см.
1   Периметр    периметр    NOUN    Ncmsnn  Animacy=Inan|Case=Nom|Gender=Masc|Number=Sing   4   nsubj   _   _
2   рівнобедреного  рівнобедрений   ADJ Ap-msgf-ep  Aspect=Perf|Case=Gen|Gender=Masc|Number=Sing|VerbForm=Part|Voice=Pass   3   amod    _   _
3   трикутника  трикутник   NOUN    Ncmsgn  Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing   1   nmod    _   _
4   дорівнює    дорівнювати VERB    Vmpip3s Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin    0   root    _   _
5   40  40  NUM Mlc-a   Case=Acc|NumType=Card|Uninflect=Yes 6   nummod:gov  _   _
6   см  см  NOUN    Y   Abbr=Yes|Animacy=Inan|Case=Gen|Gender=Masc|Number=Plur|Uninflect=Yes    4   obj _   SpaceAfter=No
7   ,   ,   PUNCT   U   _   10  punct   _   _
8   а   а   CCONJ   Ccs _   10  cc  _   _
9   основа  основа  NOUN    Ncfsnn  Animacy=Inan|Case=Nom|Gender=Fem|Number=Sing    10  nsubj   _   _
10  дорівнює    дорівнювати VERB    Vmpip3s Aspect=Imp|Mood=Ind|Number=Sing|Person=3|Tense=Pres|VerbForm=Fin    4   conj    _   _
11  10  10  NUM Mlc-a   Case=Acc|NumType=Card|Uninflect=Yes 12  nummod:gov  _   _
12  см  см  NOUN    Y   Abbr=Yes|Animacy=Inan|Case=Gen|Gender=Masc|Number=Plur|Uninflect=Yes    10  obj _   SpaceAfter=No
13  .   .   PUNCT   U   _   4   punct   _   _

# sent_id = 2
# text = Знайдіть бічні сторони трикутника.
1   Знайдіть    знайти  VERB    Vmem-2p Aspect=Perf|Mood=Imp|Number=Plur|Person=2|VerbForm=Fin  0   root    _   _
2   бічні   бічний  ADJ Ao--pasn    Animacy=Inan|Case=Acc|Number=Plur   3   amod    _   _
3   сторони сторона NOUN    Ncfpan  Animacy=Inan|Case=Acc|Gender=Fem|Number=Plur    1   obj _   _
4   трикутника  трикутник   NOUN    Ncmsgn  Animacy=Inan|Case=Gen|Gender=Masc|Number=Sing   3   nmod    _   SpaceAfter=No
5   .   .   PUNCT   U   _   1   punct   _   SpaceAfter=No
```

Рис. 12. Результат виконання UDPipe 2.12 моделі

NLP система

Далі розглянуто роботу інтелектуальної системи зі сформованою геометричною онтологією та елементами обробки природної української мови. Сирий текст задачі, що надходить до системи: «Периметр рівнобедреного трикутника = 40 см, а основа = 10 см. Знайдіть бічні сторони трикутника.».

Для виведення розв'язку цієї задачі програма проходить декілька етапів: попереднє оброблення сирого тексту, де спеціальні символи, що властиві математичним задачам, замінюються на буквену форму (наприклад, знак «=» — на «дорівнює»); оброблення тексту за допомогою UDPipe 2.12 моделі [6], що розбиває текст на токени, далі формує леми слів, за допомогою лематизації, визначає морфологічні риси кожного токена, проставляє залежності між словами, за допомогою додатково встановлених частин мови (цей етап є значущим, оскільки мінімізує словоформи та створює зв'язки між токенами); додаткова перевірка лем за допомогою словника ВЕСУМ [3], для зменшення похибки моделі, що використано.

Після етапів, що присвячено попередньому обробленню сирого тексту, відбувається етап, який намагається вичленити з підготовленого тексту всі можливі дані, які прописано в програмі, зокрема в геометричній онтології.

У такий спосіб програма виокремить ключові токени, як-от «трикутник», «периметр», «основа», «рівнобедрений», а також числа, які можна пов'язати зі словами-токенами.

Треба також зазначити, що в програмі сформовано додатковий модуль, який відповідає за те, щоб визначити, що є шуканим у задачі. У випадку задачі, яку розглянуто, програма поверне «['plur', 'side', 'side', 'polygon']». Це позначення означає, що програмі потрібно повернути бічні сторони фігури, у множині. Для формування такого позначення використано створений алгоритм, на основі слів, що вжито у тексті.

Після етапів опрацювання тексту починається етап застосування сформованої геометричної онтології. На базі цієї онтології до програми спершу заносяться всі вичленені дані з підготовленого тексту, із зіставленням фігури, їхніх вказаних елементів і вказаних величин цих елементів. Наприклад, у випадку з задачею, що використано як приклад, програма формує екземпляр класу «Трикутник», де вказується атрибут «Рівнобедрений», як різновид фігури. Також значен-

```python
if self.perimeter > 0 and hasattr(self, 'sort') and self.sort == "рівнобедрений" and 1 <= known_s < 3:
    base = self.sides[1]
    side1 = self.sides[0]
    side2 = self.sides[2]
    if base.length == 0:
        base.length = self.perimeter - 2 * side1.length
    else:
        length = (self.perimeter - base.length) / 2
        side1.length = length
        side2.length = length
```

Рис. 13. Фрагмент коду

ня периметра та основи теж вносяться до системи у відповідні змінні. Далі, програма із наявними значеннями потрапляє на перевірку.

У коді на рис. 13 видно, що програма визначає значення двох бічних сторін рівнобедреного трикутника та вносить їх до відповідних змінних.

У результаті виконання програма створить назву для заданого рівнобедреного трикутника, оскільки цього не було дано в умові, та поверне, що AB = 15.0 см, CA = 15.0 см. Відповідь коректна.

Висновок

У статті описано та продемонстровано роботу двох процедур, які базуються на побудованій онтології. Одна із використанням SWRL-правил, інша — з елементами обробки природної мови. Описано застосовані фрагменти таксономічної ієрархії для предметної онтології планіметрії. На прикладах простих задач відкритого типу з короткою відповіддю проілюстровано їхні можливості. Висвітлено особливості роботи кожної системи.

Під час тестування задач Reasoner у системі Protégé не виявив суперечностей або неточностей. Натомість він знайшов шукане та визначив тип класу, застосовуючи SWRL-правила й обмеження.

Продемонстровано елементи обробки природної української мови, на якій сформульовано задачу.

Варто зазначити, що дві системи впоралися із поставленою задачею та знайшли розв'язок тому, що додатково було прописано певні правила, за допомогою яких це вдалося. Отже, чим повнішою буде побудована онтологія, тим більшу кількість класів задач система зможе охопити.

Список літератури

1. Глушков В. М. Некоторые проблемы теории автоматов и искусственного интеллекта / В. М. Глушков // Кибернетика. — 1970. — № 2. — С. 3–13.
2. Летичевский А. А. Алгоритм Очевидности Глушкова / А. А. Летичевский, А. В. Лялецкий, М. К. Мороховец // Кибернетика и системный анализ. — 2013. — № 4. — С. 3–16.
3. Рисін А. Великий електронний словник української мови (ВЕСУМ) [Електронний ресурс] / А. Рисін, В. Старко. Вебверсія 6.0.1. 2005-2022. Режим доступу: https://r2u.org.ua/vesum/.
4. Bechhofer S. OWL: Web Ontology Language / S. Bechhofer // liu L., Özsu M. T., eds. Encyclopedia of Database Systems. — Springer, Boston, MA, 2009. https://doi.org/10.1007/978-0-387-39940-9_1073.
5. List of Reasoners [Eletronic resource] // OWL @ Manchester. — Mode of access: http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/.
6. Straka Milan. UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task / Milan Straka // Proceedings of CoNLL 2018: The SIGNLL Conference on Computational Natural Language Learning. — Association for Computational Linguistics, Stroudsburg, PA, USA, 2018. — Pp. 197–207.
7. SWRL: A Semantic Web Rule Language Combining OWL and RuleML [Eletronic resource]. — Mode of access: w3.org.

References

Bechhofer, S. (2009). OWL: Web Ontology Language. In Liu, L., Özsu, M. T. (Eds.), *Encyclopedia of Database Systems*. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_1073.

Glushkov, V. M. (1970). Nekotorie problemi teorii avtomatov i iskusstvennogo intenllekta. *Kibernetika*, *2*, 3–13 [in Russian].

Letichevskii, A. A., Lyaletskii, A. V., & Morokhovets, M. K. (2013). Algoritm Ochevidnosti Glushkova. *Kibernetika i sistemnii analiz*, *4*, 3–16 [in Russian].

List of Reasoners. OWL @ Manchester. http://owl.cs.manchester.ac.uk/tools/list-of-reasoners/.

Rysin, A., & Starko, V. Velykyi elektronnyi slovnyk ukrainskoi movy (VESUM). Web version 6.0.1. 2005–2022. Mode of access: https://r2u.org.ua/vesum/.

Straka, Milan. (2018). UDPipe 2.0 Prototype at CoNLL 2018 UD Shared Task. In *Proceedings of CoNLL 2018: The SIGNLL Conference on Computational Natural Language Learning*, 197–207. Association for Computational Linguistics, Stroudsburg, PA, USA.

SWRL: A Semantic Web Rule Language Combining OWL and RuleML. http://w3.org.

O. Zhezherun, O. Smysh, A. Prudnikova

APPROACHES TO INFERENCE SEARCH IN THE ONTOLOGICAL KNOWLEDGE BASE

The article provides two approaches for the implementation of the inference search procedure in the ontological base. One is based on the SWRL-rules, the other is a system with the natural language processing elements. The procedures have been established as a part of the recommendation system, which is developed at the Faculty of Informatics at National University of Kyiv-Mohyla Academy.

We also add a description of the created approaches with their fragments of the taxonomic hierarchy for the planimetry ontology. For the performance examples, simple open-type problems with a short answer taken from the school geometry textbooks are used. The features of the approaches, how they work, as well as the capabilities they have are reviewed.

The approach with natural language processing capabilities has a module for preprocessing raw Ukrainian text using the UDPipe 2.12 model, a module for rechecking the lemmas by using VESUM dictionary, a module with a described planimetry ontology, and a module for creating an illustration of the figures (triangles).

To better illustrate the capabilities of the approaches on equal terms, we tried to use the same geometric problem. English translation of the problem: «Perimeter of an isosceles triangle = 40 cm and base = 10 cm. Find the legs of the triangle.». To solve this problem, systems should have a rule that subtracts the base from the perimeter, divides it by two, and sets the result to the correct variables (in our case, the legs of the triangle). We demonstrated that both approaches solved the problem successfully. But in order to achieve it, minor changes were added. Therefore, the more complete the ontology is, the greater the number of problem types the systems are able to cover.

Having analyzed the results of the study, we can conclude that the systems are effective for solving geometric problems. The next step may be to combine the capabilities of the approaches to form a more complete knowledge base.

Keywords: natural language processing, recommendation system, geometry, knowledge base, ontology, Protégé, OWL, Reasoner, SWRL.

Матеріал надійшов 15.11.2023