Міністерство освіти і науки України
Український державний університет науки і технологій

Кваліфікаційна наукова
праця на правах рукопису

Жучий Лариса Ігорівна

УДК 0004.82

ДИСЕРТАЦІЯ

ІНТЕГРАЦІЯ ТА УЗГОДЖЕННЯ ДАНИХ ІНФОРМАЦІЙНИХ СИСТЕМ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ ОНТОЛОГІЧНИМИ ЗАСОБАМИ

122 – комп’ютерні науки
12 – інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____Л.І. Жучий

Науковий керівник Шинкаренко Віктор Іванович
доктор технічних наук, професор

Дніпро – 2022

АНОТАЦІЯ

Жучий Л. І. Інтеграція та узгодження даних інформаційних систем залізничного транспорту онтологічними засобами. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 «Комп'ютерні науки» – Український державний університет науки і технологій, Дніпро, 2022.

Дисертація присвячена питанням обґрунтування та розробки методів інтеграції роз'єднаних інформаційних систем залізниць та відповідної нормативної документації, підвищення контролю їх узгодженості.

У дисертаційній роботі отримані нові науково обґрунтовані теоретичні та експериментальні результати, що у сукупності є суттєвими для рішення актуальної науково-технічної задачі обробки даних та перевірки їх узгодженості на залізничному транспорті.

Наукова новизна отриманих результатів.

В роботі вперше:

1) виконано концептуалізацію і формалізацію онтології різних типів джерел, що дозволяє об'єднувати та узгоджувати таблиці, представлені в різних інформаційних та програмних середовищах. На відміну від інших, вона враховує специфічну табличну структуру даних та заснована на поступовому узагальненні зв'язків між елементами даних;
2) виконано концептуалізацію прототипу онтологічного забезпечення залізничної колії та його інтеграцію з поїзною, вагонною і відправочною моделями АСК ВП УЗ-Є. На відміну від інших, він враховує формалізовані положення нормативно-правових актів;
3) формалізовано процедуру формування онтологій залізничного домену засобами конструктивно-продукційного моделювання. На відміну від інших, вона враховує інтеграцію роз'єднаних джерел даних.

Поліпшено:

4) технології обробки даних і процесів інформаційного супроводження: паспорту під'їзної колії; допустимих швидкостей руху поїздів на залізничних коліях загального користування на основі відповідного онтологічного забезпечення;
5) методи конструктивно-продукційного моделювання у частині розробки ланцюгів взаємопов’язаних конструкторів та їх застосування у залізничному домені.

Отримали подальший розвиток:

6) засоби семантичного анотування нормативних документів. Для цього представлена схема анотування на основі онтології залізничного домену.

Практичне значення отриманих результатів полягає в розробці прототипу на основі онтологій, який дає формалізоване представлення технологічних процесів залізниці, що дозволяє перевірити узгодженість даних інформаційних систем між собою та з нормативною документацією.

Зараз на українських залізницях використовується централізована система «Єдина Автоматизована Система Управління Вантажними Перевезенням» (АСК ВП УЗ-Є), яка об'єднує основні операційні підсистеми Укрзалізниці. У зв'язку з тим, що АСК ВП УЗ-Є створена шляхом об'єднання і доопрацювання окремих підсистем, віна до сих пір страждає від фрагментації, що є певною проблемою. Крім того, виникла потреба в певній децентралізації у зв'язку з реорганізацією Укрзалізниці та створенням на її базі кількох компаній та співпрацею з інформаційними системами зарубіжних країн, іншими перевізниками та деякими замовниками.

У першому розділі виконано аналіз онтологічних розробок в залізничному та інших доменах. Встановлено що онтології на залізничному транспорті Євросоюзу використовуються для інтеграції даних опису інфраструктури, розкладу руху поїздів та інших. При цьому недостатньо уваги приділяється нормативному забезпеченню перевізного процесу. Існують програмні засоби для анотування текстів, видобування знань з таблиць і розробки онтологій, але вони не використовуються для підтримки перевізного процесу на залізничному транспорті України. Визначено, що актуальним

завданням є анотація нормативної документації для встановлення зв'язку між онтологією та текстами інструкцій.

Науково обґрунтовано можливості використання онтологічних засобів на залізничному транспорті для: формалізації нормативного забезпечення; перетворення та інтеграції даних; перевірки узгодженості даних та інструкцій інформаційної системи.

У другому розділі представлені розроблені методи: концептуалізації табличного представлення знань; формування онтології залізничної інфраструктури; інтеграції прототипу онтології залізничної інфраструктури з: онтологією джерел; онтологією поїзної, вагонної і відправочної моделей залізниці; конструктивно-продукційного моделювання онтологічного забезпечення документообігу.

Здійснена формалізація онтологічних елементів табличного представлення знань та їх відношень у вигляді графічної нотації формальної мови. Запропоновано метод формування складних понять. Концептуалізація здійснюється на основі глибокої деталізації відношень між різними поняттями.

В основі концепту лежить застосування відношення порядку, варіативність якого дає можливість моделювати різні реальні об'єкти різної природи.

Розроблено метод семантичного контролю заснований на моделі багаторівневої конкретизації та інтеграції онтологій джерел та залізничної інфраструктури. Механізми реалізації складових онтологій та їх інтеграції продемонстровано на прикладі.

У третьому розділі розроблено засоби онтологічної підтримки для паспорту під'їзної колії та допустимих швидкостей залізничних колій загального користування.

Виконано формалізацію табличного представлення даних та правил інструкцій залізничного транспорту для паспорту під’їзної колії та обмежень швидкості руху онтологічними засобами та інструментами «перетворення даних» (data wrangling) та «видобування даних» (data extraction). Це виключає частину трудомісткого процесу введення концептів в онтологію. Розроблено засоби семантичного контролю даних з різних джерел про допустимі швидкості на елементах справних залізничних колій загального користування. Для формалізації обмежень «Правил технічної експлуатації

залізниць України» використовується метод семантичного анотування. Формування онтології здійснюється на основі композиції відношень.

Розроблено модульну онтологію для інтеграції даних відомостей стрілочних переводів і колій, наказів, які встановлюють допустимі швидкості на елементах залізничної інфраструктури, «Правил технічної експлуатації залізниць України» та «Будівельних норм». Цей підхід забезпечує зв'язок між природно-мовними інструкціями, інформаційними системами та онтологіями обмежень швидкості поїздів.

Онтологія несправності залізничної колії об'єднує інформацію з наступних джерел: баз даних, станційних креслень та природно-мовних інструкцій. Семантична перевірка здійснюється за допомогою формалізованих інструкцій з технічного обслуговування колії, колійних робіт та автоматизованої видачі попереджень про обмеження швидкості. Застосування розробленої онтології надає можливість підвищення безпеки руху поїздів шляхом перевірки дотримання швидкісного режиму і несправності колії, своєчасного зняття обмежень і перевірки достовірності ручного введення даних. Досліджено проблему уніфікації та інтелектуалізації інформаційних систем залізничного транспорту в Україні засобами онтологічного забезпечення. Розроблено модульну онтологію, яка дозволяє інтегрувати інформаційні системи залізничних підсистем: станційну, вагонну, поїзну, відправочну.

У четвертому розділі доцільність запропонованого підходу експериментально підтверджена на прикладі ділянок залізниці «Маріуполь Порт-Волноваха» України, «Plovdiv-Burgas» Болгарії та «Coswig-ESig G» Німеччини та відповідного нормативного забезпечення. Онтологія може бути застосована як до високошвидкісних і звичайних залізниць (наприклад до 160 км/год), а також в подальшому інтегрована з поїзною моделлю. Оцінка онтології була проведена експертним шляхом і за допомогою застосунку OntOlogy Pitfall Scanner.

Ключові слова: онтологія, залізниця, OWL, база знань, база даних, конструктивно-продукційне моделювання, інформаційна система, нормативно-правове забезпечення, інструкція, семантичний аналіз, анотування текстів, таблиця.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА (LIST OF PUBLICATIONS)

Наукові праці, в яких опубліковані основні наукові результати дисертації:

Стаття у закордонному виданні включеному до міжнародних наукометричних баз (МНБД) Scopus та належить до III квартилю (Q3):

1) Shynkarenko, V., **Zhuchyi, L.**, & Ivanov, O. Ontology-Based Semantic Checking of Data in Railway Infrastructure Information Systems. Foundations of Computing and Decision Sciences, 47(3), 291-319. DOI: https://doi.org/10.2478/fcds-2022-0016

Публікації в наукових виданнях, включених до переліку наукових фахових видань України, затверджених МОН України:

2) **Zhuchyi, L. I.** (2022). Ontological Support for Harmonization and Integration of Ukrzaliznytsia Information Systems Data. Science and Transport Progress, (1 (97)), 32-49. DOI: https://doi.org/10.15802/stp2022/265335
3) Shynkarenko, V., & **Zhuchyi, L.** (2022). Ontological suitability analysis of railway tracks for high-speed traffic. Computer systems and information technologies, (3), 11-21. DOI: https://doi.org/10.31891/csit-2022-3-2
4) Shynkarenko, V., & **Zhuchyi, L.** (2022) Constructive-synthesizing modeling of ontological document management support for the railway train speed restrictions. Science and Transport Progress, (2 (98)), 59-68. DOI: https://doi.org/10.15802/stp2022/268001

Матеріали наукових конференцій, що індексуються в включені до міжнародних наукометричних баз (МНБД) Scopus:

5) Shynkarenko V., **Zhuchyi L.**, Ivanov O. Conceptualization of the Tabular Representation of Knowledge. International Scientific and Technical Conference on Computer Sciences and Information Technologies. Vol. 2: 16th IEEE International Conference on Computer Science and Information Technologies (CSIt2021), Lviv, 22–25 September 2021. P. 248–251. DOI: 10.1109/CSIT52700.2021.9648761. проіндексовано в SCOPUS

6) Shynkarenko V., **Zhuchyi L.** Ontological harmonization of railway transport information systems. CEUR Workshop Proceedings. 2021. Vol. 2870: 5th International Conference on Computational Linguistics and Intelligent Systems. Vol. I: Main Conference, COLINS 2021, 22–23 April 2021. P. 541–554. проіндексовано в SCOPUS
7) Shynkarenko, V., & **Zhuchyi, L.** (2022). Semantic Checking of Different Type Information Sources About Permitted Speeds in Railway Transport. CEUR Workshop Proceedings. 2022. Vol.: International Conference on Computational Linguistics and Intelligent Systems. Vol.: Main Conference, COLINS2022, 2022. проіндексовано в SCOPUS

Матеріали наукових конференцій:

8) Shynkarenko, V. I., & **Zhuchyi, L. I.** (2022). Ontological support of metallurgical and machine-building technological processes. Informacijni tehnologiї v metalurgiї ta mashinobuduvanni (pp. 178-180). Дніпро
9) **Жучий, Л. І.** (2020). Розвиток онтологій виробництва та транспорту. Інформаційні технології в металургії та машинобудуванні (с. 152-157). Дніпро
10) **Жучий, Л. І.** (2018). Концептуальне та онтологічне моделювання залізничного транспорту. Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті (с. 90). Дніпро
11) **Жучий, Л. И.** (2019). Способы представления онтологий железнодорожного транспорта. Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті (р. 79). Дніпро
12) Shynkarenko, V., & **Zhuchyi, L.** (2021). Semantic checking of train speed limit warnings. Modern information and communication technologies on a transport, in industry and education (p. 142). Дніпро
13) **Жучий, Л. И.**, & Шинкаренко, В.И. (2020). Сравнительный анализ предметно-ориентированных языков. Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті (р. 84). Дніпро

14) **Жучий, Л. И.**, & Шинкаренко, В.И. (2020). Формализация глоссария онтологическими средствами. Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті (с. 83). Дніпро

15) **Жучий, Л. І.**, Шинкаренко, В.І., & Іванов, О.П. (2020). Аналіз розвитку онтологій залізничного транспорту. Проблеми та перспективи розвитку залізничного транспорту (с. 377). Дніпро

16) **Zhuchyi, L. I.** Ontologies in the transportation context (2022). Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті. (с. 115) Дніпро

ABSTRACT

Zhuchyi L. I. Integration and harmonization of railway transport information systems data by ontological means – Qualifying scientific work as a manuscript.

Thesis submitted for obtaining the Doctor of Philosophy degree in Information technology, speciality 122 – Computer Science – Ukrainian State University of Science and Technologies, Dnipro, 2022.

The dissertation is devoted to issues of feasibility demonstration and development of methods of integration of siloed railway information systems and relevant regulations, improvement of checking their consistency.

In the dissertation work, new scientifically based theoretical and experimental results were obtained, which in the aggregate are essential for solving the actual scientific and technical problem of railway transport data processing and checking their consistency.

The scientific novelty of the obtained results.

At work for the first time:

1) the ontology of various types of sources was conceptualized and formalized, which allows for combining and harmonizing the tables presented in various information and software environments. Unlike others, it takes into account the specific tabular structure of data and is based on the gradual generalization of relationships between data elements;
2) the conceptualization of the prototype of the ontological support of the railway track and its integration with the train, wagon and dispatch models of the United Automated Management System of Freight Transportation of Ukrzalisnytsya (AMS FT UZ-U). Unlike others, it takes into account the formalized regulations;
3) the procedure for forming railway domain ontologies is formalized through constructive-synthesizing modelling. Unlike others, it takes into account the integration of siloed data sources.

Improved:

4) data processing technologies and information support processes: connection track passport; permitted railway train speeds on public railway tracks based on the corresponding ontological support;
5) methods of constructive-synthesizing modelling in the part of developing chains of interconnected constructors and their application in the railway domain.

Received further development:

6) means of semantic annotation of regulations. For this, an annotation scheme based on railway ontology is presented.

The practical value of the obtained results is in the development of a prototype based on ontologies, which provides a formalized representation of the railway technological processes, which allows for checking the consistency of the data of information systems with each other and with regulations.

Now on Ukrainian railways, the centralized system AMS FT UZ-U is used, which unites the main operational subsystems of Ukrzaliznytsya. Since the AMS FT UZ-U was created by combining and augmenting individual subsystems, it still suffers from fragmentation, which is a certain problem. In addition, there was a need for a certain decentralization because of the reorganization of Ukrzaliznytsya and the creation of several companies on its basis and cooperation with information systems of foreign countries, other carriers and some customers.

In the first section, ontological developments in the railway and other domains are analyzed. It has been established that the European Union railway transport ontologies are used for the integration of infrastructure description data, train timetables and others. At the same time, not enough attention is paid to the transportation process regulations. There are software tools for text annotation, table knowledge extraction and ontology development, but they are not used to support the Ukrainian railway transport transportation process. It was determined that the actual task is the annotation of regulatory documentation to establish a connection between ontology and the regulation texts.

The feasibility of using ontological tools in railway transport is justified for: formalization of regulatory support; data transformation and integration; checking the consistency of information systems data and regulations.

In the second section, author presents the developed methods for:

1) conceptualization and of knowledge tabular representation;

2) formation of the railway infrastructure ontology;

3) integration of railway infrastructure ontology prototype with:

   a) ontology of sources;

   b) ontology of railway train, wagon and dispatch models;

4) constructive-synthesizing modelling of ontological support of document flow.

The ontological elements of tabular knowledge representation and their relations are formalized in the form of a formal language graphic notation. A method of forming complex concepts is suggested. The conceptualization basis is deep detailing of relations between different concepts.

The basis of the concept is the application of the order relation, the variability of which makes it possible to model different real objects of different natures.

A method of semantic checking based on a model of multi-level concretization and integration of ontologies of sources and railway infrastructure has been developed. Mechanisms of implementation of ontologies and their integration are demonstrated in examples

In the third section, the means of ontological support for the connection track passport and permitted speeds of public railway tracks have been developed.

The tabular presentation of data and railway transport regulation rules for the connection track passport and speed restrictions has been formalized by ontological means and tools of «data wrangling» and «data extraction». This eliminates part of the time-consuming process of ontology population.

Means of semantic control of data from various sources on permitted speeds on operational public railway track elements have been developed. To formalize the restrictions of «Technical exploitation Rules of Ukrainian railways» the method of semantic annotation is used. Ontology formation is based on the composition of relations.

A modular ontology has been developed for the integration of data on railway switches and tracks, orders that establish permitted speeds on the railway infrastructure elements, «Technical exploitation Rules of Ukrainian railways» and «Building codes». This approach provides a connection between natural language instructions, information systems and ontologies of the railway train speed restrictions.

The railway track defect ontology combines information from the following sources: databases, station drawings and natural language regulations. Semantic checking is carried out with the help of the railway track maintenance formalized, track work and automated issuance of speed restriction warnings regulations. Application of the developed ontology makes it possible to increase the safety of railway train traffic by checking compliance with the speed regime and railway track defects, timely removal of restrictions and checking the validity of manual data entry.

The problem of unification and intellectualization of information systems of railway transport in Ukraine utilizing ontological support has been investigated. A modular ontology has been developed that allows for the integration of information systems of railway subsystems: station, wagon, train, and dispatch.

In the fourth section, the feasibility of the proposed approach is experimentally confirmed on the example of the sections of the railway «Mariupol Port-Volnovakha» of Ukraine, «Plovdiv-Burgas» of Bulgaria and «Coswig-ESig G» of Germany and the corresponding regulations. The ontology can be applied to both high-speed and conventional railways (for example, up to 160 km/h), as well as further integrated with the railway train model. The evaluation of the ontology was carried out by a domain expert and by the Ontology Pitfall Scanner.

Keywords: ontology, railway, OWL, knowledge base, database, constructive-synthesizing modelling, information system, regulations, instruction, semantic analysis, annotation of texts, table.

СПИСОК АБРЕВІАТУР ТА УМОВНИХ ПОЗНАЧЕНЬ

ВСП, ВЗП – Відомості Стрілочних Переводів и Залізничних Колій відповідно

КОЗК, КОСП – Книги Огляду Залізничних Колій и Стрілочних Переводів відповідно

ФВМПОШ, КЗПОШ – Форма Видачі Машиністу Поїзда и Книга Запису Попереджень про Обмеження Швидкості відповідно

МВАВВП – Методичні Вказівки з Автоматизованої Видачі і Відміни Попереджень

ІРП – Інструкція з Руху Поїздів

ПТЕ – Правила Технічної Експлуатації залізниць

ІПУЗК – Інструкція з Поточного Утримання Залізничних Колій

ІЗБВКР – Інструкція із Забезпечення Безпеки при Виконанні Колійних Робіт

RV – Resources Vocabulary

TS – Table Structure

TSCR, TTCR – Table Software та Table Type Classification Rules

SVR, TSTCR – Structure Validation та Table Station Classification Rules

SWM, SVM – Structure Wrangling та Structure Validation Rules

DWM – Data Wrangling Rules

RIV – Railway Infrastructure Vocabulary

IEL – Infrastructure Elements Lists

IRB – Inspections Records Books

SLWBF – Speed Limits Warnings Books and Forms

MR – Matching Rules

RRR – Railway Regulations Rules

DIM, DCM – Data Integration and Data Checking Models respectively

ISA – Intelligent Speed Adaptation

ЗМІСТ

ВСТУП

Актуальність роботи. Актуальною проблемою перевізного процесу залізниць України є роз'єднаність баз даних та інформаційного забезпечення. Залізниця включає в себе безліч підсистем, одною з яких є підсистема колійного господарства. Роз'єднаність не дозволяє узгодити дані різних підсистем і може бути вирішена шляхом розробки онтологічного забезпечення залізничного транспорту.

Ручні ланцюги передачі та пошуку даних викликають проблеми з надійністю даних. Рух поїздів вимагає високого рівня безпеки. При передачі даних телефонограмами дотримуватися узгодженості даних інформаційних систем і інструкцій досить складно. Це сприяє виникненню помилок, що виявляються фахівцями відповідних підсистем. Жодні інструкції залізничного транспорту України не формалізовані.

У процесі транспортування важливо виявити невідповідності даних інформаційних систем та інструкцій. Для цього їх необхідно представити в єдиному вигляді логічних аксіом та формалізувати текстові інструкції.

У зв'язку з цим актуальними є дослідження з формалізації, інтеграції та анотування нормативного забезпечення та інтеграція з даними інформаційних систем залізниць України. Це сприяє підвищенню безпеки руху поїздів.

Автоматизація формування онтологій дозволяє полегшити трудомісткий процес розробки онтологій залізничного домену, де необхідно враховувати дані інформаційних систем, так і нормативне забезпечення для перевірки їх узгодженості.

Дослідження процесів:

1) розробки онтологій представлені в роботах N. F. Noy, D. L. McGuinness, M. Fernández-López, A. Gómez-Pérez, M. C. Suárez-Figueroa, V. Presutti, E. Daga, A. Gangemi, E. Blomqvist, В. Литвина, Л. Глоби, Д. Досина, Ю. Рогушиної, В. Скалозуба, В. Ільмана, В. Шинкаренка та ін.;
2) інтеграції даних інформаційних систем онтологічними засобами представлені в роботах J. Tutcher, R. Lewis, M. Katsumi, M. Fox, D. Corsar, L. Zhao та ін.;

3) формалізації нормативних документів онтологічними засобами представлені в роботах A. Jovic, P. Pauwels, D. Van Deursen, G. Guizzardi, S. Zhang та ін..;
4) анотування нормативних документів онтологічними засобами представлені в роботах A. Gangemi, Д. И. Муромцев, T. Diamantopoulos, M. Roth та ін.;
5) перевірки узгодженості даних інформаційних систем між собою та з нормативними документами онтологічними засобами представлені у роботах A. Gangemi, S. Peroni, X. Fiorentini та ін..

В області транспортних онтологій вони зазвичай базуються на інтеграції даних, а не на формалізації нормативних документів. Такі онтології дозволяють виконувати перевірку узгодженості засобами здорового глузду і не враховують технологію роботи залізниць.

На відміну від цього, у дисертації перевіряється узгодженість та інтеграція даних інформаційних систем, заснована на формалізації нормативних документів залізничного транспорту.

Зв'язок роботи з науковими програмами, темами, планами. Дисертаційна робота виконана відповідно до стратегії АТ «Укрзалізниця» на 2019–2023 роки, зокрема за напрямами інтеграції та стандартизації інформаційних систем, підвищення достовірності даних та автоматизації бізнес-процесів, а також директиви Євросоюзу 2008/57/ЕС [55].

Наукові дослідження, викладені в дисертації, виконані згідно з напрямом наукової роботи кафедри «Комп'ютерні інформаційні технології» Українського державного університету науки і технологій, а саме є частиною науково-дослідних робіт «Інструментальна підтримка систем обробки природно-мовних документів» (2022 р. № держреєстрації 0122U002086) та «Моделювання в задачах розробки програмного забезпечення» (2021 р. № держреєстрації 0121U109167), у яких дисертант приймала участь у якості виконавця.

Мета та задачі дослідження. Метою дисертації є розробка методів інтеграції роз'єднаних інформаційних систем залізничного транспорту та відповідної нормативно-правової документації, підвищення контролю їх узгодженості.

Для досягнення поставленої мети поставлені і вирішені наступні завдання:

1) виконати аналіз розроблених онтологій транспортного і суміжних доменів; існуючих інформаційних систем залізниць Євросоюзу та України;
2) розробити конструктивно-продукційну модель формалізації процесу розробки онтологій;
3) розробити модульну онтологічну модель інформаційних систем АСК ВП УЗ-Є.
4) розробити метод інтеграції таблиць баз даних різних інформаційних систем та нормативної документації залізниць;
5) розробити метод перевірки узгодженості даних інформаційних систем та нормативної документації онтологічними засобами;
6) оцінити адекватність та узгодженість розробленої моделі:
   a) програмним шляхом із використанням середовища розробки Protégé;
   b) експериментальним шляхом з використанням даних інформаційних систем Укрзалізниці та країн Європи.

Об’єктом дослідження є процеси глобалізації інформаційного забезпечення українських та європейських залізниць.

Предметом дослідження є моделі, методи та програмні засоби розробки онтології інтеграції та узгодження даних інформаційних систем залізничного транспорту.

Методи дослідження. Для вирішення поставлених задач було використано: онтологічне моделювання на основі дескрипційної логіки (description logic), об'єктно-орієнтований аналіз, теорія множин, конструктивно-продукційне моделювання, методи системного аналізу (модель input-process-output).

Наукова новизна отриманих результатів.

В роботі вперше:

1) виконано концептуалізацію і формалізацію онтології різних типів джерел, що дозволяє об'єднувати та узгоджувати таблиці, представлені в різних інформаційних та програмних середовищах. На відміну від інших, вона

враховує специфічну табличну структуру даних та заснована на поступовому узагальненні зв’язків між елементами даних;

2) виконано концептуалізацію прототипу онтологічного забезпечення залізничної колії та його інтеграцію з поїзною, вагонною і відправочною моделями АСК ВП УЗ-Є. На відміну від інших, він враховує формалізовані положення нормативно-правових актів;
3) формалізовано процедуру формування онтологій залізничного домену засобами конструктивно-продукційного моделювання. На відміну від інших, вона враховує інтеграцію роз’єднаних джерел даних.

Поліпшено:

4) технології обробки даних і процесів інформаційного супроводження: паспорту під'їзної колії; допустимих швидкостей руху поїздів на залізничних коліях загального користування на основі відповідного онтологічного забезпечення;
5) методи конструктивно-продукційного моделювання у частині розробки ланцюгів взаємопов’язаних конструкторів та їх застосування у залізничному домені.

Отримали подальший розвиток:

6) засоби семантичного анотування нормативних документів. Для цього представлена схема анотування на основі онтології залізничного домену.

Практичне значення отриманих результатів. Розроблено прототип інформаційної системи на основі онтологій, який дає формалізоване представлення технологічних процесів залізниці, що дозволяє перевірити узгодженість даних інформаційних систем між собою та з нормативною документацією.

Практичною частиною використання досягнутого наукового результату є закладення основ розвитку онтологічного забезпечення технологічних процесів залізничного транспорту. Підтверджено можливість досягнення поставлених цілей онтологічними засобами на залізничному транспорті.

Практичне значення підтверджується впровадженням в:

1) Українському державному університеті науки і технологій:

a) при підготовці магістрів за спеціальністю 121 «Інженерія програмного забезпечення» при викладанні дисциплін «Інформаційні системи на залізничному транспорті» та «Інтернет-технології»;

b) при підготовці аспірантів за спеціальністю 122 «Комп'ютерні науки» – «Ефективність інформаційних систем та комп'ютерних технологій»;

2) компанії railML.org® Дрезден, Німеччина:

a) частина проекту з автоматизованої сертифікації, де використовуються методи перевірки узгодження даних;

b) частина проекту з перетворення даних, де використовуються методи інтеграції схем онтологій;

c) рекомендації з розробки онтології railML.

Особистий внесок здобувача. Основні результати дисертаційної роботи опубліковано в роботах у співавторстві:

[179] – розробка моделі табличного представлення знань;

[178] – об’єднання моделей АСК ВП УЗ-Є онтологічними засобами;

[176, 177] – розробка моделі допустимих швидкостей залізничної колії, семантичне анотування тексту Правил технічної експлуатації залізниць України;

[175, 180, 223 - 221] – розробка моделей багаторівневої конкретизації та під'їзної колії. Семантична анотація таблиць її інформаційного забезпечення.

[173] – конструктивно-продукційне моделювання онтологічного забезпечення обмеження швидкості руху поїздів.

[174] – експериментальні дослідження та оцінка застосовності розробленого методу до даних Укрзалізниці та залізниць країн Європи.

В одноосібній статті [215] та тезисах [216, 218-220] – огляд існуючих онтологічних розробок на транспорті.

Апробація результатів дисертації. Результати дисертаційної роботи представлено на всеукраїнських та міжнародних науково-практичних конференціях: 42nd railML conference (Oslo, 2022), Інформаційні технології в металургії та машинобудуванні (Дніпро, 2020, 2022), Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті (Дніпро, 2019, 2021, 2022),

Проблеми та перспективи розвитку залізничного транспорту (Дніпро, 2022), International Conference on Computer Sciences and Information Technologies (Львів, 2021), International Conference on Computational Linguistics and Intelligent Systems (Харків, Gliwice (Польща), 2021, 2022).

Публікації. Результати дисертаційної роботи опубліковано в семи наукових працях.

Стаття у закордонному виданні включеному до міжнародних наукометричних баз (МНБД) Scopus та Web of Science належить до ІІІ квартилю (Q3) – 1.

У фахових та рекомендованих Міністерством освіти і науки (МОН) України для публікації результатів дисертацій – 3.

Матеріали міжнародних конференцій, що індексуються МНМБ Scopus – 3.

У тезах доповідей міжнародних та всеукраїнських конференцій – 8.

Структура та обсяг дисертації. Дисертаційна робота складається із вступу, 4 розділів, висновків, списку використаних джерел і додатків. Загальний обсяг дисертації становить 189 сторінки, в тому числі 104 сторінки основної частини, 45 рисунків, 20 таблиць, 2 додатки на 52 сторінках та список використаних джерел із 227 найменувань на 23 сторінках.

РОЗДІЛ 1
АНАЛІЗ ПІДХОДІВ ДО РОЗРОБКИ ОНТОЛОГІЙ ТА ЇХ ЗАСТОСУВАННЯ НА ЗАЛІЗНИЧНОМУ ТРАСПОРТІ

Стратегія розвитку Укрзалізниці включає наступні напрямки: інтеграція та стандартизація інформаційних систем, підвищення надійності даних та автоматизація бізнес-процесів. Одним з напрямків вирішення поставлених задач є інтеграція залізничних інформаційних систем онтологічними засобами без зміни їх структури.

Далі розглянуті наступні питання:

1) аналіз існуючих онтологічних розробок на транспорті;
2) визначення підходів до застосування розробок суміжних доменів до завдань розвитку Укрзалізниці.

Виконана систематизація онтологічних розробок за:

1) призначенням їх неонтологічних ресурсів для розробки схеми онтології (нормативних документів, стандартів і навчальних матеріалів);
2) застосуванням форматів вихідних даних (db, csv);
3) рівнем інтеграції даних;
4) метою програмного забезпечення, заснованого на онтології.

Використовуються такі методи системного аналізу, як модель input-process-output (IPO). Метод систем IPO використовується наприклад для представлення літературного огляду про Вікіпедію [95]. Він групує статті за критерієм опису вхідних і вихідних даних Вікіпедії. Транспортним онтологіям присвячений огляд [97].

1.1 Інформаційні системи Укрзалізниці

Згідно з [137] Інформаційне забезпечення Укрзалізниці включає наступні системи:

1) АСК ВП УЗ-Є – Єдина Автоматизована Система Керування Вантажними Перевезеннями;
2) АСК ПП УЗ – Автоматизована Система Керування Пасажирськими Перевезеннями;
3) АСМК – Автоматизована Система Управління Ресурсами Власності;

4) АСБО ФОБОС – Автоматизована система обліку структурного підрозділу залізниці;
5) АРМ ПВ – Автоматизована робоче місце прийомоздавальника;
6) АРМ ДНЦ – Автоматизоване робоче місце поїзного диспетчера;
7) АС МЕСПЛАН – Автоматизована система документообігу замовлень на перевезення вантажів та формування планів;
8) AS Клієнт – Автоматизована система формування електронних перевізних документів;
9) АС ЕРПВ – Автоматизована система управління експлуатацією та ремонтом пасажирських вагонів.

Згідно висновків [178] бази даних є в якійсь мірі роз'єднаними і фрагментованими.

1.2 Основні класифікації онтологій

Існують різні визначення і класифікації онтологій. По Груберу, онтологія – це явна специфікація концептуалізації [74].

Згідно з [36], «онтологія визначає загально вживані слова та концепти (значення), зо використовуються для описання та представлення області знань, і таким чином стандартизує значення. Онтологія включає наступне:

1) класи (загальні речі) багатьох доменів, що представляють інтерес;
2) екземпляри (конкретні речі);
3) відношення між цими речами;
4) властивості (і значення властивостей) цих речей;
5) функції та процеси, що включають ці речі;
6) обмеження та правила, що включають ці речі».

Онтології класифікуються як «онтології вищого рівня» (top-level ontologies), «онтології предметної області» (domain ontologies), «проблемна онтологія» (task ontologies) та «онтологія застосунку» (application ontologies) в [76], «загальні онтології» (generic ontologies) і «конкретні онтології» (specific ontologies) в [92], «вища онтологія» (upper ontology), «середня онтологія» (middle ontology), «нижча

онтологія» (lower ontology), «найнижча онтологія» (lowest ontology) в [36], багаторівневі [150] та такі, що засновані на нечіткій логіці [71]. «Онтології вищого рівня» і «онтології предметної області» пов'язані відношенням спеціалізації [76].

Інші класифікації поділяють онтології на «важкі» (heavyweight) і «легкі» (lightweight). «Легка» онтологія – це онтологія без виразної семантики, словник, наприклад Good relations [209]. Легкі онтології можуть бути тезаурусами, наприклад AGROVOC [76] (заснована на SKOS [184]), task thesaurus [164] і NCI Thesaurus [126].

«Важка онтологія – це легка онтологія … збагачена аксіомами» [63]. Наприклад, в Cell ontology [115] и Collections Ontology [25] правила представлені мовою OWL як логічні визначення і SWRL відповідно.

1.3 Основні методи розробки онтологічного забезпечення

Використовується декілька взаємодоповнюючих предметно-незалежних методів розробки онтологій: 101 [129], methontology [59], neon [187], extreme [152], метод генеруючих відображень [181, 182], розробка модульних онтологій [82].

Методи розрізняються за ступенем деталізації опису. Methontology [59] містить загальні напрямки: концептуалізація, формалізація, реалізація і обслуговування. В 101 [129] описано більш детальний (дещо видозмінений) універсальний процес розробки діаграми онтології, що відповідає визначенню онтології [36]:

1) «перелічити важливі терміни в онтології;

2) визначити класи та їх ієрархію;

3) визначити властивості класів – слоти;

4) визначити обмеження;

5) визначити правила;

6) створити екземпляри».

Концептуалізація онтології може бути виконана з використанням неонтологічних ресурсів [187] або патернів [152] для уніфікації розробки неспеціалістами. Модульна розробка онтології полегшує підтримку та повторне використання онтологій [82]. Прикладом модуля є Error ontology [147], що використовується в Publishing Workflow Ontology [65] і Collections ontology [25]. Де

Collections ontology [25] є модулем онтології RaCoOn [196]. У той же час модульність є одним з принципів розвитку онтології RaCoOn [196], де автори крім колекцій виділяють такі модулі, як «Asset Management», «Rail Core» та інші.

Особливістю розробки ієрархії класів онтології є необхідність виключення полі-ієрархій з огляду на їх складне обслуговування. [162]. Замість сильних під-класифікаційних відношень розробляються логічні визначення, як у Cell ontology [115].

По завершенні розробки онтології її тестування виконується за запитами SPARQL [152]. У базі знань, заснованій на транспортній онтології, задаються пошукові запити по всіх залізничних станціях [196], кількості поїздок з початком в зоні в певний проміжок часу [98], обмеження швидкості ділянки дороги [213].

1.4 Формалізація нормативно-правової документації онтологічними засобами

Стратегія Укрзалізниці [186] передбачає стандартизацію інформаційних систем, невід'ємною частиною яких є нормативно-правове забезпечення залізничного транспорту. Наприклад, вже використовуються на УЗ стандартні інструменти, такі як XML-схема для накладної ЦИМ/СМГС і проектування вагонів в SolidWorks, який підтримує стандарт STandard for Exchange of Product Model Data (STEP). Планується запровадити такі європейські стандарти, як Infrastructure Register (RINF) [56], Network Statement.

Слід зазначити, що на даний момент залізничні інструкції не представлені стандартною мовою, заснованою на логіці. Формалізація інструкцій – це процес перетворення текстів інструкцій в класи, відношення, обмеження і правила «якщо-то» (Таблиця 1.1). Вхідні дані – це тексти, а вихідні – схема онтології (або екземпляри, як у [40]). Згідно з [187] процес перетворення неонтологічного ресурсу полягає в наступному:

1) «знайти підходящий патерн для ре-інжинірингу не-онтологічного ресурсу:
   a) використати патерн для трансформації;
   b) виконати ситуативне перетворення;
2) ручне покращення».

Де патерни ontologydesignpatterns.org призначені для перетворення тезаурусів і схем класифікації. На кроці «ситуативного перетворення» може застосовуватись універсальний процес [129].

Неонтологічними інструктивними ресурсами для концептуалізації є нормативні документи [146], навчальні матеріали [62] та стандарти [144].

Стандарти представлені у вигляді XML-схем railML [156, 157], моделей UML Core Product Model [60], RailTopoModel [159] та текстів [144].

Залежно від типу конструкцій, представлених в стандарті, вони можуть бути основою для класів, відношень тощо, наприклад, класифікаційні схеми є основою ієрархії класів онтології, а в текстових інструкціях можуть бути конструкції, які є основою для правил «якщо, то». Існують також онтології високого рівня абстракції для опису будь-яких нормативних документів, таких як LKIF [85]. З них на транспорті здійснюється формалізація стандартів в [15, 196, 197].

Network statement є стандартним текстовим документом, який використовується для опису залізничної інфраструктури в Європі. Інструкція по його складанню містить перелік глав Network statement та їх призначення. Онтологія Network Statement Checker ontology [197] дозволяє будувати залізничні маршрути інфраструктурою з різними характеристиками, різних країн. Стандарт є основою класів онтології (Рисунок 1.1).

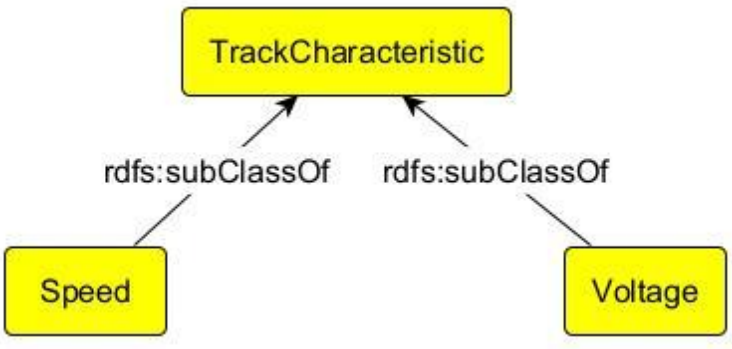


Рисунок 1.1 Класи онтології Network Statement Checker заснованої на текстовому стандарті

RailTopoModel – це стандартна модель UML для представлення залізничної інфраструктури на основі connexity graph [159] і є основою Rail Topology Ontology

[15]. UML має конструкції для представлення обмежень потужності, які лежать в основі описів класів онтології OWL (Рисунок 1.2). Наприклад, RailTopoModel містить клас PositionedRelation для представлення напрямку руху по залізничній колії і має обмеження на кількість (потужність) вузлів, пов'язаних з ним, до двох.

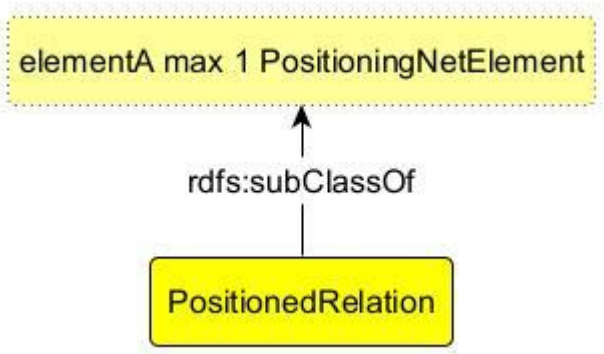


Рисунок 1.2 Обмеження онтології Rail Topology Ontology, на основі стандартів формату UML

Таблиця 1.1 Формалізація інструктивних матеріалів в існуючих онтологічних розробках

Вхідні дані	Вихідні дані
OSHA, 1926.706 Requirements for masonry construction	The Construction Safety Ontology [212]
IEEE Standard Classification for Software Anomalies	Ontology of Software Defects, Errors and Failures [48]
ISO/IEC 11404:2007 – General-Purpose Datatypes	Generic ontology of datatypes [144]
European EN 12354-3:2000 standard Звукоізоляція зовнішнього шуму	The IFC (Industry Foundation Classes) ontology, The construction element ontology [146]
ISO-10303 STandard for Exchange of Product Model Data	OntoSTEP [103]
Network Statement [158]	Network Statement Checker Ontology [197]
ESC рекомендації з діагностики та лікування гострої та хронічної серцевої недостатності	Heart Failure Ontology [94]
«Materials science and engineering: an introduction»	Functionally Graded Materials Ontology [62]
Довідник з теорії ймовірностей та математичної статистики	OntoMathPro [53]
Правила технічної експлуатації залізниць України [105]	Base ontological model of АСК ВП УЗ-Є [178]
The Core Product Model, The Open Assembly Model	Ontology for Assembly Representation [60]
railML [156]	Rail Core Ontology [196]
RailTopoModel [159]	Rail Topology Ontology [1515]

Процедурні знання (наприклад, необхідність узгодження обмеження швидкості попередження, що видається машиністу) містяться в текстових інструкціях і формалізуються як логічні визначення, композиції відношень і правила SWRL, як наприклад, в [212]. Онтологія The Construction Safety Ontology включає в себе правило (Рисунок 1.3) для перевірки узгодженості будівельних проектів і нормативних документів.

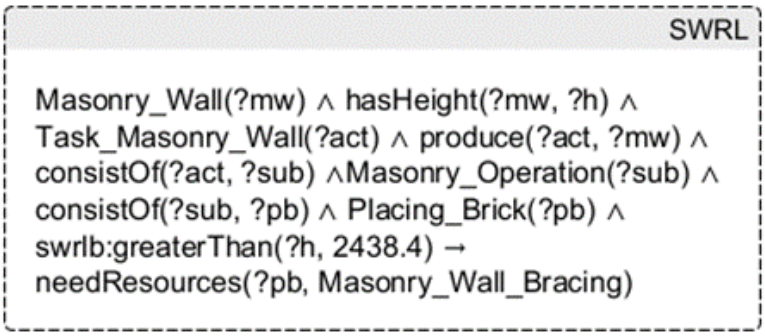


Рисунок 1.3 Правило онтології The Construction Safety Ontology на основі текстового стандарту

Онтологічні засоби на транспорті представляють стандарти залізничної інфраструктури в [15, 196, 197]. За допомогою реляційних баз даних АСК ВП УЗ-Є Укрзалізниці формалізовано перевізний процес. З їх допомогою складно представити правила нормативного забезпечення, наприклад, «Інструкції з руху поїздів» [203]. У суміжних доменах стандарти, що представляють процедурні знання, формалізовані онтологічними засобами [146, 212]. Можливою подальшою роботою на транспорті є формалізація інструкцій онтологічними засобами, як в [178].

1.5 Видобування знань з табличного і текстового представлення даних перевізного процесу

Стратегія розвитку Укрзалізниці [186] передбачає автоматизацію бізнес-процесів. В Укрзалізниці вже є системи типу «АС Клієнт», де заповнюється електронна форма для оформлення перевізного документа. Існує ще багато технологічних процесів, які передбачають ручне видобування даних з різнорідних документів, наприклад, процес введення в експлуатацію під'їзної колії. Розрахунок

профілю виконується в Excel, креслення – в AutoCAD, характеристики колії можуть зберігатися в базах даних. Тягові розрахунки для розробки графіка руху поїздів також виконуються в Excel.

Процес видобування знань складається з трьох підпроцесів: семантичної анотації текстів (може включати лінгвістичний аналіз [163]), «перетворення даних» (data wrangling) таблиць і «наповнення онтології» (ontology population).

Семантичне анотування – це процес перетворення тексту в такий, який додатково містить поняття онтології [134] (Таблиця 1.2).

Таблиця 1.2 Перетворення тексту в існуючих онтологічних розробках

Дані	Програмне забезпечення для перетворення даних
Вимоги до програмного забезпечення [40]	Mate-tools [109]
Законодавство про боротьбу з відмиванням грошей [9]	GATE [35]

Процес виглядає наступним чином:

1) розробка схеми онтології;

2) перетворення концептів онтології в теги;

3) анотування тексту тегами;

4) експорт семантичних анотацій у форматі tsv.

Анотування виконується вручну і автоматизовано, в залежності від ступеня підготовки системи машинного навчання і необхідності отримання результату без помилок. Розроблено окремі програмні засоби [193] та плагіни Protégé [131]. Функціонал INCEpTION [101] включає засоби для анотування з використанням онтологій зв’язування сутностей (entity linking) і тегів методом присвоєння семантичних ролей (semantic role labeling).

В рамках проекту S-CASE (Scaffolding Scalable Software Services) виконується анотування вимог до програмного забезпечення [40], для того щоб перевірити їх узгодженість. Розроблені інструменти, архітектура яких включає в себе модуль для «переведення» вимог в специфікації програмного забезпечення. Анотування виконується методами видобування заснованого на виділенні ознак (feature based extraction), синтаксичний аналіз на основі граматики залежностей (dependency parsing) і присвоєння семантичних ролей в mate tools і відбувається в

автоматизованому режимі. Трійки «Актор-Дія-Об’єкт» можуть бути позначені вручну або парсером відповідно до синтаксичних і семантичних властивостей слова (кількість слів між ними) і навчальних даних, а потім виправлені користувачем.

Ручне анотування активно використовується в задачах секвентування ДНК біологічного домену з двох причин: більша надійність ручних анотацій [89] і неможливість автоматично анотувати певні гени (наприклад, псевдогени) [8]. ДНК бактерії, що виробляє біопаливо, анотується вручну в [89], де для імен використовується International Protein Nomenclature розроблена National Center for Biotechnology Information, і виправлені помилки різних автоматичних систем анотування. Gene Ontology містить мільйони ручних і автоматичних анотацій [10], і є не просто словником, але містить логічні визначення класів [69], і була повторно використана у ряді онтологічних розробок.

Модульна онтологія Financial Industry Regulatory Ontology [9] розроблена для анотування фінансових нормативно-нормативних документів домену боротьби із відмиванням грошей для перевірки узгодженості бізнес-процесів. Анотування документів виконується автоматично з використанням машинного навчання для заповнення онтології екземплярами. Навчальні дані вручну анотуються в системі GATE. Програмний комплекс дозволяє виконувати запити SPARQL за текстовими документами для пошуку відповідних обмежень, зобов'язань, заборон тощо.

«Перетворення даних» (data wrangling) – це процес перетворення форматів даних, наприклад, з XLS в RDF (Таблиця 1.3). Перетворення може здійснюватися різними програмними засобами, такими як OpenRefine. Вхідні дані – це табличні дані і схема онтології, а вихідні – екземпляри онтології.

Процес виглядає наступним чином:

1) розробка схеми онтології;
2) імпорт файлів і даних онтології в «перетворювач даних» (data wrangler) (Cellfie [21], Ontop [18], OpenRefine, Karma [102])
3) анотування змісту таблиць класами і відношеннями онтології.
4) експорт даних у форматі RDF.

«Заповнення онтології» (ontology population) – це процес наповнення онтології даними з тексту [66], або таблиці [19]. При перетворення баз даних в екземпляри онтології наповнення відбувається автоматично за допомогою портів. Якщо є два файли: схема онтології та дані RDF, наповнення відбувається за допомогою конструкції owl:import у файл схеми або обох файлів в новий файл онтології.

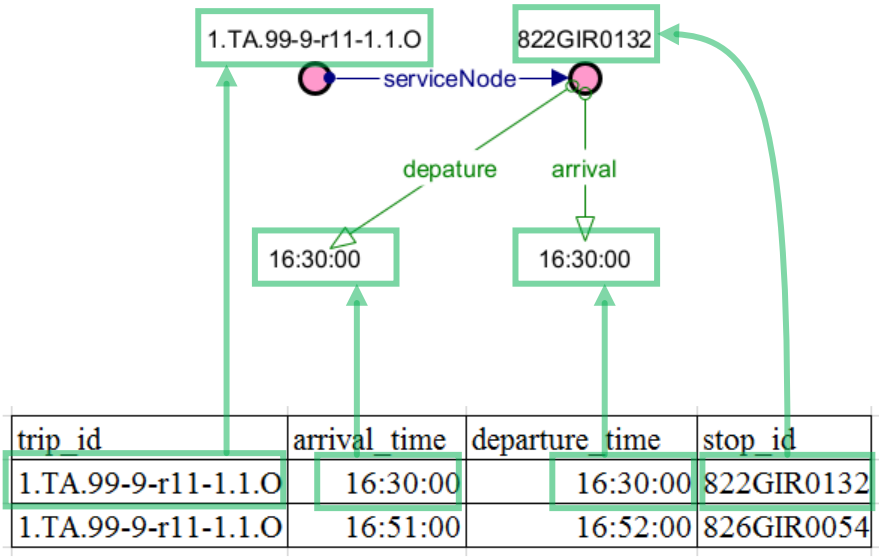
Залежно від формату, дані електронної таблиці Excel перетворюються за допомогою плагіна Protégé Cellfie [21] як у [49], баз даних – Protégé Ontop [18] як у [13], STEP – Protégé OntoSTEP [103].

Дані з таблиць креслень видобувають в Tabula [190] у файл CSV. Для перетворення таких даних в екземпляри онтології використовуються універсальні (такі, що розпізнають різні формати) програмні засоби, OpenRefine і Karma [102] як у [98].

Таблиця 1.3 Перетворення даних в існуючих онтологічних розробках

Дані	Програма перетворення даних
Таблиці https://covid19.who.int	Cellfie [21]
Association of Train Operating Companies (ATOC) розклад машиніста	OpenRefine
GTFS таблиці	Karma [102]
Smithsonian American Art Museum (SAAM) база даних	
OpenStreetMap, LIDAR дані, Yellow Pages та білі сторінки	
База даних Tata Steel	Ontop [18]
ArcGIS GFX дані	
Моделі STEP	OntoSTEP [103]

З них на транспорті для наповнення онтології RaCoOn [196] дані таблиці ATOC розкладу машиніста (working timetable) формату MCA (Common Interface File) перетворюються на екземпляри онтології. Такий формат характерний для Network Rail Integrated Train Planning System і з ним важко працювати, тому в [196] він перетворюється на реляційну базу даних перед імпортом у OpenRefine. Наприклад, екземпляри, які можна видобути з таблиці General Transit Feed Specification «stop times» сайту transportforireland Ірландських залізниць (Рисунок 1.4). Ці джерела близькі до ATOC тим, що таблиця також містить маршрути з зупинками та часом прибуття/відправлення.



trip_id	arrival_time	departure_time	stop_id
1.TA.99-9-r11-1.1.O	16:30:00	16:30:00	822GIR0132
1.TA.99-9-r11-1.1.O	16:51:00	16:52:00	826GIR0054

Рисунок 1.4 Видобування знань з розкладу поїздів на основі онтології RaCoOn

Онтологічними засобами на транспорті перетворюються дані розкладу поїздів формату МСА в [196]. За допомогою «АС Клієнт» автоматизовано введення даних перевізних документів до Укрзалізниці. У суміжних доменах дані в форматах db, csv, xls, pdf видобуваються і перетворюються онтологічними засобами. Можлива подальша робота полягає в збільшенні кількості завдань, для яких здійснюється автоматизація видобування даних. Використання стандартів RDF дозволить перевірити узгодженість даних і нормативних документів і інтегрувати дані з іншими, які мають однакову схему онтології.

1.6 Інтеграція роз'єднаних інформаційних систем онтологічними засобами

Стратегія розвитку Укрзалізниці [186] передбачає інтеграцію інформаційного забезпечення. Укрзалізниця вже розробила систему АСК ВП УЗ-Є, де інтеграція здійснюється шляхом обміну електронними та телефонограмами, на основі реляційної бази даних. Інформаційне забезпечення залишається фрагментарним і не відповідає вимогам його розширюваності.

Інтеграція даних онтологічними засобами – це процес наповнення онтології даними з різних джерел для перевірки їх узгодженості (Таблиця 1.4). Вхідні дані – це різні джерела даних (не обов'язково неоднорідні), а на вихідні – логічний висновок про узгодженість онтології. Процес виглядає наступним чином:

1) розробка схеми онтології;

2) перетворення даних з кількох джерел за допомогою однієї схеми.

3) імпорт даних в онтологію за допомогою механізму owl:import, якщо схема та дані є окремими файлами, додавання екземплярів вручну, якщо розробка здійснюється в Protégé, або програмно, якщо використовується бібліотеки java owlapi [138], jena [93], бібліотека python owlready [139].

Інтеграція даних може бути виконана в рамках одного підприємства і декількох. Для інтеграції даних з декількох підприємств онтологія повинна включати достатній набір понять. Онтології, такі як Steel cold rolling ontology, містять лише поняття про метал, а онтології на кшталт The transport disruption ontology – поняття як про транспорт, так і про інциденти, що дозволяє інтегрувати два різних набори даних.

Таблиця 1.4 Інтеграція даних в існуючих онтологічних розробках

Дані	Онтологія предметної області
АТОС розклад машиніста, Вимірювання ударного навантаження на колесо	Rail Core Ontology [196]
Системи вимірювання ударного навантаження на колесо та детектор гарячого візка	Railway infrastructure ontology [105]
GTFS специфікація маршрутів транспортних засобів, (2) звіти про інциденти в метро та (3) дані про місцезнаходження транзитних транспортних засобів у реальному часі	iCity [98]
Звіти [про інциденти] від управління транспорту, автобусних служб TransXChange, пунктів доступу NaPTAN і NPTG до громадського транспорту	The Transport Disruption Ontology [30]
Доступність, інциденти та інфраструктура	Mobility and Accessibility Ontology (MAnto) [17]
Таблиці баз данихthe Roll, Roll Grinding, and Roll Storage tables	Steel Cold Rolling Ontology [13]
Дані музеїв	EUROPEANA [45]
The Statute Staple, The Down Survey, The Books of Survey and Distribution, Dictionary of Irish Biography та the Oxford Dictionary of National Biography	Онтології на основі The Dictionary of Irish Biography, The Oxford Dictionary of National Biography [122]
Дані GPS-датчиків і обмеження швидкості різних автомагістралей	Map Ontology, Control Ontology, Car Ontology [213]
The Stanford Tissue Microarray Database, The Gene Expression Omnibus	NCI Thesaurus [126]

З них інтеграція даних на транспорті здійснюється в [196, 105, 98, 17, 213], та відповідає цілям УЗ, таким як задоволення потреб осіб з інвалідністю [200], просування внутрішнього туризму [201]. Перетворення відкритих даних,

опублікованих УЗ [199] відповідно до положення «Про затвердження Положення про набори даних, що підлягають оприлюдненню у формі відкритих даних», в linked open data як [45] для транспорту, обґрунтовано в [149].

Розроблені онтології iCity трьох рівнів абстракції, а також кілька демонстраційних додатків, включаючи додатки для вивчення транспортних потоків, а також відображення доріг і пам'яток на карті. У першому випадку на онтологію відображаються GTFS специфікації маршрутів транспортних засобів, (2) звіти про транспортні події метро, та (3) дані про місцезнаходження транспортних засобів в реальному часі, а в другому – GFX дані Neighbourhood, Land Use, Land Cover, Point of Interest, Road Segment.

Інтеграція даних (Рисунок 1.5) виконана як в прикладі «Integration with ArcGIS» використання онтологій iCity [98]. В основі рисунка лежать онтології Public Transit Ontology и Location Ontology iCity [98] та набори даних vicroadsopendata «Bluetooth sites» і «Cycle routes». Інтеграція даних дозволяє робити запити SPARQL «як доїхати до обладнаного Bluetooth місця на велосипеді»

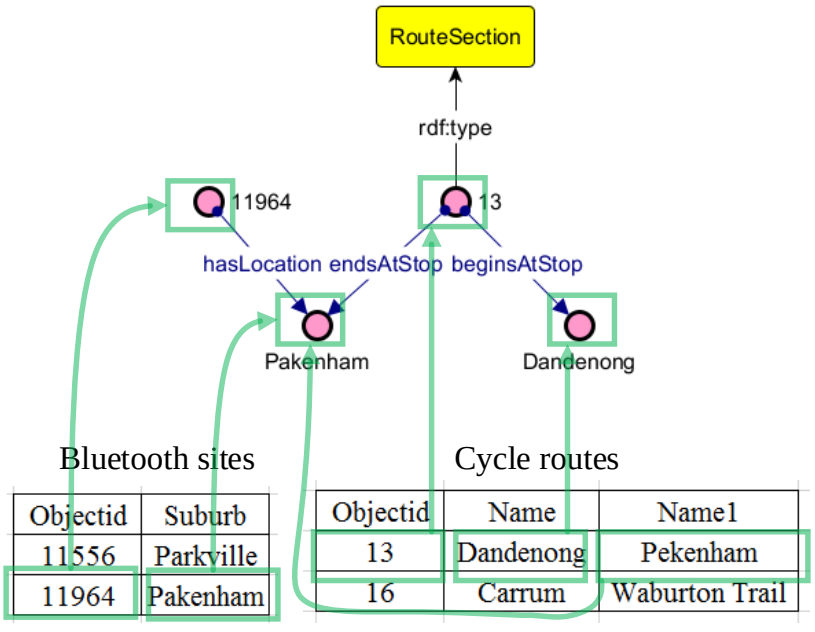


Рисунок 1.5 Інтеграція даних про маршрути та визначні пам'ятки на основі онтологій iCity

Дані ATOC Working Timetable і Wheel Impact Load Detector інтегровані з використанням онтології RaCoOn [196] щоб після їх інтеграції з OpenStreetMap,

визначити місце розташування несправних поїздів на карті. З допомогою RaCoOn інтегрують дані британських залізниць.

Виконується інтеграція WILM і HABD для визначення найактуальніших ремонтів поїздів [105].

В [213] швидкість автомобіля і дані карти про обмеження швидкості інтегруються онтологічними засобами з метою виявлення перевищення швидкості.

Інтеграція інформаційних систем може бути виконана на рівні даних, як у [213] або на рівні онтології.

Інтеграція онтологій – це процес зв'язування онтологій (Таблиця 1.5). Вхідними та вихідними даними є онтології. Процес полягає в наступному:

1) розробка схеми онтології;
2) розробка «мостової онтології» (bridging ontology) (аксіом) для зв'язування з деякою іншою онтологією;
3) імпорт розробленої онтології, мостової онтології» та деякої іншої онтології в нову онтологію.

Таблиця 1.5 Інтеграція онтологій в існуючих онтологічних розробках

Інші релевантні онтології	Онтологія предметної області
Basic Formal Ontology [7]	Functionally Graded Materials Ontology [62]
DOLCE [44]	An ontology for software [130]
Information Artifact Ontology [23]	Ontology of Datatypes [144]
	Software ontology [47]
The Unified Foundational Ontology [75]	Software Process Ontology [78]
	Ontology of Software Defects, Errors and Failures [48]
UBERON [121]	Cell Ontology [115]
Good relations [209]	Linked-Open Tenders Electronic Daily LOTED2 [43]
Wikipedia	OntoMathPro [53, 135]
Collections Ontology [25]	Rail Core Ontology [196]
InteGRail ontology [91]	Network Statement Checker Ontology [197]
Basic Geo WGS84 ontology [207]	The Transport Disruption Ontology [30]
Toronto Virtual Enterprise [54]	iCity [98]
Error ontology [147]	Publishing Workflow Ontology [65]
	Collections Ontology [25]

Виконується інтеграція з онтологією верхнього рівня, як у [62], мостовою онтологією» – [115], іншою релевантною онтологією – [196], DBpedia – [53, 135].

З них на транспорті – в [196, 30, 197]. Засобами Collections ontology [25] представлений маршрут руху поїзда, Basic Geo WGS84 ontology [30] – розташування транспортних заходів та автобусних маршрутів.

Інтеграція онтологій LOTED2 [43] и Good relations виконується композицією відношень (Рисунок 1.6), що дозволяє представляти постачальників послуг та замовників у сфері державних закупівель.

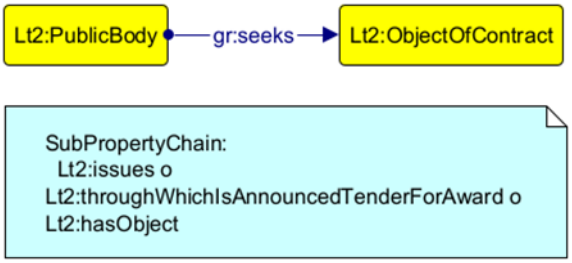


Рисунок 1.6 Інтеграція онтологій LOTED2 і Good Relations

Онтологічними засобами на транспорті інтегрують дані карт і розкладу поїздів [196], швидкості автомобіля [213], обладнання для людей з обмеженими можливостями [17] та інші. За допомогою реляційних даних АСК ВП УЗ-Є Укрзалізниці виконана інтеграція станційних, вагонних, поїзних та інших моделей. У суміжних областях онтологічними засобами інтегрують онтології предметних областей і онтології, що представляють клієнтів підприємств [43]. Можливою подальшою роботою є розвиток онтологічного забезпечення інтеграції даних Укрзалізниці, як у [178].

1.7 Перевірка узгодженості даних таблиць з нормативною документацією або моделлю онтологічними засобами

Стратегія розвитку Укрзалізниці [186] передбачає забезпечення структурних підрозділів достовірною інформацією. В Укрзалізниці кожен етап передачі даних супроводжується перевіркою фахівців. Наприклад, при передачі попереджень, вони перевіряються на відповідність до книги попереджень про обмеження швидкості відповідно до вказівок щодо попередження [195]. При поданні заявки про обмеження швидкості вона перевіряється енергодиспетчером на достатність заходів щодо

забезпечення безпеки відповідно до «правил безпеки експлуатації контактної мережі та пристроїв електропостачання автоматичного блокування залізниць» [185].

Перевірка структури джерел даних певною мірою виконуються автоматизованим способом в базах даних, але не в таких програмах для розрахунків і креслень, що застосовуються на УЗ, як AutoCAD. Попередження про тимчасові обмеження швидкості, що передаються машиністу електровоза, повинні відповідати реальній несправності залізничної колії і узгоджуватися з нормативними документами, що регламентують обробку несправності.

Можливі розбіжності можуть бути пов'язані з роз'єднанням інформаційних систем підсистем колійного господарства залізниць і управління рухом поїздів. Крім того, для проведення ремонтних робіт на залізничній колії представник колійного управління повинен визначити обмеження швидкості, необхідне для проведення робіт, і подати заявку вручну.

Розглянемо існуючу послідовність обробки інформації про несправність шляху

1) шляховиками при виявленні визначається обмеження швидкості і робляться записи про:
   a) несправність колії в книзі огляду залізничних колій ПУ-28 (КОЗК) згідно з інструкцією з поточного утримання залізничної колії (ІПУЗК) [228];
   b) про обмеження швидкості в книзі попередження про обмеження швидкості ПУ-84 (КЗПОШ) відповідно до Інструкції з руху поїздів по залізницях (ІРП) [203] та відповідних баз даних (БД);
2) шляховики передають інформацію про тип і місце несправності і обмеження швидкості представникам системи управління руху телефонограмою, згідно з яким представники системи управління руху вводять запис в КППОС представників системи управління руху згідно ІРП [203];
3) представники системи управління руху перевіряють швидкість КЗПОШ і форми видачі машиністу попереджень про обмеження швидкості перед тим як видати водієві відповідно до вказівок щодо автоматизованої видачі та скасування попереджень (МВАВВП [195]);

4) представниками системи управління руху видається машиністам поїздів обмеження швидкості в ФВМПОШ відповідно до правил технічної експлуатації залізниць (ПТЕ [204]);
5) при виконанні ремонту несправності робиться запис з датою ремонту в КОЗК і відповідній базі даних згідно ІПУЗК;
6) запис про скасування обмеження швидкості фіксується в КЗПОШ ПУ-84 та відповідному БД згідно ІРП [203];
7) інформація про скасування обмеження швидкості передається представникам системи управління руху, згідно з якими представникам системи управління руху роблять запис в КЗПОШ представниками системи управління руху згідно ІРП [203].

Така послідовність складна і з огляду на ручний ланцюг передачі інформації можливі невідповідності, які зрідка виявляються представниками управління руху.

Перевірка узгодженості онтології – це процес застосування правил і обмежень до даних (Таблиця 1.6). Вхідними даними є схема онтології і дані, а вихідними даними – онтологія з семантичним контролем джерела. Процес виглядає наступним чином:

1) розробка словника онтології;
2) розробка правил – друга частина схеми онтології;
3) заповнення онтології екземплярами;
4) перевірка узгодженості онтології різонером (reasoner) для OWL – Hermit, Pellet, для SWRL – Drools, для The Shapes Constraint Language (SHACL) – TopBraid SHACL API тощо.

Таблиця 1.6 Перевірка узгодженості даних в існуючих онтологічних розробках

Спосіб представлення джерел	Онтологія предметної області
MappingMaster [140]	COviD-19 Ontology [49]
RDB to RDF Mapping Language (R2RML) [155]	iCity [98]
Json script	Rail Core Ontology [196]
R2RML	Geospatial ontology [188]
R2RML, Ontop	Steel Cold Rolling Ontology [13]
R2RML, Ontop	iCity [98]
R2RML	SAAM ontology [189]
RDF Data Cube [192]	Organization Ontology [67]
CSV on the Web (CSVW) [34]	Function Ontology [116, 24]

Залежно від функціональних вимог (перетворення даних або перевірка узгодженості даних з нормативною підтримкою) програмні засоби використовують вихідні онтології або онтології домену. Наприклад, для формалізації стандарту була розроблена онтологія Generic ontology of datatypes [144], а для опису джерела даних використовуються CSVW [34].

Для перевірки узгодженості з моделями даних, наприклад, розробляється онтологія [104] для аналізу надійності двигуна, [41] – аналізу ризиків будівельних проектів, [65] – видавничого процес, [14] – розрахунку показників ефективності на транспорті (інтеграція онтології KPIONTO [39] і моделі Transmodel [22]), [13] – металопрокату, [214] – опису статистичного аналізу в біологічному домені.

Обмеження потужності CSVW [34] регламентується словами, тому що це схема RDF, легка онтологія, наприклад «Таблиця повинна мати один або кілька стовпців, а порядок стовпців у списку є значущим і повинен зберігатися програмами».

Для легких онтологій типу RDF Data Cube и GeoSPARQL [68], розробляються пакети обмежень [38, 67, 165] з використання мов SHACL [171] і Shape Expressions language (ShEx) [170].

Розглянемо приклад перевірки узгодженості даних з онтологічною моделлю OBCS [214] засобами OWL. Обмеження кардинальності для опису класу «one dimensional cartesian spatial coordinate datum» виключає наявність в онтології двох значень для координат x в прямокутній системі координат (Рисунок 1.7).

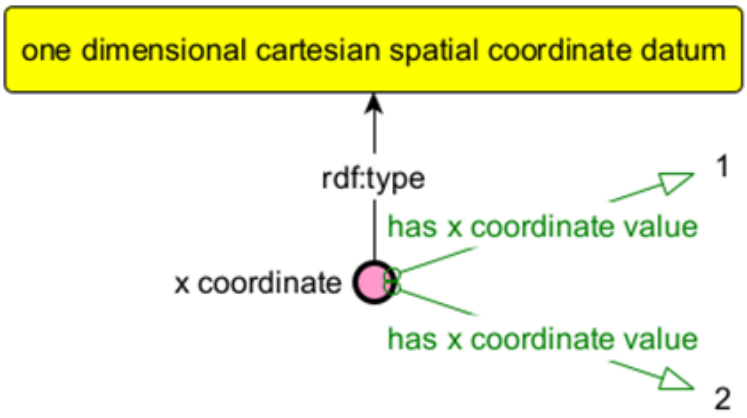


Рисунок 1.7 Неузгоджена онтологія OBCS з двома значеннями координат x

Онтологічними засобами на транспорті перевіряють узгодженість формул розрахунку показників ефективності [14]. Фахівці відповідних підрозділів

Укрзалізниці перевіряють узгодженість даних і нормативних документів. У суміжних областях онтологічними засобами перевіряють узгодженість даних нормативних документів [146]. Можливою подальшою роботою на транспорті є перевірка даних АСК ВП УЗ-Є на відповідність нормативній документації онтологічними засобами, як у [178]. У напрямку автоматизованого наповнення онтологій даними на залізничному транспорті – контроль структури джерел даних за допомогою онтологій типу [179].

1.8 Ontology learning у залізничному контексті

Онтології на транспорті у Європі розробляються у вигляді еволюції інформаційного забезпечення та її інтеграції різнорідних баз даних без зміни. Розробка виконується з використанням взаємодоповнюючих програмних засобів. Однак на залізничному транспорті перетворення даних на онтологію застосовується лише частково. У [15] перетворено UML модель RailTopoModel у схему онтології на основі методу Zedlitz та Luttenberger [211]. Під час розробки онтології Rail Core Ontology [196] використовується OpenRefine для перетворення розкладу поїздів в екземпляри онтології.

Автоматизація розробки онтологій передбачає автоматизацію розробки схеми та автоматизацію перетворення даних на екземпляри онтології.

Для схеми онтології вона виконується такими методами:

1) перетворення схеми XML, наприклад за допомогою XSD2OWL [33] та у схему онтології;
2) використання для розробки онтології контрольованої мови, наприклад OntoDL Onto2OWL [61];
3) автоматизоване семантичне анотування документів, наприклад, за допомогою text2onto [26].

Для нашої роботи актуальнішими є відображення табличних структур даних. У разі створення схеми онтології з використанням схеми бази даних SQL-DDL використовуються правила відображення. Наприклад, таблиці відображаються на

класи онтологій [5]. Інші засоби генерації онтології з урахуванням баз даних представлені у огляді [110].

Формалізація формування схеми онтології також виконується на основі мови онтологічних шаблонів (ontology pattern language) (OPL). У рамках підходу, авторами розроблено онтологічні шаблони розробки (ontology design patterns), а засобами OPL продемонстровані можливості щодо їх комбінування. Для представлення мови онтологічних шаблонів використовуються граматики [166] та мови OPL [153], засновані на правилах підстановки.

Автоматизація перетворення даних на екземпляри онтології виконується методами:

1) перетворення даних, наприклад Karma [102] та OpenRefine ;

2) присвоєння семантичних ролей, наприклад Inception [101];

3) A Virtual Knowledge Graph System, наприклад Ontop [18]

4) Cognition Relations or Concepts Using Semantics [108, 225].

Застосовуються засоби типу RDB to RDF Mapping Language для відображення даних таблиці на схему онтології з подальшим наповненням онтології екземплярами для інтеграції даних та перевірки їх узгодженості.

Іншим способом автоматизації розробки формування онтологій з використанням кількох програмних засобів є платформи для їх інтеграції. Під інтеграцією можна розуміти зв'язок сервісів у додатках, заснованих на онтологіях, де використовують повідомлення RabbitMQ і Redis [196].

Для нашої роботи більш актуальна інтеграція додатків розробки онтології в сенсі відображення різнорідних файлів, наприклад, як OntoPop [193], де проводиться відображення тегів для анотування і класів онтології. Спочатку виконується анотування документів, розробка правил мовою LangText [32] для відображення тегів та онтології, а потім наповнення онтології. Примірники онтології виходять із тексту.

В інших платформах, наприклад [42], для відображення анотацій на екземпляри онтології використовуються правила Java Annotation Patterns Engine як частина text2onto [26].

Висновки до першого розділу

Визначені аспекти існуючих онтологічних розробок, яким приділяється недостатня увага: контроль структури даних при перетворенні їх в онтологічні екземпляри і перевірка узгодженості даних з нормативним забезпеченням перевізного процесу. Відповідно до огляду [110] інтеграції даних при OL приділяється недостатня увага. Вперше закладено основу для досягнення цілей розвитку Укрзалізниці [186] онтологічними засобами, з використанням аналізу і систематизації існуючих онтологічних розробок транспортних і суміжних областей.

Науково обґрунтовані можливості використання онтологічних засобів на залізничному транспорті для:

1) формалізації нормативного забезпечення, що підтверджується роботами [48, 144, 62];
2) перетворення даних, що підтверджується роботами [196];
3) інтеграція даних, що підтверджується роботами [17];
4) перевірка узгодженості даних інформаційних систем та інструкцій, що підтверджується роботами [146, 212].

Дослідження дозволило виявити найбільш значущі онтологічні роботи на транспорті. Закладено основи концептуалізації табличного представлення знань та розробки онтології для інтеграції моделей залізничних підсистем.

Аналіз показав, що онтології в залізничному транспорті Європейського Союзу використовуються для інтеграції даних описів інфраструктури, розкладу поїздів та інших. При цьому недостатня увага приділяється нормативному забезпеченню процесу перевезення.

Існують програмні засоби для анотування текстів, видобування знань з таблиць і розробки онтологій, але вони не використовуються для підтримки процесу перевезень на залізничному транспорті України. Визначено, що актуальним завданням є анотація нормативної документації для встановлення зв'язку між онтологією та текстами інструкцій, а також їх автоматизована генерація.

За результатами розділу, опубліковано роботи [173, 174, 175, 177, 178, 179, 180, 218, 219, 220, 221, 222, 223].

РОЗДІЛ 2
МОДЕЛЮВАННЯ ОНТОЛОГІЙ ЗАЛІЗНИЧНОЇ ІНФРАСТРУКТУРИ

Дослідження існуючих онтологічних розробок на транспорті виявили необхідність перевірки структури джерел даних та їх узгодженості з нормативним забезпеченням. У цьому розділі розроблена модель табличного представлення знань та її інтеграція з моделлю залізничної інфраструктури. Розкрито порядок формування модульної онтології залізничної інфраструктури.

2.1 Абстрактні і конкретні онтології

Розроблено метод формування онтологій трьох рівнів. Абстрактний рівень представлений словником. Моделі першого рівня конкретизації збагачуються правилами і даними, доформується Tbox схеми онтології, класи словникова зв’язуються з екземплярами. У моделях другого рівня конкретизації правила зв’язуються з набором даних. Наявність другого рівня конкретизації дозволяє сформувати декілька онтологій (на основі однієї онтології першого рівня) з використанням різних даних.

Онтології абстрактного рівня визначають понятійний апарат словника термінів (концептів), до них відносяться і концепти відношень. Наприклад, поняття: залізнична станція, колія, світлофор, граничний стовпчик тощо; мати ухил, з'єднувати станцію, мати алгебраїчну різницю ухилів.

На першому рівні конкретизації абстрактні онтології збагачуються в двох напрямках: в одному формуються екземпляри понять (наприклад, конкретна станція «Київ», конкретна колія № II станції «Київ»), в іншому – правила і обмеження (наприклад, ухил повинен бути не більше 40 ‰).

На другому рівні конкретизації зв'язуються абстрактні та онтології першого рівня. Сформовані поняття мають повноту властивостей і відношень, необхідних для семантичної перевірки.

Оскільки узгодженню підлягає інформація з електронних джерел різного формату (баз даних, електронних таблиць і текстових документів), необхідно мати формалізоване і єдине представлення даних з цих джерел. Дворівнева конкретизація

виконується аналогічно онтології предметної області, описаної вище. Модель, сформована таким чином, показана на Рисунку 2.1.

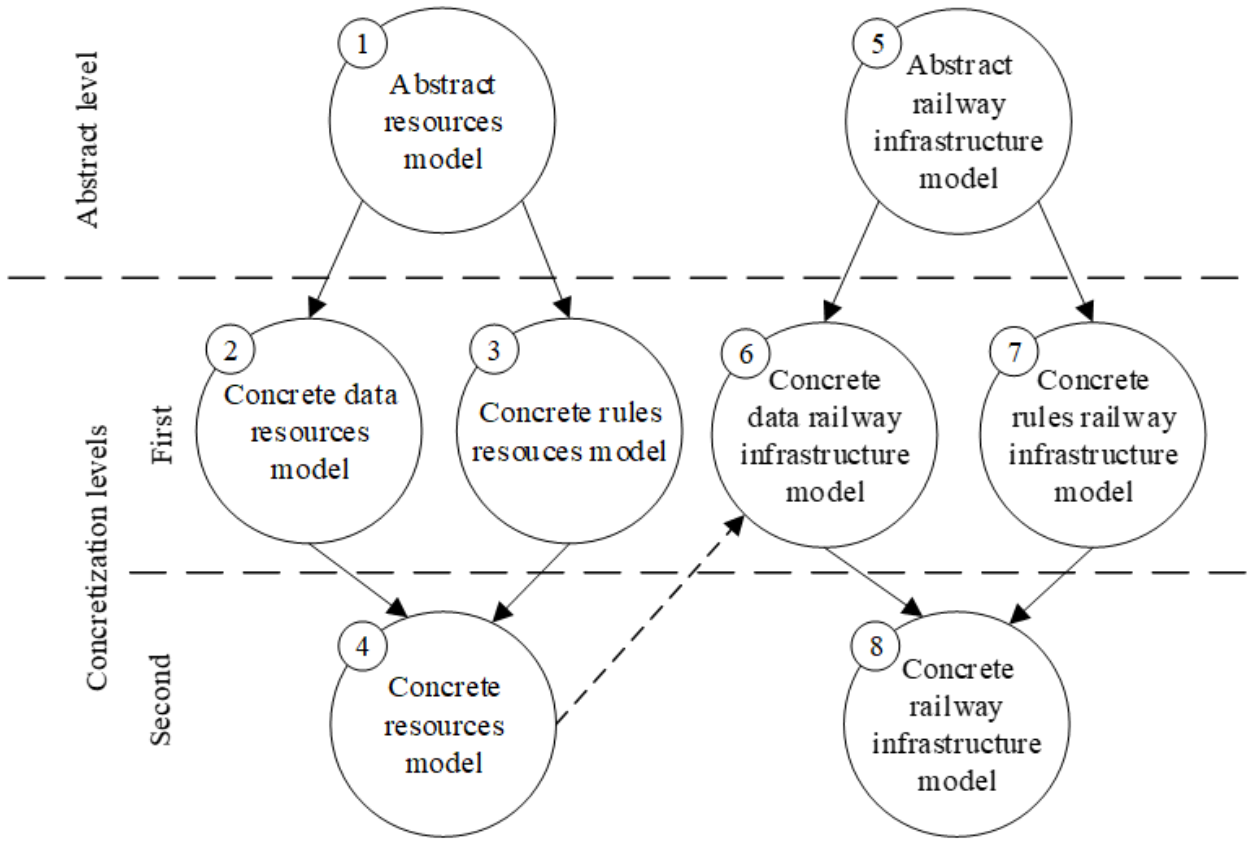


Рисунок 2.1 Інтеграція залізничної та вихідної моделей

Моделі з'єднуються між собою шляхом включення для перетворення даних в онтологічне представлення (між моделями 4 і 6).

Розглянемо далі можливості програмної реалізації розроблених моделей і взаємозв'язки між ними.

2.2 Інструментарій розробки та інтеграції онтології

У нашому дослідженні для розробки онтологій були використані наступні програмні засоби:

1) редактор онтологій у форматі OWL Protégé;

2) редактор онтологій у форматі RDF MRcube;

3) інструмент перетворення формату даних OpenRefine;

4) інструмент для видобування даних Tabula;

5) переглядач PDF Sumatra;

6) електронні таблиці для формування вихідних даних MS Excel;

7) система управління базами даних SQLite.

Онтології, в залежності від складності і мови (RDFS або OWL), обробляються різними програмними засобами. MRcube з підтримкою висновку type propagation та обмеження щодо область значень. Protégé з підтримкою шпильки OWL-DL.

В Protégé виконується instance checking, тобто перевірка узгодженості даних і схеми онтології. Висновки робить різонер (reasoner) Hermit в рамках OWL-DL. Застосовуються такі функції OWL, як property chains.

Порядок використання програмних засобів для формування моделей зведений в таблицях 2.1 - 2.2.

Таблиця 2.1 Специфікація онтологічних моделей джерел даних

Модель	Програмний засіб	Спосіб розробки	Реалізація сформованої моделі	Формат та джерела вихідної інформації
Abstract resources model	MRcube	ручний	Словник RDFS/ OWL	- таблична модель даних [179] - SWO [111]
Concrete data resources model	Sumatra	ручний	Джерела у форматі pdf	Креслення колійного розвитку станції
	Excel	ручний	Джерела у форматі xls	Креслення профілю
	SQLite	ручний	Джерела у форматі db	Відомість колій
	Tabula, OpenRefine, Protégé	автомати-зований	Дані RDF, що описують залізничну інфраструктуру	Джерела у форматі pdf, xls, db, що описують інфраструктуру залізничної станції та Abstract resources model
Concrete rules resources model	Protégé	ручний	Онтологія з процедурами перетворення даних (скриптів) і структури таблиці, правилами розпізнавання джерела даних, перевірки узгодженості структури таблиці	- процедури перетворення даних таблиць у вихідному форматі в онтологічні екземпляри; - параметри джерела даних Error ontology [147] і Abstract resources model
Concrete resources model	Protégé	ручний	Онтологія з семантичним контролем структури джерела	Concrete data resources model і Concrete rules resources model

Таблиця 2.2 Специфікація онтологічних моделейзалізничної інфраструктури

Модель	Програмний засіб	Спосіб розробки	Реалізація сформованої моделі	Формат та джерела вихідної інформації
Abstract railway infrastructure model	Protégé	ручний	Словник OWL	Понятійний апарат елементів станції за правилами технічної експлуатації залізниць (ПТЕ [204]) та державними будівельними нормами (ДБН [168])
Concrete data railway infrastructure model	Protégé	ручний / автомати-зований	Онтологія з даними, що описують залізничну інфраструктуру	Конкретна онтологія джерел другого рівня і «мостова онтологія» (bridging ontology)
Concrete rules railway infrastructure model	Protégé	ручний	Онтологія з формалізованими обмеженнями, накладеними на відношення словника абстрактної моделі	Обмеження ПТЕ [204] і ДБН [168] і Abstract railway infrastructure model
Concrete railway infrastructure model	Protégé	ручний	Онтологія з семантичним контролем узгодженості джерел даних, що описують залізничну інфраструктуру	Concrete data railway infrastructure model і Concrete rules railway infrastructure model
Bridging ontology	Protégé	ручний	Онтологія з відображенням між поняттями джерел і залізничної інфраструктури	Abstract resources model і Abstract railway infrastructure model

На Рисунку 2.2 продемонстровано порядок використання програмних засобів, а також імпортування (з використанням механізму owl:import застосунку Protege «Import ontology wizard») онтологій для їх розробки.

Розглянемо порядок на прикладі. У редакторі онтологій MRcube розробляється онтологія Abstract resources model. Таблиця із зазначенням власників під'їзної колії у форматі pdf видобувається з креслення під'їзної колії в Tabula, тобто перетворюється у формат csv.

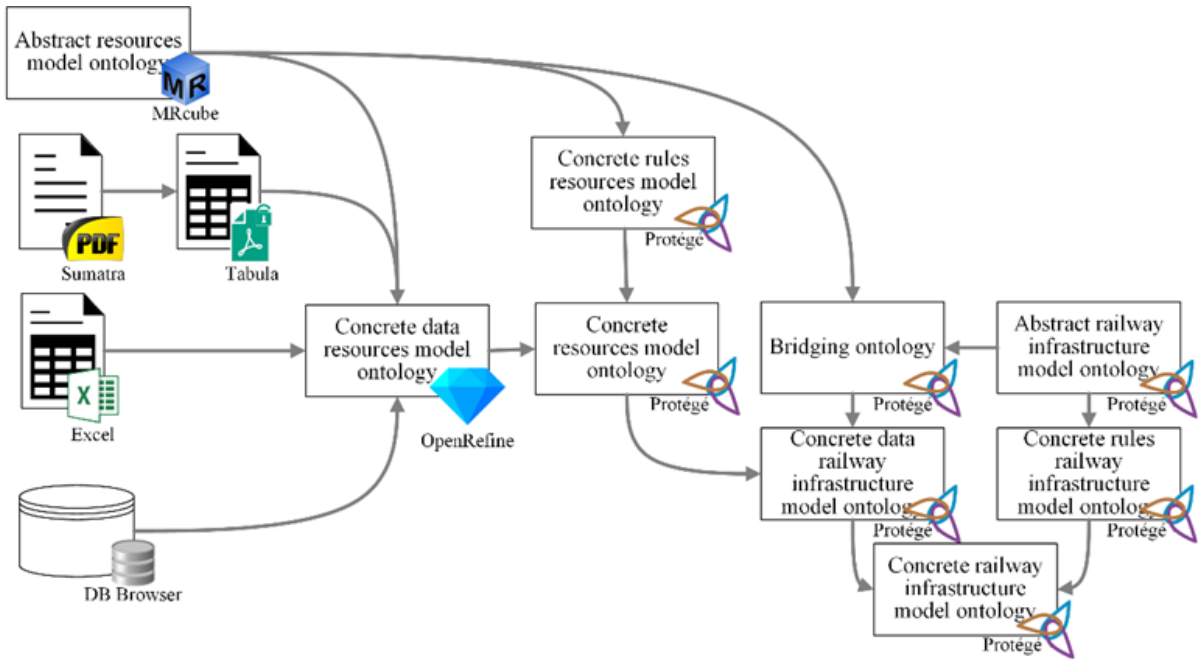


Рисунок 2.2 процедура розробки онтології

Відомість колій у форматі db, таблиця з власниками у форматі csv та профіль у форматі xls перетворюються за допомогою OpenRefine і Abstract resources model (1) в Concrete data resources model (2).

Всі наступні онтології розроблюються в редакторі онтологій Protégé і імпортуються в наступному порядку:

1) abstract resources model і abstract railway infrastructure model імпортується в bridging ontology;
2) abstract resources model імпортується в concrete data resources model і в concrete rules resources model;
3) concrete data resources model і concrete rules resources model імпортуються в concrete resources model;
4) concrete resources model і bridging ontology імпортуються в concrete data railway infrastructure model;
5) abstract railway infrastructure model імпортуються в concrete rules railway infrastructure model;
6) concrete data railway infrastructure model і concrete rules railway infrastructure model імпортуються в concrete railway infrastructure model.

Виконується наповнення онтології екземплярами з вихідних файлів INCEpTION [101], одержаних внаслідок анотування нормативно-правових документів залізниці. Анотування – це процес зміни тексту шляхом додавання до нього метаданих (тегів, концептів онтологій).

2.3 Концептуалізація табличного представлення знань

Формування моделі починається з найбільш загального концепту. Концепт – це те, з чим оперують природні та автоматизовані інформаційні системи: поняття, об'єкт, процес реального або віртуального світу, відношення, дія, властивість тощо (Рисунок 2.3).

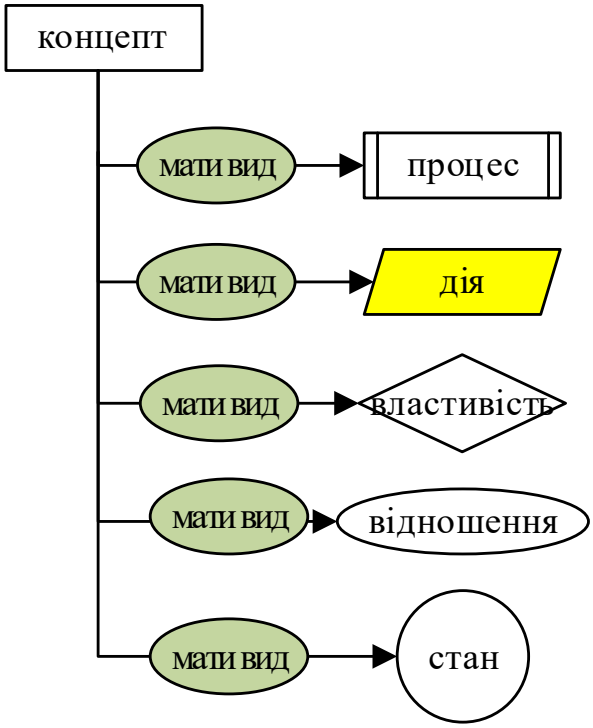


Рисунок 2.3 Основні поняття

Тут і далі застосовується формалізація онтологічних елементів і їх зв'язків у вигляді графічної нотації формальної мови. Зв'язки на схемах позначаються спрямованими стрілками, а різні види понять – відповідними геометричними фігурами.

На Рисунку 2.3 виділяються основні поняття моделювання інформаційних систем: властивість – здатність відображати певну якість понять (наприклад, колір, вага, розмір, вміст білка); відношення – осмислений або неосмислений зв'язок між

поняттями; стан – визначає значення властивостей поняття в певний момент часу: дія змінює властивості понять; процес – зміна властивостей понять з плином часу. Відношення «has kind» – це видовий зв'язок від загального до конкретного.

У реальному світі та інформаційних системах всі концепти складні, збудовані як на Рисунку 2.4.

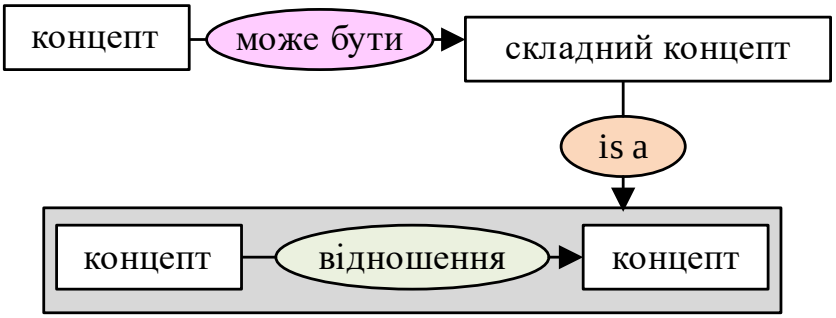


Рисунок 2.4 Складне поняття

З урахуванням рекурсивного представлення концептів на Рисунку 2.15, може бути представлений концепт будь-якої складності (з будь-якою кількістю складових концептів і різноманітними відношеннями). Оскільки поняття і відношення можуть бути порожніми, вироджене складне поняття також може бути порожнім.

Відношення, представлені на Рисунках 2.3 – 2.4 і далі інтуїтивно зрозумілі. Ми не будемо визначати їх тут.

Для зв'язування поняття і будь-яких його властивостей використовується відношення атрибутивності.

Для того щоб виключити з сукупності всі поняття, які можуть мати атрибут типу, поняття «тип» і «значення» вводиться відношенням «except».

Можна сказати, що будь-який концепт має тип (Рисунок 2.6), ідентифікатор і значення. З огляду на, що тип, ідентифікатор і значення також є концептами, попереднє твердження породжує рекурсію, яка не є прийнятною через винятки, наприклад, концепт «тип» не може мати атрибут «тип». Це враховується на Рисунку 2.5.

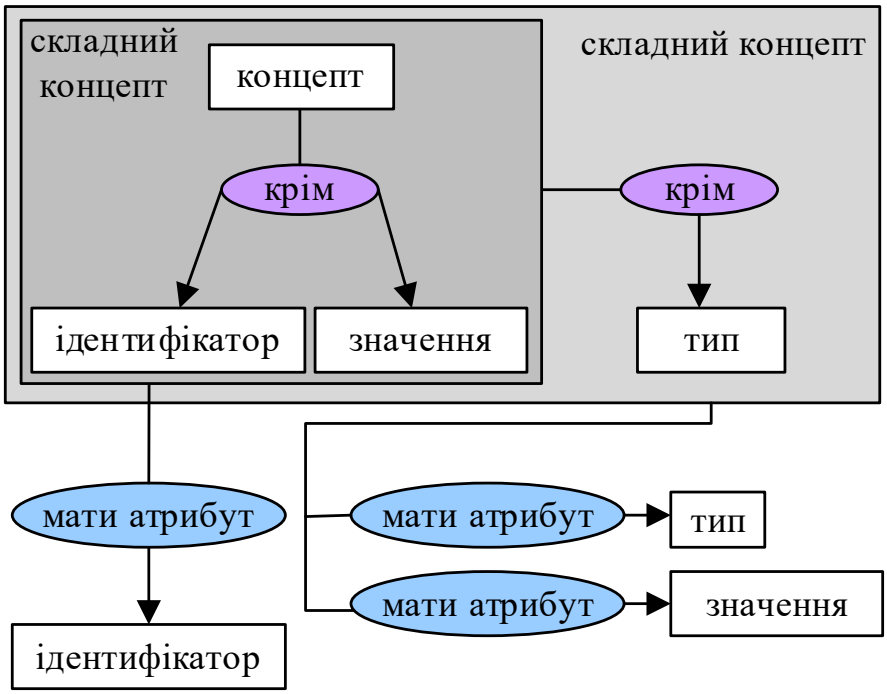


Рисунок 2.5 Обов'язкові атрибути

Для представлення типового твердження «всі поняття, крім типу і значення» скористаємося раніше визначеним «складним концептом» (темно-сірий прямокутник на Рисунку 2.5) і зв’яжемо «тип» з відношенням «мати атрибут».

Рисунок 2.6 Визначення типу

Перейдемо до розгляду основних понять, які є складними за структурою. Всі вони засновані на концепті послідовності (Рисунок 2.7).

Послідовність – це елементи впорядкованої множини, які з'єднані відношенням порядку.

Для представлення складових частин упорядкованої множини використовується відношення «has part».

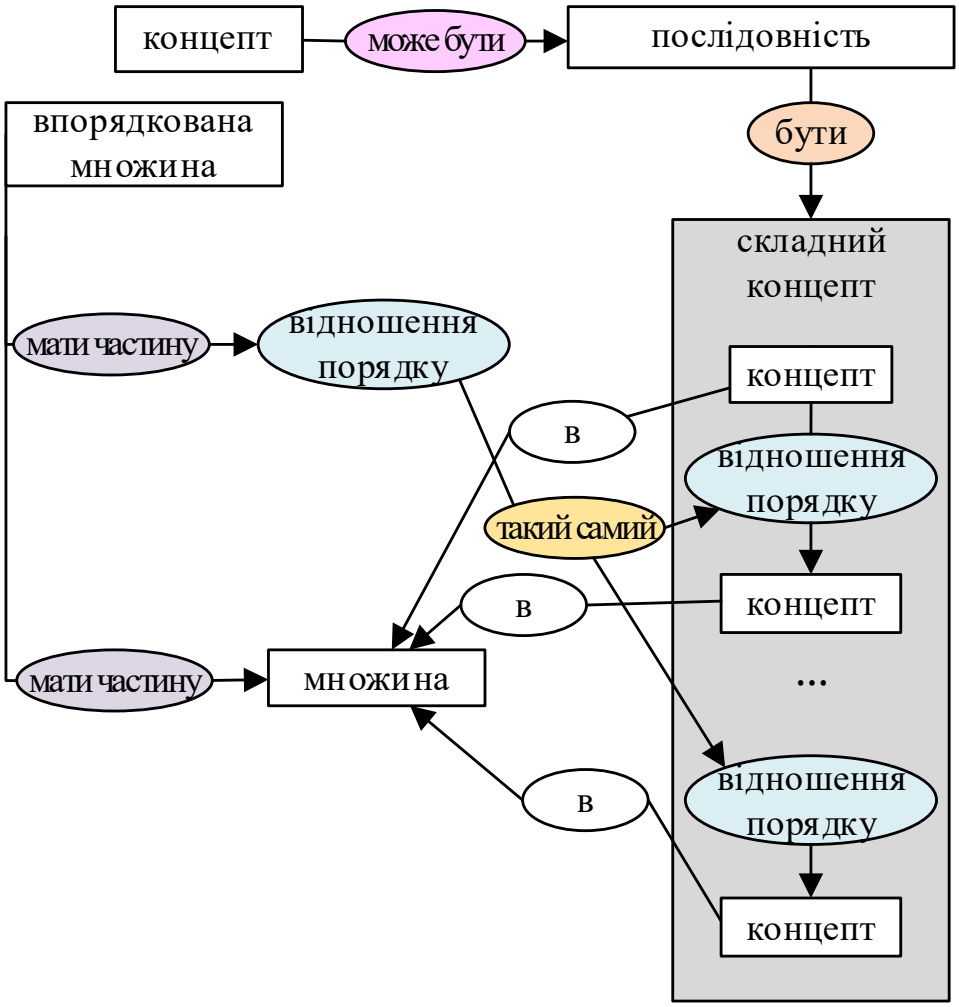


Рисунок 2.7 Концепт «послідовність»

Для моделювання послідовності два поняття пов'язані один з одним відношенням порядку за допомогою одного і того ж складного поняття. Де відношення порядку – це якесь напрямлене відношення.

Послідовність – це складне поняття (відношення «is-a»), елементи послідовності задовольняють умові, заданому відношеннями «in» для понять і «same as» для відношення порядку.

Основні складні поняття: кортеж, клас і вектор – це окремий випадок послідовності (Рисунок 2.8) з роз'ясненнями, представленими на Рисунках 2.9- 2.11.

Кортеж – це послідовність упорядкованих понять, без обмежень за типом і елементами.

Кортеж – уточнення послідовність у тому, що кортеж має ідентифікатори елементів як строка або натуральне число.

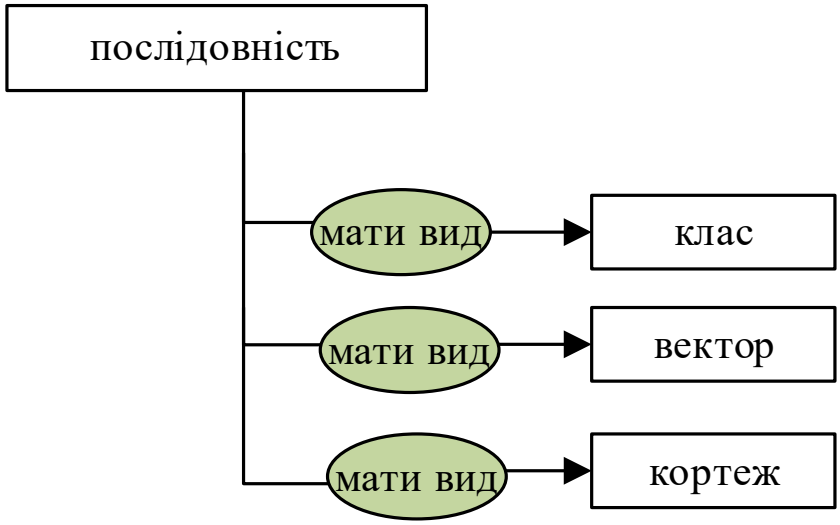


Рисунок 2.8 типи послідовностей

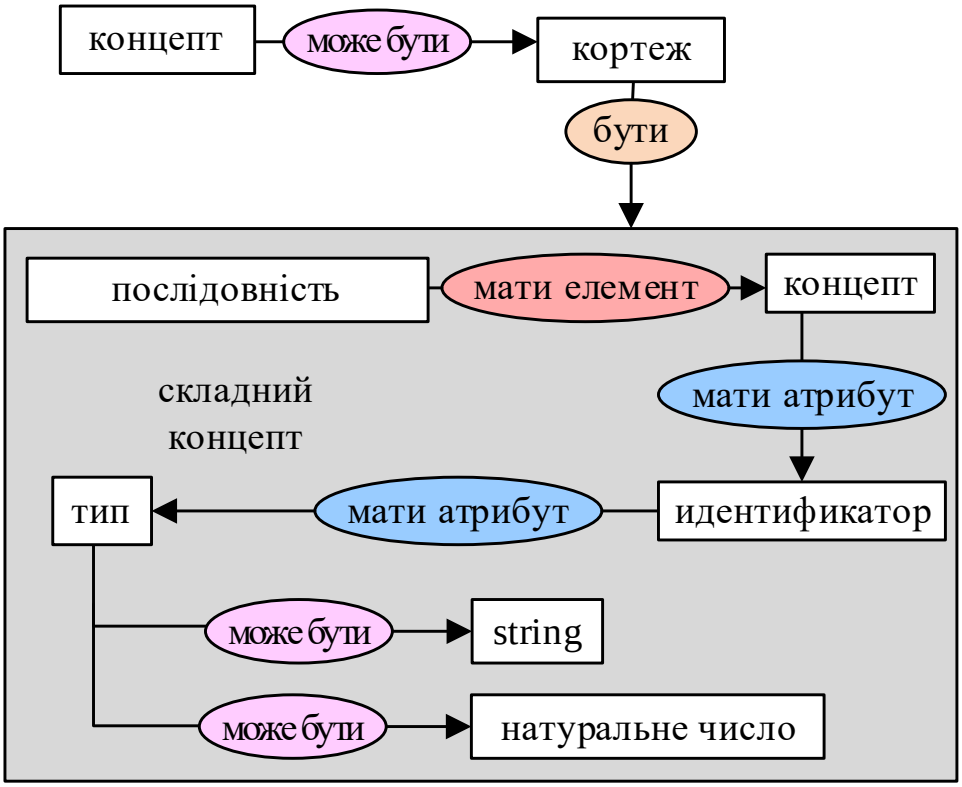


Рисунок 2.9 Концепт «кортеж»

Відношення порядку успадковується від складного поняття, відповідного послідовності і на малюнку не відображається.

Клас – це послідовність, елементом якої може бути дія.

Вектор – це сукупність упорядкованих понять одного типу.

Покажемо, як можна представити таблицю за допомогою розробленої моделі.

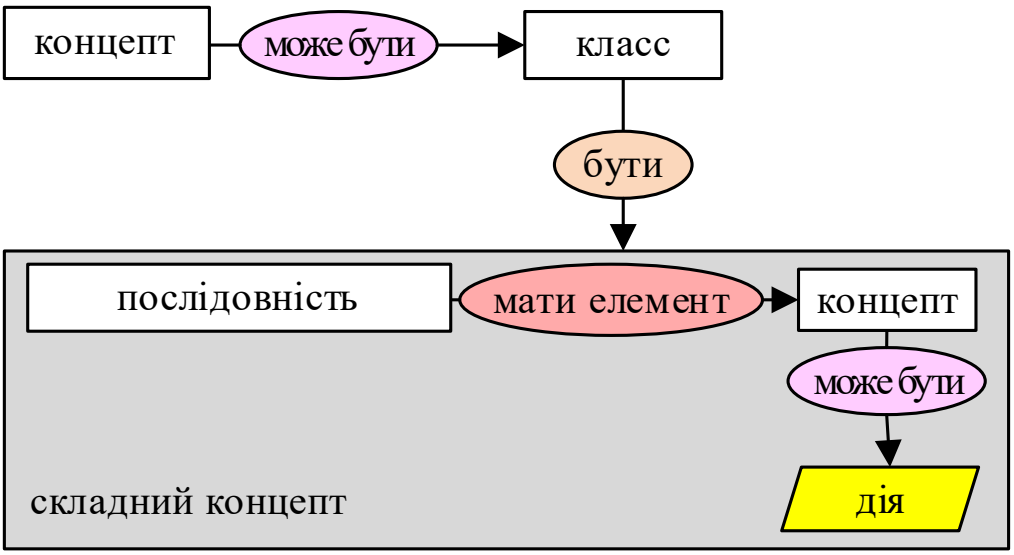


Рисунок 2.10 Концепт «клас»

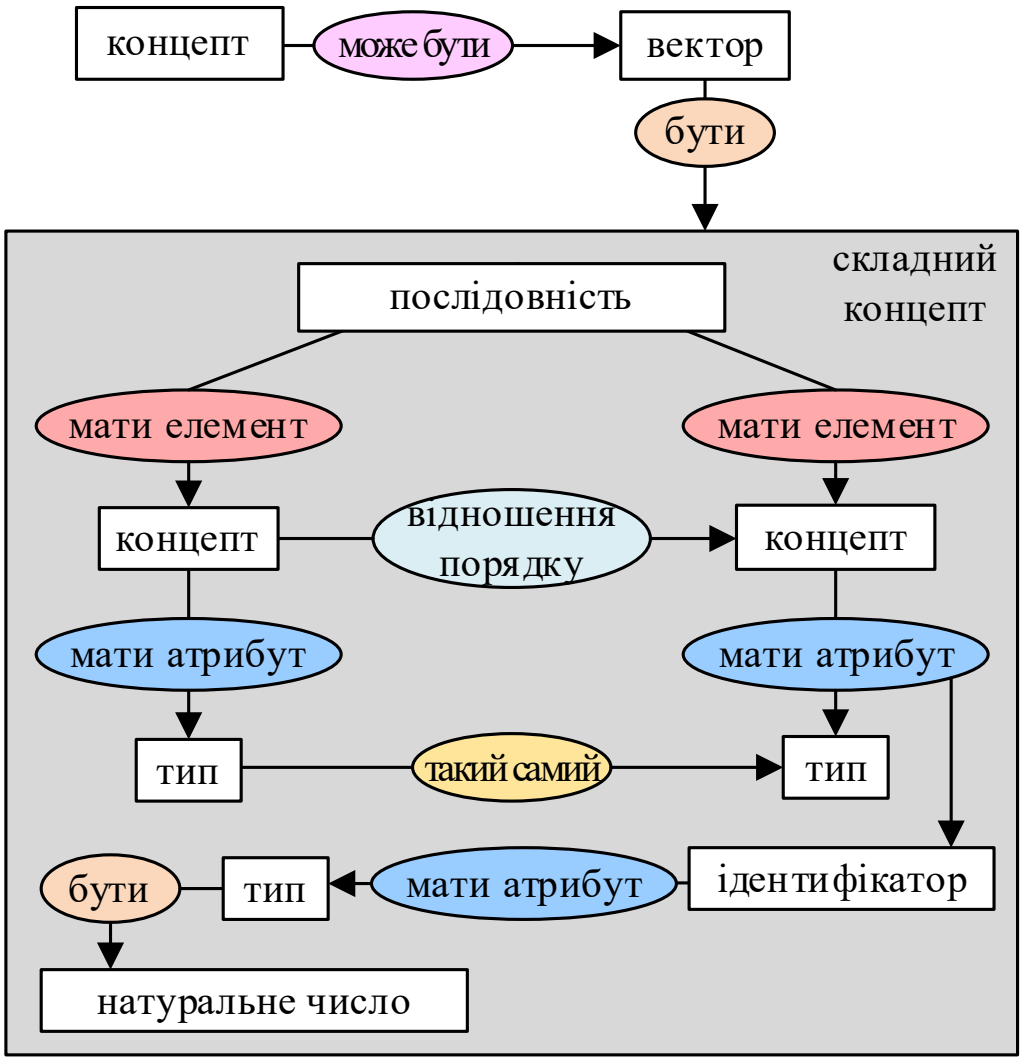


Рисунок 2.11 Концепт «вектор»

Таблиця – це складний концепт, елементом якого є послідовність, що складається з ідентифікуючого кортежу і вектору кортежів. Важливим аспектом табличного представлення знань є те, що елементи в ідентифікуючому кортежі і в кортежі зі значеннями зв'язані однаковим відношенням порядку, і що поняття з ідентифікатором і поняттям з величиною зв'язані відношенням іменування.

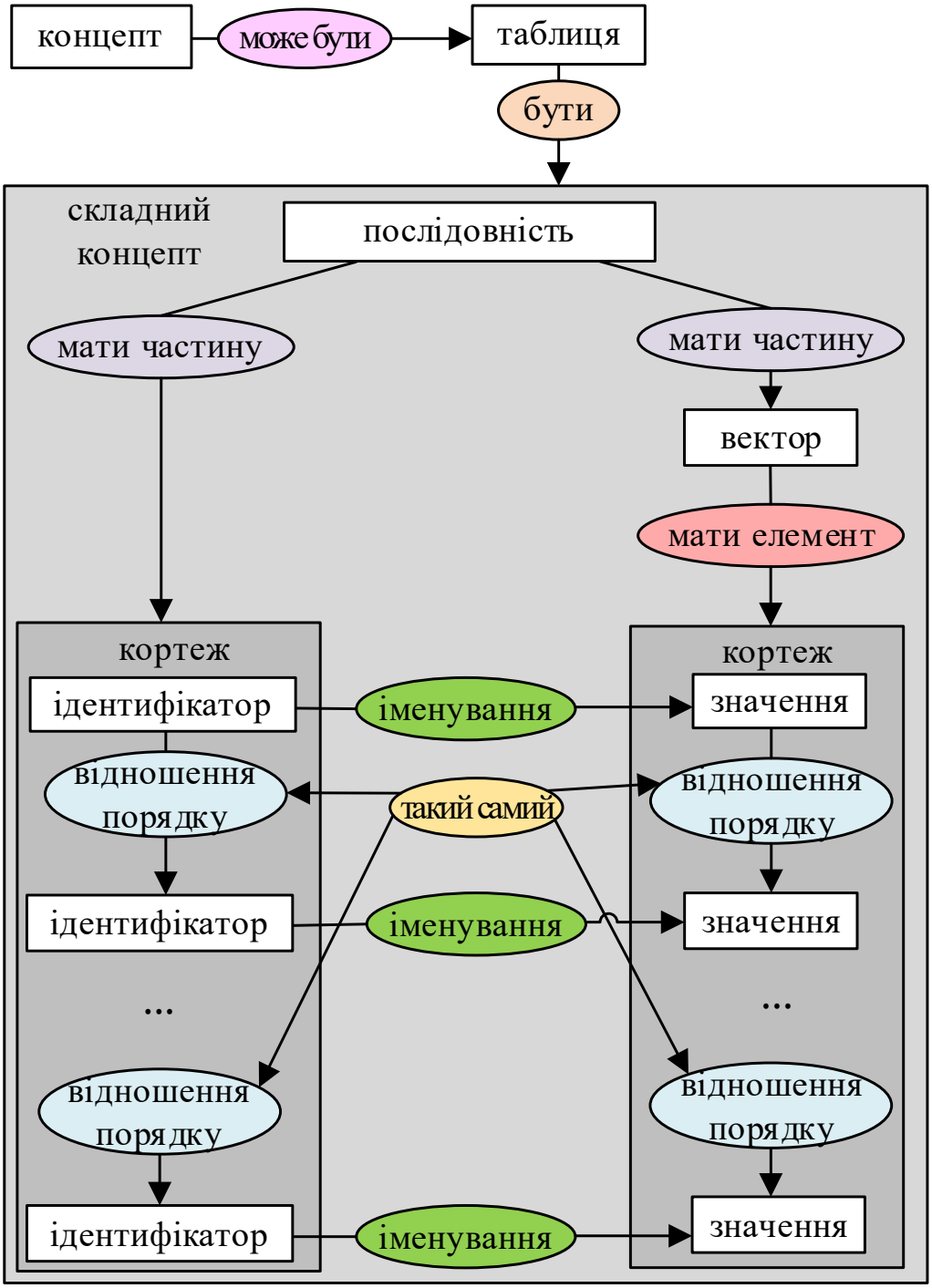


Рисунок 2.12 Конструктивний концепт «таблиця»

Представлений концепт більш загальний по відношенню до таблиць баз даних MS Word і MS Excel, з огляду на те, що в них відношення порядку – це відношення слідування, а в нашій моделі – це будь-яке напрямлене відношення.

2.4 Базова модель залізниці

Для об'єднання моделей АСК ВП УЗ-Є і їх індивідуальних онтологій була розроблена Базова модель – каркас модульної онтології, яка включає в себе онтології моделей АСК ВП УЗ-Є, нормативну документацію і бази даних підприємств замовників.

АСК ВП УЗ-Є включає в себе наступні моделі:

1) поїзну – нитки графіку руху поїздів;

2) станція – станційні схеми і нормативи технологічних операцій;

3) вагонну – дислокація вагонів;

4) відправочну – вантажі у вагонах.

Онтологічним шляхом оформлено наступну нормативно-правову документацію (конструкції природної мови [172]):

1) державні будівельні норми України споруди транспорту залізниці колії 1520 мм Норми проектування ДБН В.2.3-19-2008 (ДБН) [168];

2) інструкція з розробки графіків руху поїздів [228];

3) правила експлуатації вагонів [231];

4) правила перевезення небезпечних вантажів (ППНВ) [230].

Кожен вузол Базової Моделі відповідає під-онтології: квадратний вузол відповідає моделі АСК ВП УЗ-Є, а круглий із заповненням – нормативному документу, круглий без заповнення – базі даних підприємства клієнта (Рисунок 2.13).

Вся модель і кожен квадрат відповідає концепції модульної онтології, а кожний круг – модулю. За допомогою конструкції «owl:import» онтології модулів з'єднуються між собою. Крім неї, можуть використовуватися і засоби E-Connections [1].

Для взаємозбагачення та уніфікації онтологій-модулів вводяться відповідні класи – поняття.

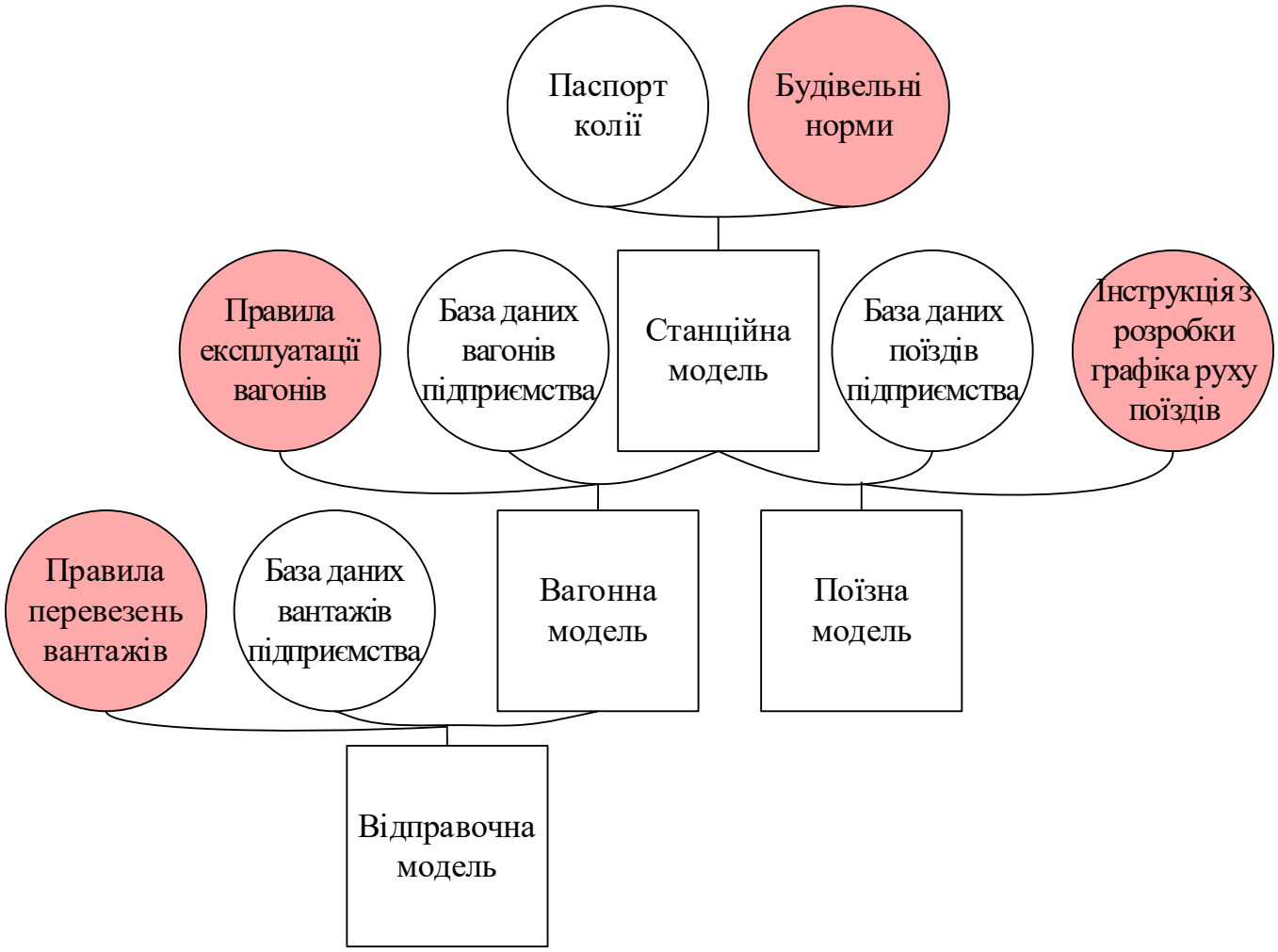


Рисунок 2.13 Базова модель інформаційних систем АСК ВП УЗ-Є

Глосарій інструкцій з руху поїздів [203] включає перегонні та станційні колії. У цих класах визначаються властивості справності і належності колії підприємству або станції.

Виходячи з того, що «мости» використовуються для «переробки … термінології в інженерному контексті» [113], перебудовується визначення під'їзної колії. Враховуючи обмеження ДБН [204], отримаємо визначення «під'їзна колія моделі станції – під'їзна колія, що з'єднує станцію «Укрзалізниці» з промисловою станцією та відповідає нормам Укрзалізниці».

Таким чином, було здійснено трансформацію визначення в контексті його технічних характеристик і уніфікації колія підприємства, нормативів колії і станційної моделі АСК ВП УЗ-Є.

За восьмизначною системою нумерації вагонів колії 1520мм визначаються такі типи вагонів: криті, спеціальні вагони, платформи, власні, піввагони, цистерни,

ізотермічні, інші. Враховано властивості справності вагона і розташування на підприємстві або станції. При узагальненні виділяється відношення-поняття «розташування», аналогічно [136, 12].

В якості «моста» використовується визначення «вагон вагонної моделі – такий, що відповідає нормам Укрзалізниці і знаходиться на колії моделі станції». Таке формулювання має на увазі відповідність технічним характеристикам вагонів, згідно з правилами експлуатації, і його розташуванню. Воно поєднує в собі вагон підприємства, норми правил експлуатації вагонів Укрзалізниці [231] та вагонну модель АСК ВП УЗ-Є. Також об'єднуються вагонна і станційна моделі, які будуть додатково формалізовані як «необхідна і достатня умова» для «додавання» вагона в вагонну модель.

Згідно з «Коментарів та роз'ясненнях щодо застосування положень правил технічної експлуатації залізниць України», поїзди відрізняються за пріоритетністю (позачергові, регулярні, призначаються при виникненні особливих умов), а для пасажирських поїздів – за видами сполучення (місцеві, приміські та міжміські). Розглянемо аспекти ваги поїзда, розташування на підприємстві або станції.

У контексті нормативів ваги поїзда згідно з інструкцією з розробки розкладу руху поїздів визначення формулюється як «поїзд поїздної моделі – це такий, що має допустиму вагу і знаходиться на колії моделі станції». Таке визначення дозволяє об'єднати станційну і поїздну моделі.

Згідно з ППНВ на залізницях України [230], вантажі класифікуються за їх властивостями. При цьому враховується завантаження у відповідний ППНВ вагон, що знаходиться на підприємстві або станції. Відповідно, вантаж-вагон можна розглядати як класифікацію за функціями, аналогічно, як і в [136, 12].

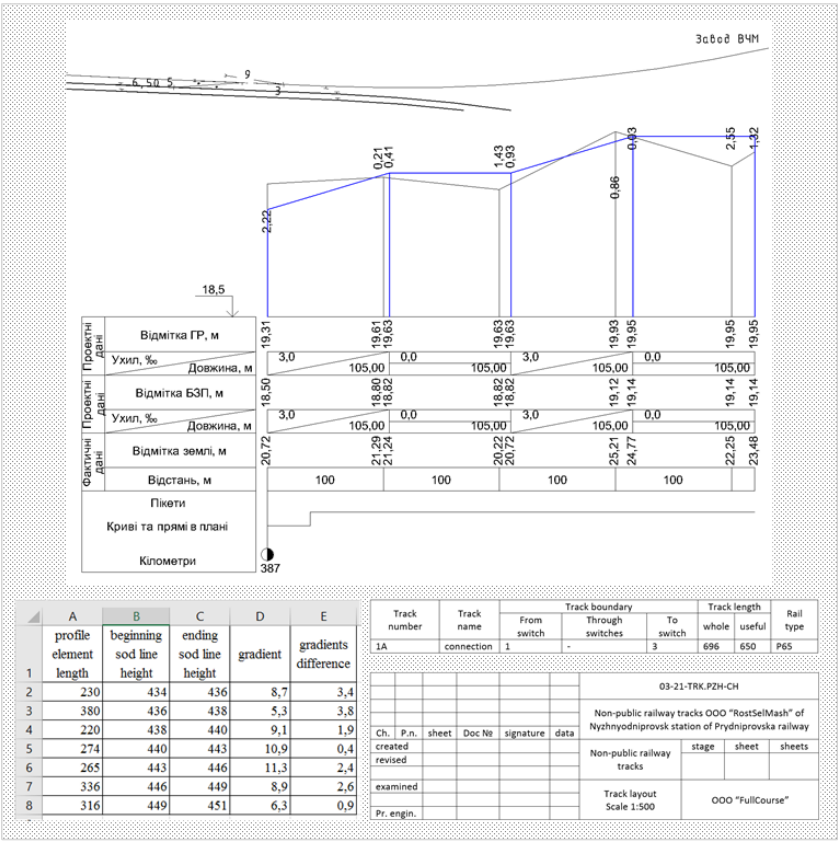
Завантажений вантаж в якийсь вагон з ППНВ [230] визначається як «вантаж відправочної моделі – такий, що завантажений у вагон вагонної моделі і відповідає правилам ППНВ». Це визначення об'єднує в собі відправочну і вагонну модель, а також станційну модель.

2.5 Порядок формування онтології залізничної інфраструктури на прикладі колійного розвитку станції

Розглянемо інформаційне забезпечення систем з точки зору під'їзних колій залізничних станцій. Вона включає в себе наступні табличні дані і нормативні документи: відомість колій, таблиця з власниками колій, поздовжній профіль колії, ПТЕ залізниць промислових підприємств [232], ДБН [204].

Розглянемо правила нормативних документів. Згідно з ДБН [168], характеристики плану, профілю і верхньої будови залізничної колії повинні задовольняти її умовам експлуатації. Для під'їзних колій алгебраїчна різниця ухилів не повинна перевищувати 13 проміле, а тип рейок – Р65.

Вихідні дані (Рисунок 2.14) представлені таблицями з різними форматами: розрахунок профілю колії виконується в Excel, креслення – в AutoCAD [224] (таблична частина рисунка перетворюється в формат pdf, а графічна представлена в ілюстративних цілях) і відомість колій знаходиться в базі даних.



	A	B	C	D	E
	profile element length	beginning sod line height	ending sod line height	gradient	gradients difference
2	230	434	436	8,7	3,4
3	380	436	438	5,3	3,8
4	220	438	440	9,1	1,9
5	274	440	443	10,9	0,4
6	265	443	446	11,3	2,4
7	336	446	449	8,9	2,6
8	316	449	451	6,3	0,9

Track number	Track name	Track boundary			Track length		Rail type
		From switch	Through switches	To switch	whole	useful	
1A	connection	1	-	3	696	650	P65

							03-21-TRK.PZH-CH
							Non-public railway tracks OOO "RostSelMash" of Nyzhnyodniprovsk station of Prydniprovska railway
Ch.	P.n.	sheet	Doc №	signature	data	Non-public railway tracks	stage / sheet / sheets
created							
revised							
examined						Track layout Scale 1:500	OOO "FullCourse"
Pr. engin.							

Рисунок 2.14 Вихідні дані на прикладі паспорта під'їзної колії (профіль, відомість колій і таблиця із зазначенням власників колії)

Паспорт під'їзної колії включає схему колійного розвитку під'їзної колії та її поздовжнього профілю. План колійного розвитку – це умовне представлення розташування залізничних колій в просторі. Поздовжній профіль – це проекція залізничної колії на вертикальну площину. Профіль складається з ухилів і горизонтальних площадок різної довжини.

На Рисунку 2.15 представлено розширення моделі Рисунка 2.1 з урахуванням різного формату і джерел даних. Нумерація онтологій на Рисунках 2.1 і 2.15 узгоджена. Позначення декількох однотипних онтологій доповнюються буквеним ідентифікатором, наприклад, «2a».

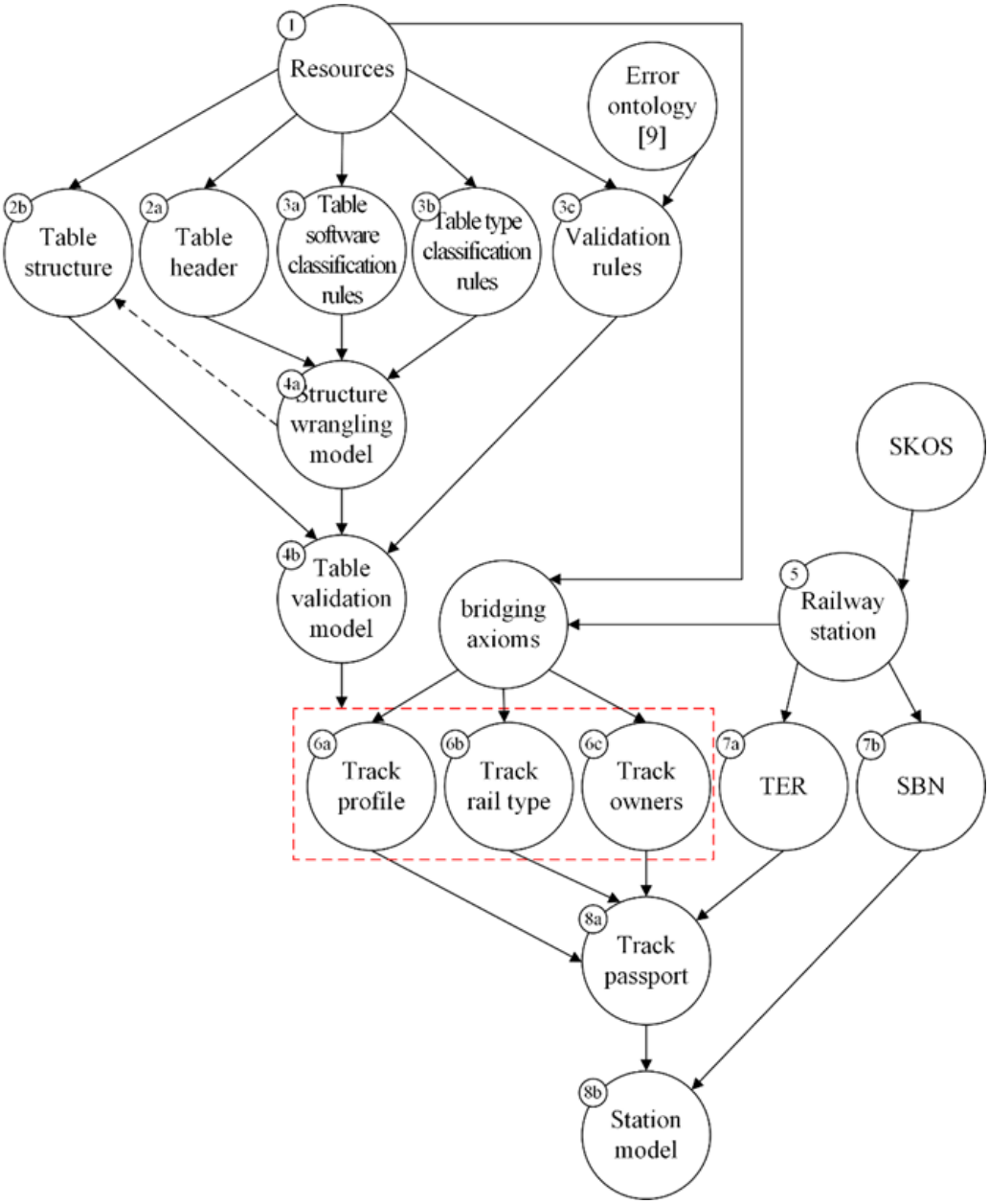


Рисунок 2.15 Онтологічне моделювання інфраструктури залізничної станції

В Таблиці 2.3 узагальнено приналежність онтологій до рівнів конкретизації Рисунка 2.1.

Таблиця 2.3 Рівні конкретизації модулів онтологій джерел та залізничної інфраструктури

Рівень	Модуль онтології	
	Онтологія джерел	Онтологія залізничної інфраструктури
Абстрактний рівень	1 «Resources»	5 «Railway station»
Перший рівень конкретизації	2a «Table header», 2b «Table structure», 3a «Table software classification rules», 3b «Table type classification rules», 3c «Validation rules»	6a «Track profile», 6b «Track rail type», 6c «Track owners», 7a «TER», 7b «SBN»
Другий рівень конкретизацій	4a «Structure wrangling model», 4b «Table validation model»	8a «Track passport», 8b «Station Model»

Генерація моделі виконується в такій послідовності:

1) формуються абстрактні онтології (моделі-словники 1 і 5);
2) в онтології (моделі 3a-3c) імпортується модель 1:
3) в онтології 3a формуються логічні визначення типу «якщо таблиця заповнена в Microsoft Excel, то це таблиця класу excelTable» і описи класів скриптів з обмеженнями універсальності типу «скрипт має вхідні дані тільки таблиці профілю залізничної колії»;
4) в онтології 3b логічні визначення типу «якщо таблиця зроблена в Excel і таблиця має кортежі <ухил, висота бровки колії...> то таблиця має тип «таблиця профілю колії»», додаються посилання на процедури перетворення структури таблиці;
5) в онтології 3c формуються обмеження, у вигляді правила, наприклад, «якщо значення представлено літералом "помилка", то воно має властивість онтології Error [147]»;
6) модель 1 імпортується в онтологію (модель 2a) і формується опис заголовка, програми заповнення таблиці і скрипту для її обробки;
7) моделі 2a і 3a, 3b імпортуються в онтологію (модель 4a) і виконується перевірка на відповідність конкретній таблиці і процедурі перетворення,

якщо встановлена відповідність, то за запитом можна отримати скрипт для перетворення структури таблиці;

8) в OpenRefine імпортується модель 1, скрипт, отриманий на кроці 7, і таблиця (формат xls з обчисленням профілю, db з обмеженнями швидкості, csv з таблицею на кресленні: власники колії, відомість колій, останні видобуваються з pdf-файлу Tabula) і таблиця перетворюється. Формується модель 2b;
9) моделі 4a, 2b і 3c імпортуються в онтологію (модель 4b) і перевіряється валідність даних таблиці;
10) включають онтологічний висновок і при наявності невідповідностей слід виправити помилки в таблиці і повторити кроки 6-9;
11) в «bridging ontology», для інтеграції онтології джерел та залізничної інфраструктури імпортуються словники джерел (модель 1) та залізничної інфраструктури (модель 5) та формуються правила такого типу: «якщо екземпляр значення таблиці пов'язаний з ідентифікатором стовпця trackNumber, то цей екземпляр є типу railwayTrack»;
12) кроки 6-10 виконуються окремо для кожного типу таблиці (наприклад, такі типи, як відомість колій залізничної станції або книга попереджень про обмеження швидкості);
13) формуються моделі онтологій 6a-6c з фактами конкретних таблиць, таблиця із зазначенням власників колії (модель 6а), відомість колій (модель 6б), профіль колії (модель 6с). Для кожної таблиці створюється окрема онтологія, куди імпортується модель 4b і «bridging ontology». Здійснюється інтеграція онтологій джерел та залізничної інфраструктури;
14) онтологія-словник інфраструктури залізничної станції (модель 5) імпортується в онтологію конкретної моделі інфраструктури залізничної станції першого рівня (моделі 7а, 7б), де правила і обмеження формуються за ДБН [168] і ПТЕ [204] (наприклад, обмеження на величину ухилу під'їзної колії);

15) онтології з даними залізничної колії (моделі 6а-6с) і онтології ПТЕ [232] (модель 7а) імпортуються в конкретну онтологію паспорта залізничної колії другого рівня (модель 8а), таблиці зв’язуються між собою і під'їзна колія перевіряється на відповідність стандартам підприємств [232];
16) включають онтологічний висновок і при наявності невідповідностей виправляють помилки в таблиці і повторяють кроки 14-16;
17) онтологія паспорта колії (модель 8а), онтологія ДБН (модель 7б), імпортуються в онтологію конкретної моделі другого рівня станційної моделі Автоматизованої Системи Контролю Вантажних Перевезень (модель 8б) для перевірки колії на відповідність стандартам Укрзалізниці (УЗ) (оператора залізничної мережі загального користування). База даних компанії інтегрується з залізничними коліями та моделлю станції Укрзалізниці.

2.6 Автоматизована генерація онтологічного забезпечення документообігу обмеження швидкості поїзда

КПМ засноване на формальних граматиках та застосовується для формалізації формування конструкцій та конструктивних процесів різної природи. Основою є узагальнений конструктор, який представлений у [233]. Тут КПМ використовується для формалізації процедури розробки онтології залізничної колії на прикладі попередження про обмеження швидкості через його несправність.

У розділі представлений лише перший конструктор, наступні доступні в Додатку А.

Інформаційне забезпечення залізничної колії щодо несправностей – попередження про обмеження швидкості включає (Рисунок 2.16):

1) заявку на видачу попереджень із зазначенням місця та обмеження швидкості;
2) опис структури даних попередження ДУ-61 з Інструкції з руху поїздів [203];
3) бланк попередження.

Розробка онтології джерел даних для ДУ-61 передбачає такі етапи:

1) заповнення таблиці бланка ДУ-61 бази даних згідно з заявкою дорожнього майстра засобами SQLite ;

2) перетворення таблиці бланка ДУ-61 формату бази даних в єдине представлення засобами OpenRefine ;
3) заповнення таблиць формату csv згідно з моделлю табличного представлення знань [179] та таблиці ДУ-61 з ІРП;
4) перетворення таблиці ДУ-61 з ІРП у таблицю з метаданими формату CSV засобами текстового редактора;
5) перетворення таблиць формату csv в єдине представлення засобами OpenRefine ;
6) перетворення таблиць метаданих ДУ-61 з ІРП та моделі табличного представлення знань формату OpenRefine у словник схеми онтології засобами RDF extension OpenRefine ;
7) перетворення таблиці бланка ДУ-61 формату OpenRefine та словника онтології в екземпляри онтології (Abox онтології) засобами RDF extension OpenRefine ;
8) перетворення словника онтології на правила онтології (Tbox онтології) засобами Protégé ;
9) інтеграція схеми та екземплярів онтології засобами Protégé.

Розроблено конструктори кожного етапу розробки з виконавцями Protégé, OpenRefine, Tabula та SQLite, а також вхідними та вихідними даними (Рисунок 2.16). Продемонстровано порядок розробки конструкторів.

Для конструктора, що породжує, вхідні дані – це заявка на видачу попереджень, ІРП і модель табличного представлення знань. Вхідні дані перетворюючого і вихідні конструкторів, що породжує і перетворює, – це конструкції, одержувані за допомогою виводу.

Розглянемо конструктор формування таблиці бази даних попереджень, що породжує, про обмеження швидкості.

Мета конструктора – формування таблиці бази попереджень про обмеження швидкості.

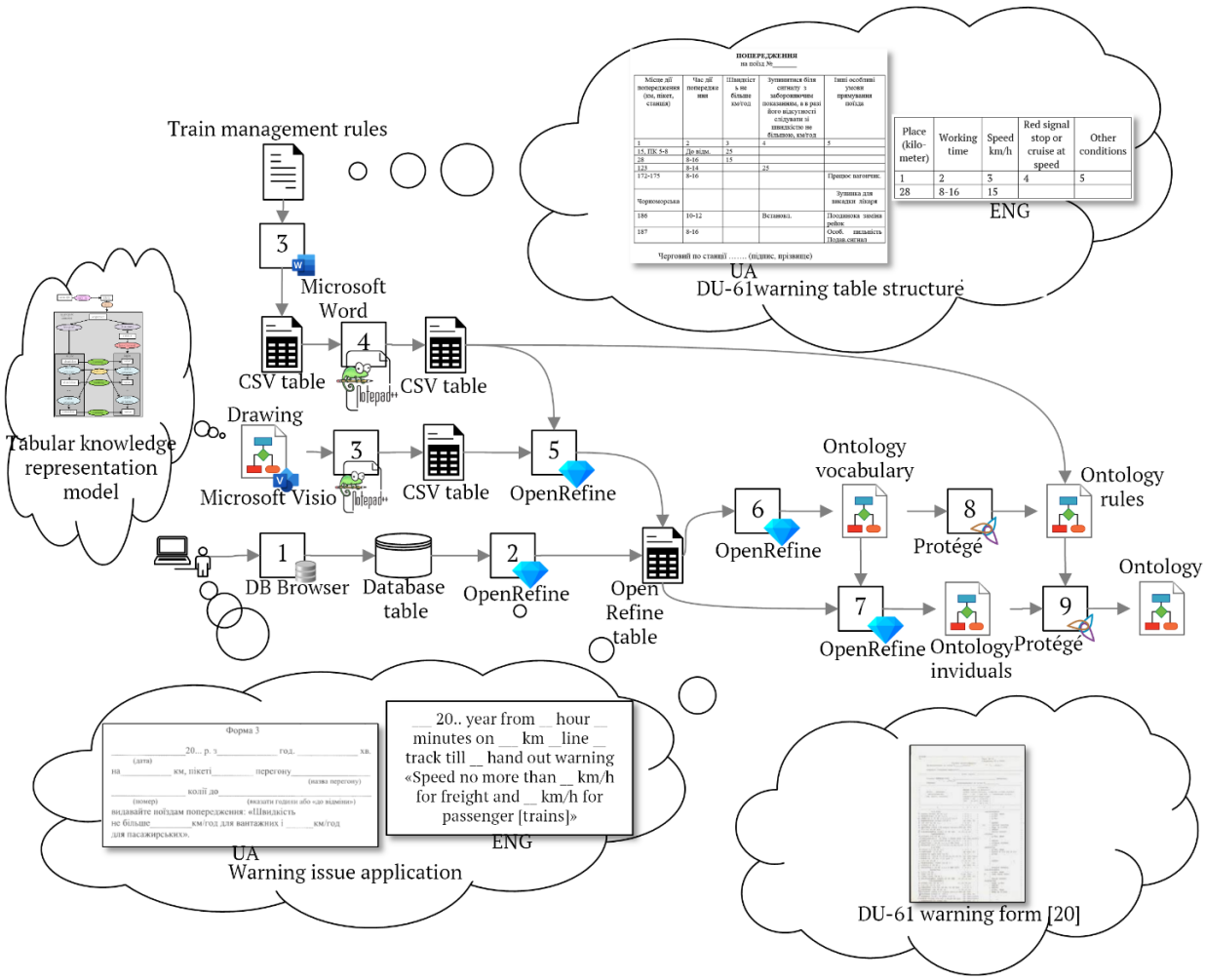


Рисунок 2.16 Порядок розробки конструкторів онтологічного забезпечення документообігу щодо обмеження швидкості руху поїздів

Виконаємо спеціалізацію узагальненого конструктора (ОК) на основі конструктивно-продукційного підходу:

$$C = \langle M, \Sigma, \Lambda \rangle \ {}_{S}\mapsto C_{tabl} = \langle M_{tabl}, \Sigma_{tabl}, \Lambda_{tabl} \rangle \qquad (2.1)$$

де C – ОК; М – неоднорідний носій ОК, що розширюється; Σ– сигнатура відношень та відповідних операцій ОК; Λ – безліч тверджень інформаційного забезпечення конструювання (ІЗК) ОК; ${}_{S}\mapsto$ – операція спеціалізації; C_{tabl} – спеціалізований конструктор (СК) формування таблиці попередження про обмеження швидкості; M_{tabl}включає множину терміналів СК $T = \{\Gamma\}$, де Γ – безліч строк, нетерміналів $N = \{TABLE, TUPLE3, TUPLE2, ELEMENT1, LIT, HEADER1\}$ правил підстановки; Σ_{tabl} – відношення та операції СК; Λ_{tabl}– інформаційне забезпечення

конструювання СК може включати семантику понять та операцій, правила підстановки, обмеження, початкові умови та умови завершення.

$\Sigma_{tabl} = \{\rightarrow, \Xi, \Theta, \Phi\}$ включає сигнатуру операцій: $\Xi = \{\circ\}$ – зв'язування; $\Theta = \{|\Rightarrow, ||\Rightarrow\}$ – підстановки; $\Phi = \{\#, @, :=, \cdot\}$ – операцій над атрибутами, $\rightarrow$ – відношення підстановки.

Семантика Λ_{tabl}включає поняття: ключ, ім'я таблиці, кількість стовпців (b), кількість рядків (f), значення атрибуту (LIT), таблиця, кортеж, відношення слідування $(\circ)$, а також [5] операції повного $(||\Rightarrow)$ та часткового $(|\Rightarrow)$ висновку, constrIndex, constrIndexPart – це строки.

Λ_{tabl}включає семантику операцій над атрибутами:

1) $\#(a, c, operation)$ – указана операція виконується при $a = c$;

2) $@(a)$ – отримання aвід зовнішнього виконавця;

3) $:= (a, b)$ – присвоєння aзначення b;

4) $\cdot (a, b, c)$ – конкатенації рядків bта cз результатом a.

Λ_{tabl} включає наступне обмеження: операція часткового виконується з урахуванням атрибуту відношення підстановки ($_t\rightarrow$) при t = 0 правило недоступне.

Правила підстановки визначаються за конкретизації конструктора.

Інтерпретуємо конструктор засобами алгоритмічного конструктора, виконавцем якого є DB Browser SQLite :

$$\langle C_{tabl} = \langle M_{tabl}, \Sigma_{tabl}, \Lambda_{tabl} \rangle\rangle, \tag{2.2}$$

$$C_{A,L} = \langle M_{A,tabl}, V_{A,tabl}, \Sigma_{A,tabl}, \Lambda_{A,tabl} \rangle \ {}_I\rightarrow$$
$$C_{I,tabl} = \langle M_{I,tabl}, \Sigma_{I,tabl}, \Lambda_{I,tabl} \rangle, \tag{2.3}$$

де $_I\rightarrow$ операція інтерпретації, $M_{A,tabl}$включає безліч алгоритмів, що реалізуються виконавцем, а також множин вхідних та вихідних даних для них, $\Sigma_{A,tabl} = \{\cdot, :\}$включає сигнатуру операцій послідовного та умовного виконання алгоритмів [233].

Нижче представлені алгоритми $V_{A,tabl}$, що реалізують відповідні операції над атрибутами: $A_1|_{a,c}^{A_j}$– алгоритм A_jвиконується при $a = c$; $A_2|_{\tilde{a}}^{a}$ – реалізація операції @;

$A_3|_b^a$- $a = b$; $A_4|_{b,c}^a$– $\cdot(a,b,c)$, де A_i– ідентифікатор алгоритму, X_i, Y_i– області визначення та значень алгоритму $A_i|_{Y_i}^{X_i}$.

Носій $\mathrm{M}_{I,tabl}$= $M_{tabl} \cup \mathrm{M}_{A,tabl}$.

Інформаційне забезпечення $\Lambda_{A,tabl}$представлено в [5], $\Lambda_{I,tabl} = \Lambda_{tabl} \cup \left\{(A_1|_{a,c}^{A_j} \downarrow \#), (A_2|_{a^\sim}^a \downarrow @), (A_3|_b^a \downarrow :=), \left(A_4|_{b,c}^a \downarrow \cdot\right)\right\}$.

Виконаємо конкретизацію конструктора:

$$C_{I,tabl} = \langle \mathrm{M}_{I,tabl}, \Sigma_{I,tabl}, \Lambda_{I,tabl} \rangle \ {}_K\!\rightarrow \qquad (2.4)$$

$$C_{K,tabl} = \langle \mathrm{M}_{K,tabl}, \Sigma_{K,tabl}, \Lambda_{K,tabl} \rangle,$$

де ${}_K\!\longrightarrow$ – операція конкретизації, $\mathrm{M}_{K,tabl}$= $\mathrm{M}_{I,tabl}$, $\Sigma_{K,tabl}$= $\Sigma_{I,tabl}$ включає $\Lambda_{K,tabl}$.

Початкові умови: лічильник заголовків таблиці headerCnt = 0,

кількість стовпців (b),

кількість рядків (f),

номер стовпця ключа keyColNum = 0,

лічильник рядків tupleCnt = 0,

лічильник значень elementCnt = 0,

конструйований конструктором індекс значення constrIndex constrIndexPart,

$t_0 = 1$, $t_1 = 0$, $t_2 = 0$, $t_3 = 0$, $t_4 = 0$, $t_5 = 0$, $t_6 = 0$, $t_7 = 0$, $t_8 = 0$, $t_9 = 0$, $t_{10} = 0$, $t_{11} = 0$.

Нетермінали : TABLE, TUPLE, TUPLE1, VECTOR, LIT, HEADER1, TUPLE2, TUPLE3, ELEMENt$_1$.

Термінали : b, f, key, table name, header, value, element, index.

Початковий нетермінал – TABLE.

Вхідні дані – заявка на видачу попередження, наприклад «на 23км видавайте поїздам попередження слідувати зі швидкістю не більше 60 км/год» та обмеження описані раніше.

Умова завершення – всі відношення підстановки недоступні;

Вихідні дані – таблиця бази попереджень (Таблиця 2.4);

Таблиця 2.4 Усічений бланк попередження ДУ-61

Place	Speed
23 км	60

Правила підстановки (включають відношення підстановки s та операції над атрибутами g):

$$s_1 = \left\langle {}_{LIT}attr\ TABLE\ \ {}_{t}\!\rightarrow TUPLE1 \circ VECTOR \circ {}_{LIT}attr\ TUPLE \right\rangle,$$

де ${}_{LIT}attr = {}_{LIT}b, {}_{LIT}f, {}_{LIT}key, {}_{LIT}table\ name.$

$g_1 = \langle := (t, 0), := (t_1 = 1), @(LIT \downarrow b \downarrow TABLE),$

$@(LIT \downarrow f \downarrow TABLE), @(LIT \downarrow key \downarrow TABLE),$

$@(LIT \downarrow table\ name \downarrow TABLE)\rangle.$

Правило s1. «ТАБЛИЦЯ» замінюється на «КОРТЕЖ1∘ВЕКТОР ∘ТАБЛИЦЯ ».

t_0 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_1 стає недоступним. Правило s_1 має бути недоступним, тому що в ланцюжку залишився нетермінал TABLE.

t_1 присвоюється 1. Тому правило s_2 стає доступним.

Зовнішнім виконавцем вводяться атрибути таблиці: кількість стовпців (b), кількість рядків (f), ключ, ім'я таблиці.

Оскільки доступно правило s_2 переходимо до правила s_2.

$t_0 = 0$, $t_1 = 1$, $t_2 = 0$, $t_3 = 0$, $t_4 = 0$, $t_5 = 0$, $t_6 = 0$, $t_7 = 0$, $t_8 = 0$, $t_9 = 0$, $t_{10} = 0$, $t_{11} = 0$.

Розглянемо приклад реалізації:

${}_{LIT}b, {}_{LIT}f, {}_{LIT}key, {}_{LIT}table\ name\ TABLE \overset{s_1}{\Rightarrow}$

Підстановка 1. $TUPLE1 \circ VECTOR \circ {}_{LIT}b, {}_{LIT}f, {}_{LIT}key, {}_{LIT}table\ name\ TABLE \overset{s_2}{\Rightarrow}$

$\langle t_0 = 0, t_1 = 1, LIT \downarrow b \downarrow TABLE = 2, LIT \downarrow f \downarrow TABLE = 1,$

$LIT \downarrow key \downarrow TABLE = Place,$

$LIT \downarrow table\ name \downarrow TABLE = \text{ДУ61}\rangle.$

$$s_2 = \langle TUPLE1\ \ {}_{t_1}\!\rightarrow {}_{LIT}value\ header \circ HEADER1\rangle,$$

$g_2 = \langle := (t_2, 1), +(headerCnt, headerCnt, 1),$

$@(LIT \downarrow value \downarrow header),$

$\#(LIT \downarrow value \downarrow header, key, key, := (keyColNum, headerCnt)),$

$\#(headerCnt, b, := (t_7, 1)),$

$\#(headerCnt, b, := (t_2, 0)), \#(headerCnt, b, := (t_3, 1))\rangle.$

Правило s_2. " TUPLE 1" замінюється на "заголовок ∘HEADER 1 ".

t_2 присвоюється 1. Тому правило s_3 стає доступним.

Збільшується лічильник заголовків headerCnt.

Зовнішнім виконавцем вводиться заголовок.

Якщо заголовок дорівнює ключу, то keyColumnNum присвоюється значення headerCnt.

Якщо лічильник headerCnt дорівнює кількість стовпців (b), то t_7 присвоюється 1. Завдяки цьому стає доступним правило s_8 для заміни нетерміналу HEADER 1 на ε.

Якщо лічильник headerCnt дорівнює кількість стовпців (b), то t_2 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_2 стає недоступним. Так як було досягнуто кількість стовпців початкової умови і більше не треба додавати заголовки до таблиці.

Якщо лічильник headerCnt дорівнює кількість стовпців (b), то t_3 присвоюється 1. Тому правило s_4 стає доступним. Переходять до введення значень.

Таким чином, є два випадки поведінки конструктивної системи.

Переходять до правила s_4 якщо в таблиці один стовпець

t_0 =0, t_1 =1, t_2 =0, t_3 =1, t_4 =0, t_5 =0, t_6 =0, t_7 =1, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 0

або перехід до правила s_3 якщо таблиці 2 і більше стовпців.

t_0 =0, t_1 =1, t_2 =1, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =0, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 0.

Продовжимо розглядати приклад реалізації:

Підстановка 2.

$$_{LIT}value\,header \circ HEADER1 \circ VECTOR \circ {}_{2}b,\ {}_{1}f,{}_{place}key,{}_{\text{ДУ}61}table\ name\,TABLE \overset{s_3}{\Rightarrow}$$

$$\langle t_2 = 1, headerCnt = 1, LIT \downarrow value \downarrow header = place, keyColNum = 1\rangle.$$

$$s_3 = \langle HEADER1 \ {}_{t_2}\!\rightarrow {}_{LIT}value\,header \circ HEADER1\rangle,$$

$$g_3 = \langle +(headerCnt, headerCnt, 1),$$

$$\#(headerCnt, b, t_2, 0),\ @(LIT \downarrow value \downarrow header),$$

$$\#\big(headerCnt, b, := (t_3, 1)\big),\ \#(headerCnt, b, := (t_7, 1)),$$

$$\#(LIT \downarrow value \downarrow header, lit \downarrow header \downarrow key, := (keyColNum, headerCnt))\rangle.$$

Правило 3. " HEADER 1" замінюється на "заголовок ∘HEADER 1 ".

Збільшується лічильник заголовків headerCnt.

Якщо лічильник headerCnt дорівнює кількість стовпців (b), то t_2 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_2 стає недоступним. Так як було досягнуто кількість стовпців початкової умови і більше не треба додавати заголовки до таблиці.

Зовнішнім виконавцем вводиться заголовок.

Якщо лічильник headerCnt дорівнює кількість стовпців (b), то t_3 присвоюється 1. Тому правило s_4 стає доступним. Переходять до введення значень.

Якщо лічильник headerCnt дорівнює кількість стовпців (b), то t_7 присвоюється 1. Завдяки цьому стає доступним правило s_8 для заміни нетерміналу HEADER 1 на ε.

Якщо заголовок (нетермінал з header ∘HEADER 1) дорівнює ключу (введеному зовнішнім виконавцем атрибуту таблиці), то keyColumnNum присвоюється значення headerCnt.

Таким чином, є два випадки поведінки конструктивної системи.

Переходять до правила s_4 якщо у таблиці два стовпці

t_0 =0, t_1 =1, t_2 =0, t_3 =1, t_4 =0, t_5 =0, t_6 =0, t_7 =1, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 0.

або перехід до правила s_3, якщо в таблиці 3 і більше стовпців.

t_0 =0, t_1 =1, t_2 =1, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =0, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 0.

Продовжимо розглядати приклад реалізації:

Підстановка 3. $_{place}value\,header \circ {}_{LIT}value\,header \circ HEADER1 \circ$

$VECTOR \circ {}_{2}b,\ {}_{1}f,_{place}key,_{\text{ДУ}61}table\ name\,TABLE \overset{s_4,s_8}{\Longrightarrow}$

$\langle headerCnt = 2, t_2 = 0, LIT \downarrow value \downarrow header = speed, t_3 = 1, t_7 = 1\rangle.$

$s_4 = \langle VECTOR\ {}_{t_3}\!\rightarrow TUPLE2 \circ TUPLE3\rangle,$

$g_4 = \langle := (t_5, 1), +(tupleCnt, tupleCnt, 1), \#(tupleCnt, f, := (t_8, 1))\rangle.$

Правило 4. " VECTOR " замінюється на " TUPLE 2 ∘TUPLE 3 ".

t_5 присвоюється 1. Тому правило s_6 стає доступним.

Збільшується лічильник tupleCnt.

Якщо лічильник tupleCnt дорівнює кількості рядків (f), то t_8 присвоюється 1. Завдяки цьому стає доступним правило s_9 для заміни нетерміналу TUPLE 3 на ε. Оскільки було досягнуто кількість рядків початкової умови, більше додавати рядки непотрібно.

Таким чином, є два випадки поведінки конструктивної системи.

Переходять до правила s_9 якщо у таблиці один кортеж

t_0 =0, t_1 =1, t_2 =0, t_3 =1, t_4 =0, t_5 =1, t_6 =0, t_7 =1, t_8 =1, t_9 =0, t_{10} = 0, t_{11} = 0.

Або поки не переходять до правила s_9 якщо таблиці 2 і більше кортежів.

t_0 =0, t_1 =1, t_2 =0, t_3 =1, t_4 =0, t_5 =1, t_6 =0, t_7 =1, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 0

Продовжимо розглядати приклад реалізації:

Підстановка 4. ${}_{place}value\,header \circ {}_{speed}value\,header \circ TUPLE2 \circ TUPLE3 \circ {}_{2}b,\ {}_{1}f,{}_{place}key,{}_{\text{ду}61}table\ name\,TABLE \overset{s_6,s_9}{\Longrightarrow}$

$\langle t_5 = 1, tupleCnt = 1, t_8 = 1\rangle.$

$s_5 = \langle TUPLE3 \ {}_{t_4}\!\rightarrow TUPLE2 \circ TUPLE3\rangle,$

$g_5 = \langle := (t_5, 1), +(tupleCnt, tupleCnt, 1), := (t_4, 0),$
$\#(tupleCnt, f, := (t_8, 1))\rangle.$

$s_6 = \langle TUPLE2 \ {}_{t_5}\!\rightarrow \ {}_{LIT}attr\,element \circ ELEMENT1\rangle,$

де ${}_{LIT}attr\,element = {}_{LIT}type,{}_{LIT}index,{}_{LIT}value\,element.$

$g_6 = \langle := (elementCnt, 1), @(lit \downarrow value \downarrow element),$
$\#(elementCnt, headerCnt, := (t_9, 1)), : = (t_6, 1),$
$\#\big(elementCnt, headerCnt, := (t_{10}, 1)\big),$
$\#(elementCnt, keyColNum,$
$:= (constrIndexPart, LIT \downarrow value \downarrow element)),$
$\#(elementCnt, keyColNum,$
$\cdot (constrIndex, constrIndexPart, LIT \downarrow table\ name)),$
$\#(elementCnt, headerCnt, := (\ elementCnt, 0))\big)\rangle.$

Правило s_6. «TUPLE2» замінюється на "Element ∘ELEMENt$_1$ ".

Збільшується лічильник elementCnt.

Зовнішнім виконавцем вводиться значення.

Якщо лічильник elementCnt дорівнює headerCnt, то t_9 присвоюється 1. Завдяки цьому стає доступним правило s_{10} для заміни нетерміналу ELEMENt$_1$ на ε. Так як було досягнуто кількість елементів, що дорівнює headerCnt, більше додавати елементи не треба.

t_6 присвоюється 1. Тому правило s_7 стає доступним. Слід зазначити, що якщо завдяки попередній операції при elementCnt дорівнює headerCnt нетермінал ЕЛЕМЕНТ1 замінений на ε, то правило s_7 недоступне.

Якщо лічильник elementCnt дорівнює headerCnt, на правилі s_6, то t_{10} присвоюється 1. Оскільки кортежі 1 елемент, і є ключ. Правило s_{11} стає доступним.

Якщо лічильник elementCnt дорівнює keyColumnNum, змінній constrIndexPart присвоюється значення елемента таблиці.

Якщо лічильник elementCnt дорівнює keyColumnNum, то змінній constrIndex присвоюється результат конкатенації constrIndexPart та імені таблиці.

Якщо лічильник elementCnt дорівнює headerCnt, то elementCnt присвоюється 0. Переходять до підстановки індексів елементів таблиці.

Таким чином, є два випадки поведінки конструктивної системи.

Переходять до правила s_{11} якщо у таблиці 1 стовпець

t_0 =0, t_1 =1, t_2 =0, t_3 =1, t_4 =0, t_5 =1, t_6 =1, t_7 =1, t_8 =1, t_9 =1, t_{10} = 1, t_{11} = 0.

або перехід до правила s_7 якщо таблиці 2 і більше стовпців.

t_0 =0, t_1 =1, t_2 =0, t_3 =1, t_4 =0, t_5 =1, t_6 =1, t_7 =1, t_8 =1, t_9 =0, t_{10} = 0, t_{11} = 0.

Продовжимо розглядати приклад реалізації:

Підстановка 5. ${}_{place}value\,header \circ {}_{speed}value\,header \circ$

${}_{LIT}type, {}_{LIT}index, {}_{LIT}value\,element \circ ELEMENT1 \circ {}_{2}b,\ {}_{1}f, {}_{place}key, {}_{\text{ДУ61}}table\ name\,TABLE \overset{s_7}{\Rightarrow}$

$\langle elementCnt = 1, LIT \downarrow value \downarrow element = 23k\text{м},$

$t_6 = 1, j = 23km, constrIndex = 23km\text{ДУ61}\rangle.$

$s_7 = \langle ELEMENT1 \ {}_{t_6}\!\rightarrow \ {}_{LIT}attr\, element \circ ELEMENT1\rangle,$

$g_7 = \langle +(elementCnt, elementCnt, 1),$

$\#(elementCnt, headerCnt, := (t_6, 0)),$

$@(LIT \downarrow value \downarrow element), \#\big(elementCnt, headerCnt, := (t_{10}, 1)\big),$

$\#(elementCnt, keyColNum,$

$:= (constrIndexPart,\ LIT \downarrow value \downarrow element)\big),$

$\#(elementCnt, keyColNum,$

$\cdot\,(constrIndex, constrIndexPart, LIT \downarrow table\ name \downarrow table)),$

$\#(elementCnt, headerCnt, := (\ elementCnt, 0))\big\rangle.$

Правило s_7. «ELEMENt$_1$» замінюється на "Element ∘ELEMENt$_1$ ".

Збільшується лічильник elementCnt.

Якщо лічильник elementCnt дорівнює headerCnt, то t_6 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_7 стає недоступним.

Якщо лічильник elementCnt дорівнює headerCnt, то t_9 присвоюється 1. Завдяки цьому стає доступним правило s_{10} для заміни нетерміналу ELEMENt$_1$ на ε. Так як було досягнуто кількість елементів, що дорівнює headerCnt, більше додавати елементи не треба.

Зовнішнім виконавцем вводиться значення.

Якщо лічильник elementCnt дорівнює headerCnt, то t_{10} присвоюється 1. Правило s_{11} стає доступним.

Якщо лічильник elementCnt дорівнює keyColumnNum, змінній constrIndexPart присвоюється значення елемента таблиці.

Якщо лічильник elementCnt дорівнює keyColumnNum, то змінній constrIndex присвоюється результат конкатенації constrIndexPart та імені таблиці.

Якщо лічильник elementCnt дорівнює headerCnt, то elementCnt присвоюється 0. Переходять до підстановці індексів елементів таблиці.

Таким чином, є два випадки поведінки конструктивної системи.

Переходять до правила s_{11} якщо у таблиці 2 стовпця

t_0 =0, t_1 =1, t_2 =0, t_3 =1, t_4 =0, t_5 =1, t_6 =0, t_7 =1, t_8 =1, t_9 =1, t_{10} = 1, t_{11} = 0.

або перехід до правила $s_{7,}$ якщо в таблиці 3 і більше стовпців.

t_0 =0, t_1 =1, t_2 =0, t_3 =1, t_4 =0, t_5 =1, t_6 =1, t_7 =1, t_8 =1, t_9 =0, t_{10} = 0, t_{11} = 0.

Продовжимо розглядати приклад реалізації:

Підстановка 6. $_{place}value\,header \circ {}_{speed}value\,header \circ$

$_{string}type,_{LIT}index,_{23км}value\,element \circ$

$_{LIT}type,_{LIT}index,_{LIT}value\,element \circ ELEMENT \circ {}_{2}b,\ {}_{1}f,_{place}key,_{ДУ61}table\ name\,TABLE \overset{s_{11}}{\Rightarrow}$

$\langle elementCnt = 2, t_5 = 0, LIT \downarrow value \downarrow element = 60,$

$t_{10} = 1, elementCnt = 0\rangle.$

$s_8 = \langle HEADER1\ {}_{t_7}\!\rightarrow \varepsilon\rangle,$

$g_8 = \langle\varepsilon\rangle.$

$s_9 = \langle TUPLE3\ {}_{t_8}\!\rightarrow \varepsilon\rangle,$

$g_9 = \langle\varepsilon\rangle.$

$s_{10} = \langle ELEMENT1\ {}_{t_9}\!\rightarrow \varepsilon\rangle,$

$g_{10} = \langle\#(tupleCnt, f\ AND\ elementCnt, headerCnt,$

$:= (t_{11}, 1)\big), := (t_4, 1), := (t_5, 0),$

$:= (t_8, 0), := (t_9, 0)\rangle.$

Правило $s_{10.}$ «ЕЛЕМЕНТ1» замінюється на ε.

Якщо лічильник tupleCnt дорівнює кількості рядків (f) і лічильник elementCnt дорівнює лічильнику headerCnt, то t_{11} присвоюється 1. Тому правило s_{12} стає доступним.

t_4 присвоюється 1. Тому правило s_5 стає доступним. Слід зазначити, що, якщо у правилі s_4 була зроблена заміна нетермінала TUPLE 3 на ε, то правило s_5 застосувати не вдасться.

t_5 присвоюється 0.

t_8 присвоюється 0.

t_9 присвоюється 0.

Таким чином, є два випадки поведінки конструктивної системи.

Переходять до правила s_{12} якщо у таблиці 1 кортеж

$t_0=0, t_1=1, t_2=0, t_3=1, t_4=0, t_5=1, t_6=0, t_7=1, t_8=1, t_9=1, t_{10}=0, t_{11}=1.$

або перехід до правила s_4, якщо в таблиці більше 1 кортежу.

$t_0=0, t_1=1, t_2=0, t_3=1, t_4=1, t_5=0, t_6=0, t_7=1, t_8=0, t_9=0, t_{10}=0, t_{11}=0.$

Продовжимо розглядати приклад реалізації:

Підстановка 9. ${}_{place}value\,header \circ {}_{speed}value\,header \circ$

${}_{string}type,_{23км ДУ61}index,_{23км}value\,element \circ$

${}_{string}type,_{23км ДУ61}index,_{60}value\,element \circ {}_{2}b,\ {}_{1}f,_{place}key,_{ДУ61}table\ name\,TABLE \overset{s_{12}}{\Rightarrow}$

$\langle t_{11}=1 \rangle.$

$s_{11}=\langle LIT \downarrow index\ {}_{t_{10}} \rightarrow n \rangle,$

$g_{11}=\langle +(elementCnt, elementCnt, 1),$

$\#\big(elementCnt, headerCnt, := (t_{10}, 0)\big),$

$\#\big(elementCnt, headerCnt, := (t_9, 1)\big)\big\rangle,$

Правило s_{11}. Оскільки elementCnt присвоїли 0, виконаний до початку кортежу. Виконується послідовна заміна нетерміналу LIT атрибут індекс елементів таблиці.

При кожній ітерації збільшується лічильник elementCnt.

Якщо лічильник elementCnt дорівнює headerCnt, то t_{10} присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_{11} стає недоступним.

Якщо лічильник elementCnt дорівнює headerCnt, то t_9 присвоюється 1. Правило s_{10} стає доступним.

Таким чином, є два випадки поведінки конструктивної системи.

Переходять до правила s_{11} при послідовній заміні нетерміналу LIT

$t_0=0, t_1=1, t_2=0, t_3=1, t_4=0, t_5=1, t_6=0, t_7=1, t_8=1, t_9=0, t_{10}=1, t_{11}=0.$

або перехід до правила s_9 якщо лічильник elementCnt дорівнює headerCnt.

$t_0=0, t_1=1, t_2=0, t_3=1, t_4=0, t_5=1, t_6=0, t_7=1, t_8=1, t_9=1, t_{10}=0, t_{11}=0.$

Продовжимо розглядати приклад реалізації:

Підстановка 7. ${}_{place}value\,header \circ {}_{speed}value\,header \circ$

${}_{string}type,_{23км ДУ61}index,_{23км}value\,element \circ$

${}_{string}type,_{LIT}index,_{60}value\,element \circ ELEMENT1 \circ {}_{2}b,\ {}_{1}f,_{place}key,_{ДУ61}table\ name\,TABLE \overset{s_{11}}{\Rightarrow}$

$\langle elementCnt = 1 \rangle$.

Підстановка 8. ${}_{place}value\,header \circ {}_{speed}value\,header \circ$

${}_{string}type,_{23км ДУ61}index,_{23км}value\,element \circ {}_{string}type,_{23км ДУ61}index,_{60}value\,element \circ$

$ELEMENT1 \circ {}_{2}b,\ {}_{1}f,_{place}key,_{ДУ61}table\ name\,TABLE \overset{s_{10}}{\Rightarrow}$

$\langle elementCnt = 2, t_9 = 1, t_{10} = 0 \rangle$.

$s_{12} = \langle TABLE\ {}_{t_{11}} \rightarrow eof \rangle$,

$g_{12} = \langle \varepsilon \rangle$.

Підстановка 10. ${}_{place}value\,header \circ$

${}_{speed}value\,header \circ {}_{string}type,_{23км ДУ61}index,_{23км}value\,element \circ$

${}_{string}type,_{23км ДУ61}index,_{60}value\,element \circ eof$

Висновки до другого розділу

Виконана формалізація онтологічних елементів табличної структури даних та їх зв'язків у вигляді графічної нотації формальної мови. Запропоновано метод формування складних понять онтологічними засобами. Концептуалізація здійснюється на основі глибокої деталізації взаємозв'язків між різними поняттями.

Ця концепція заснована на застосуванні відношення порядку, варіативність якого дозволяє моделювати різні реальні об'єкти різної природи.

Розроблено метод семантичного контролю, що виконується на основі моделі багаторівневої конкретизації та інтеграції онтологій джерел та залізничної інфраструктури.

Запропоновано спосіб автоматизації видобування та перетворення даних різнорідних документів, що описують станційні колії.

Запропоновано підхід формалізації обмежень нормативно-правової документації методами анотування природно-мовних текстів а також розроблено онтологічні визначення нормативної документації для поєднання моделей АСК ВП УЗ-Є.

Закладено основу для автоматизації формування онтології залізничного домену конструктивно-продукційним методом. Науково обґрунтовано можливості використання в онтологічному домені на залізничному транспорті конструктивно-продукційного моделювання для: автоматизованого формування онтологій залізничного домену від джерел таблиць баз даних до схеми і екземплярів онтології та бази знань.

Набули розвитку методи ontology learning для інтеграції даних різноманітних джерел. Розроблений метод дозволяє виконати інтеграцію різноманітних джерел даних (структури таблиці ДУ-61 ІРП, бланка та заявки на видачу попередження), а також предметно орієнтований на залізницю. Дозволяє сформувати онтологію від її джерел даних (формату бази даних та csv) до схеми та екземплярів. Виконана формалізація формування онтології, що дозволяє розширити можливості автоматизації інтеграції та узгодження даних засобами онтологій.

За результатами розділу, опубліковано роботи [173, 178, 179, 180].

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ ОНТОЛОГІЙ

Для вирішення завдань дисертаційної роботи і реалізації методів, описаних в розділі 2, розроблені онтології джерел і залізничної інфраструктури. Вони дозволяють перевірити структуру даних і узгодженість з нормативним забезпеченням.

Метою цього етапу є:

1) семантична перевірка та узгодження даних інформаційного забезпечення під'їзної колії з вимогами нормативної документації на основі моделі багаторівневої конкретизації та інтеграції онтологій;
2) зв'язування значень швидкості на залізничній колії з наказу про встановлення допустимих швидкостей на елементах залізничної інфраструктури з характеристиками колії та виконання перевірки узгодженості даних таблиць з формалізованими обмеженнями правил технічної експлуатації (ПТЕ) [204] та державних будівельних норм (ДБН) [168] на основі онтологічних засобів;
3) підвищити узгодженість даних різних підсистем та технологічних ланцюгів обробки даних про несправності залізничної колії без змін баз даних та програмного забезпечення підсистем залізничної колії та управління рухом поїздів.

3.1 Реалізація моделей онтологій абстрактних та конкретних з правилами

Прототип розробленого онтологічного забезпечення семантичного контролю паспорта під'їзної колії включає базу знань з 33 класами, 24 відношеннями та 72 екземплярами. Екземпляри стосуються однієї під'їзної колії залізничної станції. Масштаби інформаційної системи Укрзалізниці можуть бути представлені таким чином: залізниці України складаються з 6 підрозділів, кожен з яких включає близько 300 залізничних станцій. Вузлові залізничні станції можуть мати десятки під'їзних колій.

3.1.1 Реалізація абстрактних і конкретних з правилами resources model-ей

Онтологія абстрактної моделі джерел 1 являє собою єдину схему опису таблиць різних форматів і задовольняє визначення легкої онтології [70] в тому, що містить поняття і прості відношення, які пов'язують їх між собою. Кожен формат даних має свої особливості, які враховуються при конкретизації.

На Рисунку 3.1 представлений словник онтології джерел з використанням нотації Graffoo [58]. «Жовті прямокутники представляють класи (суцільний контур) і обмеження (пунктирний контур), зелені паралелограми представляють типи даних, стрілки, що починаються із зафарбованого кола, представляють об’єктні відношення, стрілки, що починаються з незафарбованого кола, представляють відношення даних, а інші стрілки відображають твердження між ресурсами» [25].

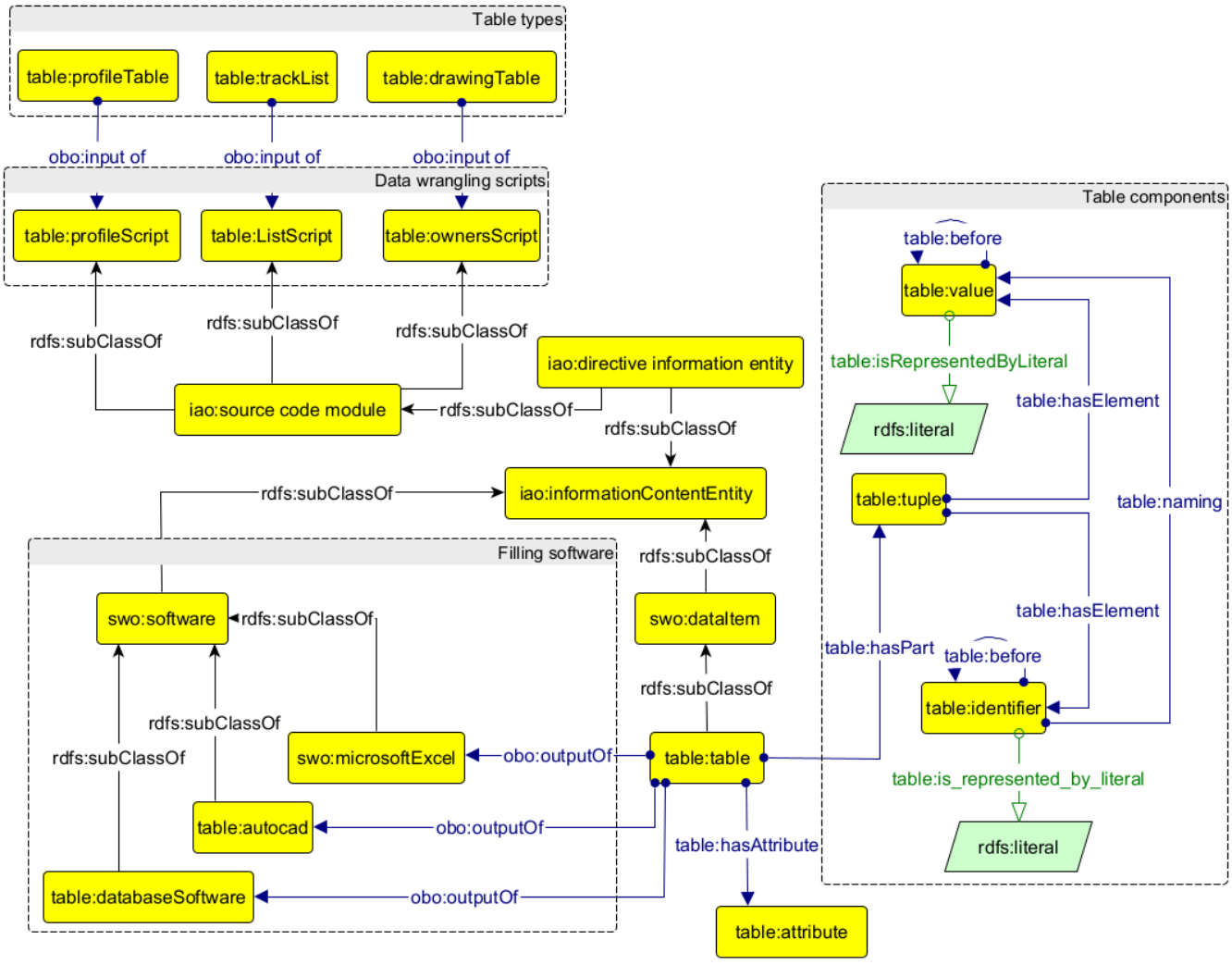


Рисунок 3.1 Легка онтологія-словник джерел даних

В онтології використовуються поняття з Information artifact ontology [23], Relational Ontology [191], що є частиною The Open Biological and Biomedical Ontology (OBO) Foundry, и SWO [111]:

1) rdfs: <http://www.w3.org/2000/01/rdf-schema#>;

2) iao: <http://purl.obolibrary.org/obo/iao.owl#>;

3) obo: <http://purl.obolibrary.org/obo/>;

4) swo: <http://www.ebi.ac.uk/swo/>;

5) table: <http://www.semanticweb.org/larisk0/ontologies/2022/2/table#>.

Контуром «filling software» на Рисунку 3.1 виділено концепти програмного забезпечення для заповнення таблиць, контуром «data wrangling scripts» – скрипти перетворення вихідних даних у екземпляри онтології, контуром «table components» – компоненти табличного представлення знань, а контуром «table types» – види (наприклад, відомості та профілі колій) таблиць вихідних даних.

В контурі «table components» Рисунку 3.1 показано концепти для представлення структури таблиць: класи «таблиця», «кортеж», «ідентифікатор» (у випадку з таблицею це назва стовпця) і «значення». Відношення «hasPart» зв'язує таблицю з кортежами. Це відношення використовується тому, що таблиця не існує без кортежів з епістемологічної точки зору. Відношення «hasElement» зв'язує кортеж зі значеннями та ідентифікаторами, «naming» – ідентифікатори зі значеннями. Відношення «before» – це окремий випадок відношення порядку, необхідного так як кортеж – це упорядкована множина. Він конкретизується для кожної таблиці і кожної пари стовпців, наприклад, для таблиці профілю і відомості колій ієрархія відношень виглядає наступним чином (Рисунок 3.2).

Загальна модель [179] спеціалізована для таблиць Excel класами «profileTable», а також відношенням «hasAttribute» і класом атрибутів для зв’язку значень, що належать до різних таблиць. Відношення конкретизується як enterpriseAttribute і trackAttribute для зв'язку відомості колій з таблицею із зазначенням власників колії і профілю колії відповідно. Клас інструментів Microsoft Excel імпортується із software ontology [111].

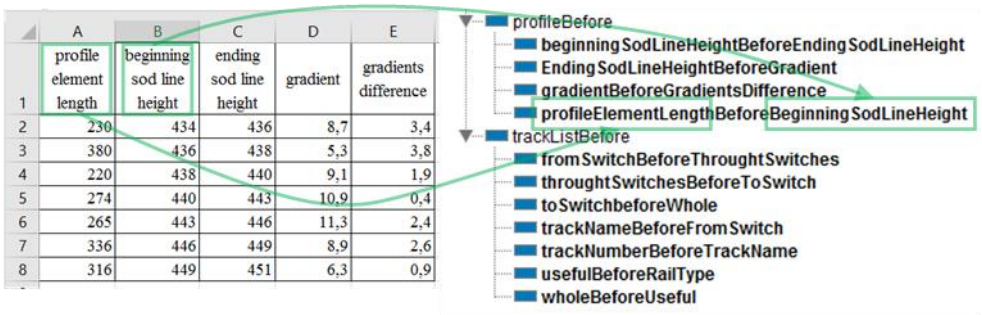


Рисунок 3.2 Специфікація зв'язку замовлення для профільних таблиць і списку залізничних колій

В контурі «filling software» Рисунку 3.1 представлено програми Таблиці 2.1 і Рисунка 2.2. Це типові програми (Excel, AutoCAD і деякі СКБД), що використовуються для обробки даних на залізниці. Як в SWO [111] використовується відношення «output of» для зв'язку вихідних даних і програмного забезпечення.

В контурі «table types» Рисунка 3.1 представлено таблиці вихідних даних Рисунку 2.14 (профіль і відомість колій, а також таблиця із вказанням власників). У контурі «data wrangling scripts» Рисунку 3.1 представлені скриптові програми для перетворення даних кожного виду таблиць рожевого контура (ownersScript, profileScript, listScript). Так само, як і в OntoDM [143] використовується відношення «input of» для зв'язування скриптової програми і вхідних даних.

Конкретна модель джерел з правилами 3 виконана у вигляді трьох онтологій 3a-3c. Онтологія імпортує онтологію абстрактної моделі джерел 1 і error ontology [147].

В онтології 3a представлені логічні визначення для класифікації екземплярів таблиці за інструментом її виконання (наприклад, клас ExcelTable на Рисунку 3.3), а також обмеження класу скриптів для перетворення структури таблиці (наприклад, клас profileScript на Рисунку 3.3). А в онтологія 3b представлені логічні визначення для класифікації екземпляра таблиці за ідентифікаторами заголовків (наприклад, клас таблиці профілю Рисунку 3.3).

Онтологія 3c містить правило, за яким таблиці присвоюється властивість структурної помилки на основі екземплярів таблиці, перетвореної в OpenRefine. Правило (Рисунок 3.3) розроблено на мові Semantic Web Rule Language (SWRL).

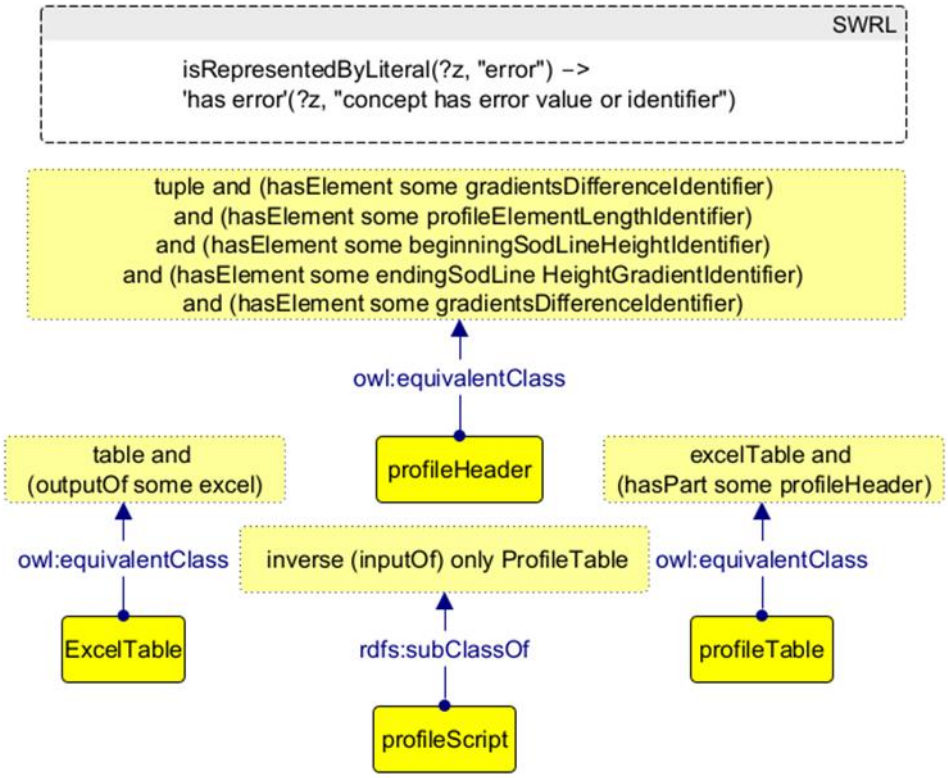


Рисунок 3.3 Приклади конструкцій конкретної моделі джерел з правилами 3

3.1.2 Інтеграція моделей джерел і залізничної інфраструктури

Виконується відображення схем онтології для інтеграції онтологій джерел і залізничної інфраструктури (Рисунок 3.4). В «bridging ontology» імпортуються онтології Concrete data resources model і Concrete rules resources model (онтологій 5 і 1 Рисунок 2.15).

На Рисунку 3.4 конструкція (f) не відповідає позначенню Graffoo [58] і розроблена згідно з рекомендаціями до зручних для пошуку, доступних, сумісних та багаторазових зв’язаних даних (Findable, Accessible, Interoperable and Reusable linked data) [31].

Класи зв'язуються відношеннями owl:equivalentClass (Рисунок 3.4 конструкція d)) і логічними визначеннями (Рисунок 3.4 конструкція e)).

Відношення зв'язуються між собою відношеннями owl:equivalentProperty (Рисунок 3.4 конструкція f)) та композиції (Рисунок 3.4 конструкції b) – c)).

Рисунок 3.4 Приклади аксіом для інтеграції онтологій

Класи і відношення онтологій джерел і залізничної інфраструктури зв'язані так, що різонер (reasoner) пере-класифікує екземпляри онтології Concrete resources model (4) як екземпляри онтології Abstract railway infrastructure model (5). Відображення виконує наступні завдання:

1) пере-класифікація екземплярів класу «valueClass» до класу онтології залізничної інфраструктури, що відповідає значенню стовпця таблиці. Це виконується за допомогою відношення «naming» онтології джерел, яка зв'язує ідентифікатори та значення таблиці (Рисунок 3.1) і продемонстровані на класі «railwayTrack» конструкцією e) Рисунок 3.4;
2) зв'язування нових пере-класифікованих екземплярів відношеннями онтології залізничної інфраструктури. При зв'язуванні екземплярів, які належать до

однієї таблиці, це виконується за допомогою відношення порядку «before», що зв’язує значення одного кортежу (Рисунок 3.1) і продемонстровано на відношенні «hasRailType» конструкцією c) Рисунок 3.4.

Для екземплярів різних таблиць (таблиці з власниками інфраструктури та профілем колії) використовується відношення hasTrackAttribute і hasEnterpriseAttribute (Рисунок 3.1). При зв'язуванні таблиці, із сказанням власників і відомості колій, екземпляр колії зв'язується з екземпляром класу enterpriseAttribute відповідно до композиції відношень відношенням «hasEnterpriseAttribute» (конструкція a) Рисунок 3.4). Тому досить пов'язати їх як еквівалентні відношення hasEnterpriseAttribute і connectsEnterprise, що продемонстровано конструкцією e) Рисунок 3.4.

У випадку таблиці профілю відношення gradientBeforeGradientsDifference використовується як вказівник на стовпець «Gradients difference», що продемонстровано конструкцією b) Рисунок 3.4.

3.1.3 Реалізація абстрактних і конкретних з правилами моделей залізничної інфраструктури

Онтологія-словник абстрактної моделі інфраструктури залізничної станції 5 являє собою єдину схему опису залізничної інфраструктури різних джерел (наприклад, джерел опису профілю колії, типів рейок і власників). Словник включає в себе наступні класи: «railwayTrack», «railType», «enterprise», «gradientsDifference» і відношення: «hasRailType», «connectsEnterprise», «hasGradientsDifference».

Словник також містить визначення термінів з Інструкції про рух поїздів на залізницях України, зв'язаних з класами відношенням skos:definition імпортованої онтології SKOS.

Конкретна модель інфраструктури залізничної станції з правилами 7 розроблена для випадків перевірки паспорта під'їзної колії, перевірки заявленої швидкості справної і несправної колії загального користування.

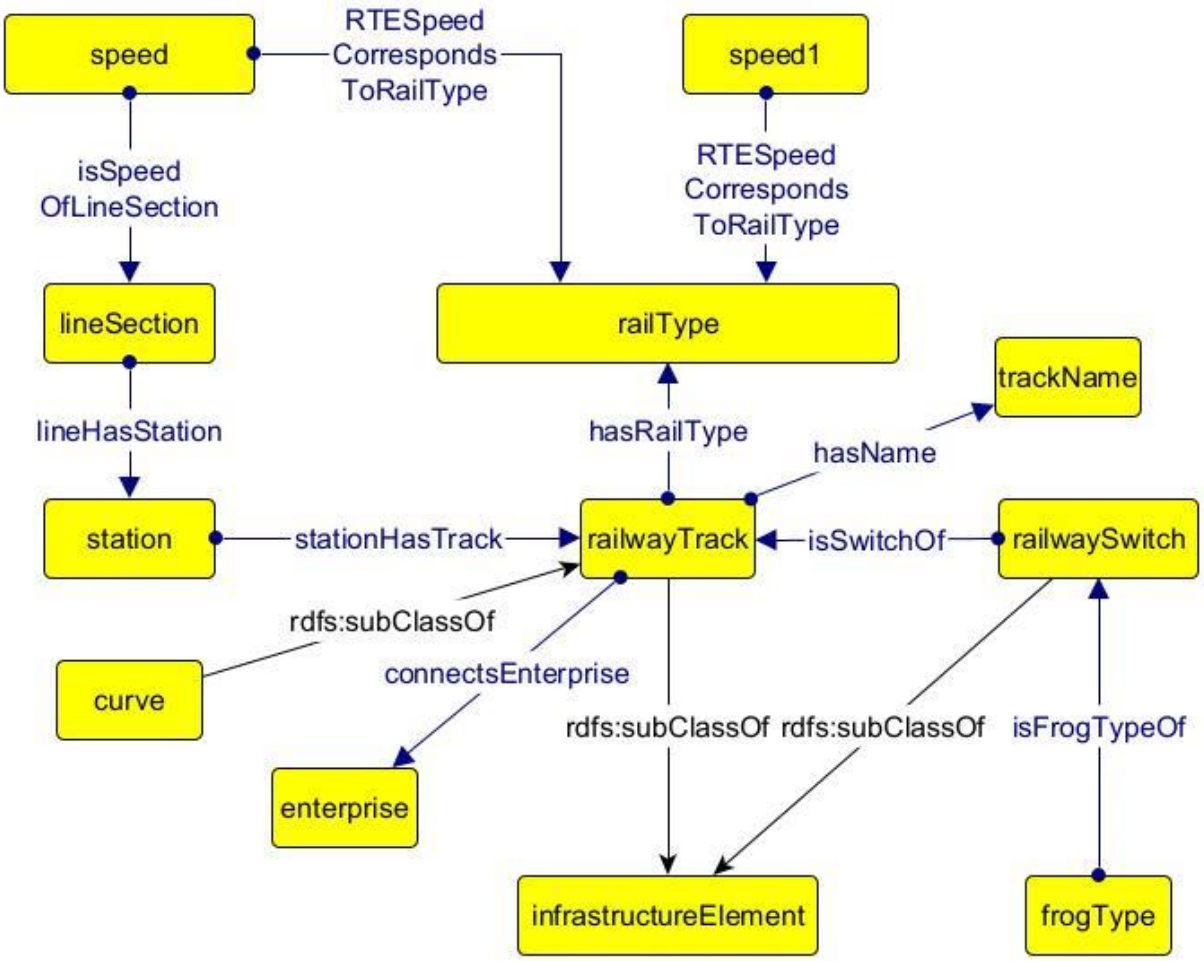


Рисунок 3.5 Словник інфраструктури залізничного станції в частині під'їзної колії

Для перевірки паспорта під'їзного колії – у вигляді двох онтологій 7a-7b. В онтологію імпортується абстрактна модель залізничної інфраструктури (онтологія 5 Рисунок 2.15).

Онтологія розроблена для інтеграції баз даних підприємств клієнтів та залізниці. Розроблено два різних пакети нормативних документів для Укрзалізниці та промислових станцій. Тому колії, що належать підприємству, та колії, що належать до інформаційних систем Укрзалізниці, мають відповідати різним обмеженням ПТЕ для підприємств [232] та ДБН [168] відповідно. Онтології баз даних інтегруються з використанням логічних визначень, наведених на Рисунку 3.6.

В онтології моделі 7a формалізовані обмеження ПТЕ [232] на кількість підприємств, з'єднаних під'їзною колією. В онтології моделі 7b формалізовані три типи обмежень ДБН [168].

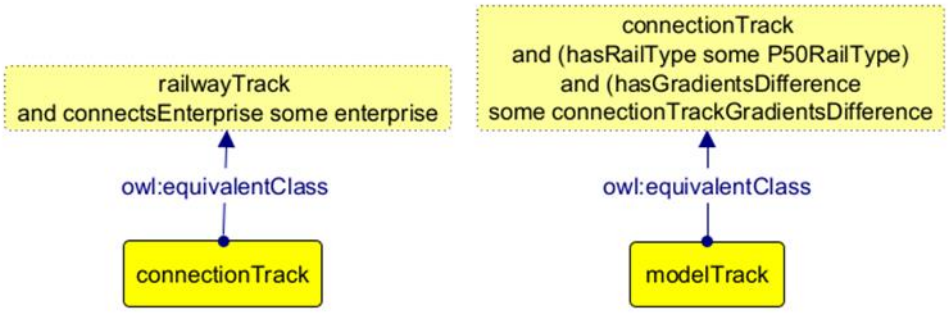


Рисунок 3.6 Приклади правил онтології інфраструктури залізничного вокзалу

На Рисунку 3.6 колія підприємства представлена класом «connectionTrack» модель 7a, колія додана до станційної моделі інформаційної системи Укрзалізниці – класом «modelTrack» моделі 7b.

Для перевірки швидкості справної колії загального користування розроблена онтологія конкретної моделі залізничної інфраструктури з правилами в Protégé.

Застосовано конструкцію owl:hasKey для data property speedIsRepresentedByLiteral, categoryIsRepresentedByLiteral, wordIDIsRepresentedByLiteral. Тому відношення стає обернено-фукціональним (inverse fuctional) і всі значення швидкості з однаковим значенням зв'язуються різонером (reasoner) відношенням owl:sameAs, що дозволяє використовувати композиції відношень Рисунку 3.7.

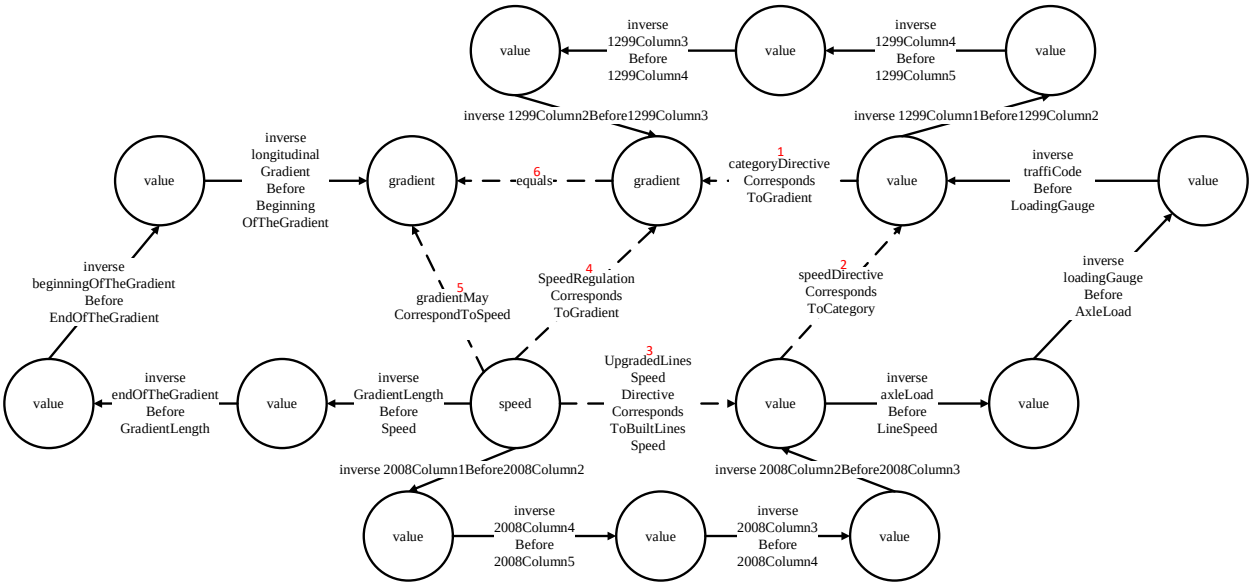


Рисунок 3.7 Взаємозв'язок складу для перевірки узгодженості даних інструкції та інформаційної системи

Композиціями відношень зв'язуються таблиці анотацій Directive 2008/57/EC [55] і Commission Regulation (EU) No 1299/2014 [27] і таблиця Performance parameters for passenger traffic з Commission Regulation (EU) No 1299/2014 оскільки відповідь на питання про допустимі ухили міститься в твердженнях про відповідність «ухилу і категорії колії», «категорії колії і швидкості», а також «швидкості новозбудованих і реконструйованих ліній».

```
Speed Directive Corresponds To Gradient SuperProperty Of upgraded Lines Speed Directive Corresponds To Built Lines Speed o speed Directive Corresponds To Category o category Directive Corresponds To Gradient
```

Зв’язуються ухили опису інфраструктури і нормативного документа композицією відношень equals.

```
equals SuperProperty Of gradient May Correspond To Speed o inverse speed Regulation Correspond To Gradient
```

Після цього вони порівнюються за приналежністю до одного інтервалу значень.

Для перевірки відповідності марки хрестовини і призначення колії стрілочного переводу розроблені композиції відношень, за допомогою яких таблиці відомості стрілочних переводів і колій з'єднуються між собою, композиції відношень для обробки анотацій і обмеження для перевірки відповідності даних таблиць відомостей і наказу правилам ДБН [168] і ПТЕ [204].

Композиції відношень використовуються для зв'язування відомостей стрілочних переводів і колій в композиціях так, щоб отримати факт, контрольований ПТЕ [204], тобто зв'язати назву колії з хрестовиною стрілочного переводу, що лежить на ній.

Для зв'язування значень таблиць відомостей колій і стрілочних переводів послідовно виконуються такі операції:

1) зв'язування станційної колії і марки хрестовини стрілочного переводу відношенням «is switch frog type» (Рисунок 3.8) композицією відношень `isSwitchFrogTypeOf SuperProperty Of isFrogTypeOf o isSwitchOf`;

2) зв'язування назви колії і марки хрестовини стрілочного переводу відношенням «drawing corresponds to» (Рисунок 3.8) композицією відношень `drawingCorrespondsTo SuperPropertyOf isSwitchFrogTypeOf o hasName`.

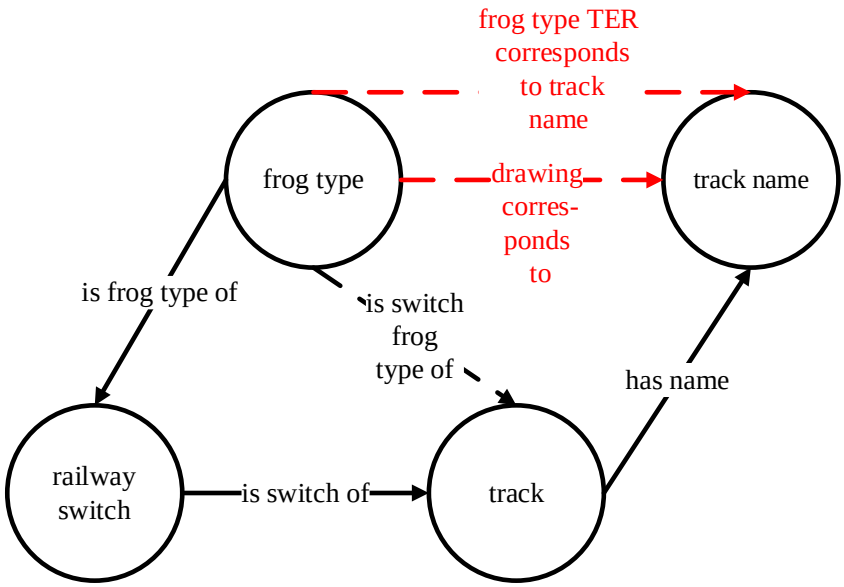


Рисунок 3.8 Композиції відношень для зв'язування даних таблиць відомостей стрілочних переводів і колій

Класифікація та перевірка назв колій і стрілочних переводів виконується різонером (reasoner) за аксіомами типу:

1) `mainTrackName Equivalent To trackName and isRepresentedByLiteral some {“головна”}`
2) `mainHameFrogType EquivalentTo trackName and isRepresentedByLiteral some {“головна”}`
3) `mainHameFrogType SubClass Of frogType and listCorrespondsTo some mainTrackName`

Розглянемо випадок несправної колії загального користування.

Згідно ІРП [203], обмеження швидкості записуються в КЗПОШ і в ФВМПОШ. Це таблиці КЗПОШ і ФВМПОШ, що описуються такими класами, як «bookSpeed», «defect», «formSpeed» і такими відношеннями, як «bookSpeedCorrespondsToActualDefect», «equals», «formSpeed».

За даними МВАВВП, перед тим як видати ФВМПОШ черговий по парку залізничних станцій порівнює обмеження швидкості КЗПОШ і ФВМПОШ. Це ті ж таблиці, але зі значенням швидкості класу «speedValue» і відношеннями для перевірки рівності «isSpeedValueOf», «hasSpeedValue».

Згідно з інструкціями ІПУЗК [228] та ІЗБВКР [227], результати огляду записуються в КОЗК та КОСП. Таблиці відповідних книг представлені класами типу «defectValue», «defect», «infrastructureElement» и відношеннями типу «isActualDefectOf», «isActualDefectValueOfTheElement», «isActualDefectOf».

Для перевірки відповідності швидкості і несправності були сформовані такі класи, як «P65SwitchBigRampWearingOut» і «mediumSettlementValue».

Згідно з ПТЕ [204], обмеження швидкості або особливі умови діють в разі несправності колії і або виконання колійних робіт. Рівень ПТЭ не визначає нові таблиці, але додає відношення, яке зв'язує колію з обмеженням «isActualSpeedLimitOf».

Конкретна модель залізничної інфраструктури з правилами розроблена у вигляді п'яти моделей TSRR, TMR для інтеграції і двох TMR для перевірки даних на відповідність інструкціям колійного господарства.

Розроблені конструкції для методів обробки існуючої і виправленої несправності залізничної колії. Перший випадок ділиться на чотири рівні, а другий на три. Тому що ПТЕ [204] не містить інструкцій щодо обробки скасування обмеження швидкості.

Розглянемо випадок діючої несправності і обмеження швидкості. Згідно ПТЕ [204], наявність обмеження швидкості на колії означає наявність несправності. Згідно ІПУЗК [228], наявність у колії несправності означає, що ця несправність має значення. Згідно ІРП [203], наявність обмеження швидкості на колії означає, що швидкість ФВМПОШ відповідає такій в КЗПОШ. Відповідно до МВАВВП [195], має бути рівність значень цих швидкостей. На Рисунку 3.9, стрілками позначено відношення, кругами – класи, а пунктирні стрілки – це композиції відношень. Рівень МВАВВП виділений зеленим кольором, помаранчевим – ІРП [203], синім – ІПУЗК [228] та ІЗБВКР [228], чорним – ПТЕ [204].

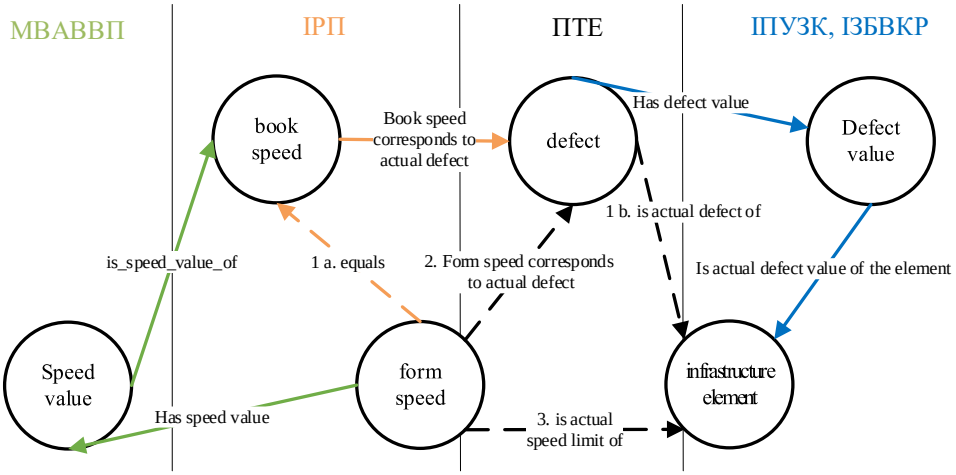


Рисунок 3.9 Багаторівнева агрегація відношень діючої несправності та обмеження швидкості залізничної інфраструктури

Розглянемо випадок ремонту і скасування попередження про обмеження швидкості (Рисунок 3.10). Відповідно до ІПУЗК [228] і ІЗБВКР [227], усунення несправності означає, що дана несправність пов'язана з деяким значенням (наприклад, просадка колії 4 мм) і в результаті ремонту була усунена вся несправність (4 мм просідання). Згідно ІРП [203], припинення дії обмежень у зв’язку з несправністю означає, що зроблена заявка на скасування обмеження швидкості, а скасування обмеження швидкості для елемента інфраструктури означає, що відміна дана, а ремонт виконаний.

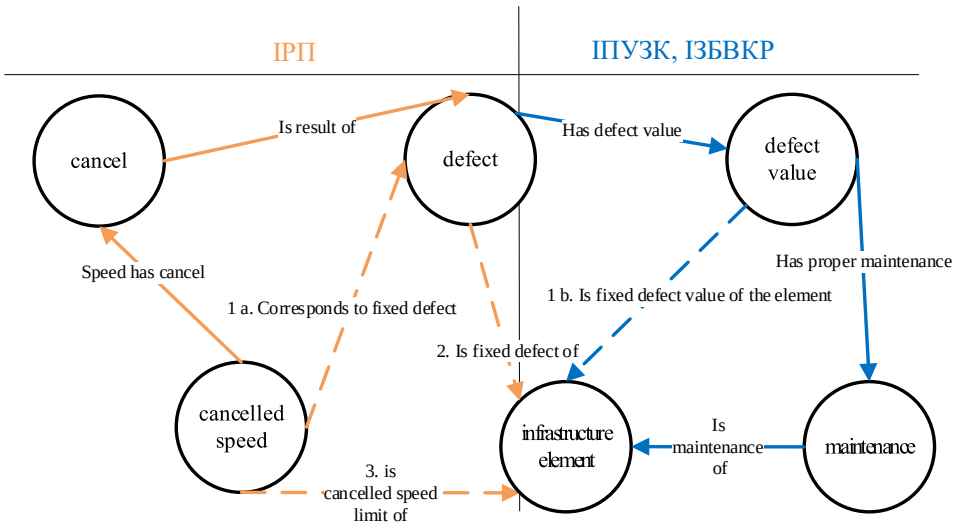


Рисунок 3.10 Багаторівнева агрегація відношень ремонту та скасування обмежень швидкості поїздів

Композиції відношень для різних рівнів агрегації нормативних документів узагальнені в Таблиця 3.1.

Таблиця 3.1 композиції відношень нормативних документів несправності залізничної колії

Інструкція	Зв'язані таблиці	Композиція відношень
МВАВВП [195]	ФВМПОШ, КЗПОШ	`Equals SuperProperty Of hasConditionsValue o isConditionsValueOf`
		`equals SuperProperty Of hasspeedvalueoisspeedvalueof`
ІРП [203]	КЗПОШ	`correspondsToFixedDefect SuperProperty Of speedHasCancel o isResultOf`
		`isCancelledSpeedLimitOf SuperProperty Of correspondsToFixedDefect o isFixedDefectOf`
	ФВМПОШ, КЗПОШ	`formConditionsCorrespondToActualWork SuperProperty Of equals o bookConditionsCorrespondToActualWork`
		`formSpeedCorrespondsToActualDefect SuperProperty Of equals o bookSpeedCorrespondsToActualDefect`
ІПУЗК [228]	КЗПОШ, КОЗК	`isActualDefectOf SuperProperty Of hasDefectValue o isActualDefectValueOfTheElement`
		`isFixedDefectOf SuperProperty Of hasDefectValue o isFixedDefectValueOfTheElement`
	КОЗК	`isFixedDefectValueOftTeElement SuperProperty Of hasProperMaintenance o isMaintenanceOf`
ІЗББКР [227]	КЗПОШ	`isActualWorkOf SuperProperty Of hasWorkValue o isActualWorkValueOfTheElement`
ПТЕ [204]	КЗПОШ, КОЗК	`isActualConditionOf SuperProperty Of FormConditionsCorrespondtoActualWork o isActualWorkOf`
		`isActualSpeedLimitOf SuperProperty Of formSpeedCorrespondsToActualDefect o isActualDefectOf`

Розроблено логічні визначення такого типу для перевірки узгодженості несправності колії і обмеження швидкості:

1) `rampWearingOut Equivalent To defect and (hasName some {"ramp wearing out"})`

2) bigRampWearingOut Equivalent To rampWearingOut and hasDefectValue some bigRampWearingOutValue
3) bigRampWearingOutValue Equivalent To rampWearingOutValue and isRepresentedByLiteral some xsd:int[> "10"^^xsd:int, < "12"^^xsd:int]
4) P65SwitchBigRampWearingOut Equivalent To bigRampWearingOut and isActualDefectOf some P65Switch or isFixedDefect of some P65Switch
5) rampWearingOutValue Equivalent To defectValue and hasName some {"ramp wearing out"}
6) 25Speed Equivalent To BookSpeedRestriction and bookSpeedRestriction hasSpeedValue some 25SpeedValue
7) 25SpeedValue Equivalent To speedValue and isRepresentedByLiteral some {"25"}

Розроблені обмежень такого типу для перевірки узгодженості несправності колії та обмеження швидкості:

1) 25Speed SubClass Of bookSpeedCorrespondsToActualDefect only P65SwitchBigRampWearingOut
2) 25Speed SubClass Of correspondsToFixedDefect only P65SwitchBigRampWearingOut
3) bookSpeedRestrictionHasSpeedValue Inverse Of isSpeedValueOf
4) P65SwitchMediumRampWearingOut Disjoint With P65SwitchBigRampWearingOut
5) bigRampWearingOutValue Disjoint With MediumRampWearingOutValue

3.2 Подальша інтеграція розробленої онтології з інформаційними системами залізниці

Розглянемо частини онтологій, які відображають деякі технологічні процеси, що відносяться до кількох підонтологій і, відповідно, кількох інформаційних підсистем АСК ВП УЗ-Є.

3.2.1 Підонтологія вагонної моделі

В онтології, що відповідає базі даних вагонів, концептуалізація виконана на основі восьмизначної нумерації вагонів, а ієрархічні відношення в таксономії – з використанням міцних відношень (strong relationships) rdfs:subclassOf. Як необхідна умова і квантор універсальності owl:allValuesFrom представлено обмеження за призначенням вагона для перевезення вантажів з ПТЕ для підприємств [232]. Але для того, щоб представники вагонного господарства пропустили вагон на станційні колії, необхідна відсутність несправності, яка загрожує безпеці руху поїздів. Як приклад, обмеження по висоті гребеня колеса з ПТЕ [204] для УЗ було формалізовано за допомогою наступних конструкцій OWL owl:withRestrictions і xsd:maxInclusive. В онтологію вагонної моделі імпортуються модулі-онтології бази даних підприємства та обмеження з нормативно-правової документації Укрзалізниці та зв'язуються між собою за допомогою відношення перетину owl:intersectionOf як необхідна і достатня умова (loose coupling):

```
Class: boxcar EquivalentTo: boxcar and hasFlange some ≤18
and hasLocation some modelTrack
```

Тоді, при введенні екземпляра вагона підприємства в вагонну модель при врахуванні умов ПТЕ для підприємств [232] і Укрзалізниці [204], вагон пере-класифікується різонером (reasoner) як вагон вагонної моделі АСК ВП УЗ-Є.

З точки зору визначення тут поняття boxcar пов'язане відношенням «hasFlange» зі значенням підрізу і «hasLocation» зі станційною колією. Онтологія включає аксіоми OWL, наприклад, твердження, що всі boxcar – це car (підкласс), і що критий вагон car111 стоїть на колії 2А, що належить станційній моделі. Існують також теореми про

те, що, наприклад, якщо у вагона є підріз колеса, то він не повинен перевищувати 14 мм.

Тобто у випадку з модулями, зв'язаними з вагонами, різонер (reasoner) шукає екземпляри, які не відповідають необхідним умовам (справність і придатність для перевезення вантажів), щоб виділити невідповідні вагони, а також екземпляри, що відповідають достатнім умовам, щоб додати їх з моделі підприємства до моделі УЗ.

Таблиця 3.2 Онтологічне забезпечення моделі вагона

Підонтологія	Призначення підонтології	Визначення (нормативне/ розроблене)	Аксіоми OWL
Вагони підприємства	Знаходження вагона на підприємстві в їх базі даних	вагони вантажні – вагони, призначені для перевезення вантажів	:car rdf:type owl:Class ; rdfs:subClassOf [rdf:type owl:Restriction ; owl:onProperty :carries ; owl:allValuesFrom :freight] ; owl:disjointWith :freight.
Правила експлуатації вагонів	огляд вагона на предмет несправності ПТЕ [204]	(несправностей)... понаднормативних дефектів... колеса... (вертикальний підріз гребеня висотою понад 18 мм)	:has_flange rdf:type owl:DatatypeProperty ; rdfs:range [rdf:type rdfs:Datatype ; owl:onDatatype xsd:integer ; owl:withRestrictions ([xsd:minInclusive 0] [xsd:maxInclusive 14])].
Вагонна модель	Внесення вагона підприємства у вагонну модель станції шляхом формалізації знань, що відповідають базі даних вагонів підприємства та правилам експлуатації вагонів	Class: boxcar EquivalentTo: boxcar and hasFlange some ≤18 and hasLocation some modelTrack	carModel:modelCar rdf:type owl:Class; owl:equivalentClass [owl:intersectionOf (enterprise_cars:car [rdf:type owl:Restriction ; owl:onProperty car_model:have_location ; owl:someValuesFrom stationModel:modelTrack] [rdf:type owl:Restriction ; owl:onProperty car_normatives:hasFlange ; owl:someValuesFrom xsd:integer]) ; rdf:type owl:Class].

3.2.2 Підонтологія поїзної моделі

Онтологія поїздів підприємства розроблена шляхом концептуалізації на основі глосарія ПТЕ для підприємств [232]. В якості необхідної умови представлено обмеження на кількість локомотивів в поїзді за допомогою конструкції owl:minQualifiedCardinality. Для того щоб поїзд був доданий до графіку руху поїздів, необхідно, щоб його вага відповідала нормативу для графіку, що в модулі онтології досягається з використанням необхідної умови owl:withRestrictions і xsd:maxInclusive. В онтологію поїзної моделі імпортуються модулі з поїздами підприємства та нормативами для графіка руху поїздів Укрзалізниці і зв’язуються між собою з використанням відношення перетину owl:intersectionOf:

```
modelTrain EquivalentTo: train and hasWeight some ≤3400 and hasLocation some modelTrack
```

При введені копії поїзда в модель підприємства, якщо він має локомотив і його вага менше нормативного, то він буде класифікований різонером (reasoner) як поїзд поїзної моделі.

З точки зору визначення, поняття поїзд, як і вагон, пов'язане відношенням «hasWeight» зі значенням ваги і відношенням «hasLocation» із колією станції. Наприклад, онтологія включає наступну аксіому, що якщо поїзд знаходиться на станційній колії і його вага не перевищує нормативну, то він належить до поїзної моделі Укрзалізниці.

3.2.3Підонтологія відправочної моделі

Онтологія вантажів підприємства розроблена шляхом концептуалізації на основі класифікації вантажів з ППНВ [230]. Для того щоб вантаж у вагоні був прийнятий до перевезення колією загального користування, необхідно, щоб прийомоздавальникки переконалися, що вантаж відповідає вагону додатка 2 ППО. В онтологію відправочної моделі імпортуються онтології модулі вантажів підприємства і обмеження з ППНВ [230] і зв'язуються між собою:

```
modelFreight EquivalentTo: enterpriseFreight and hasLocation some modelBoxcar
```

Таблиця 3.3 Онтологічне забезпечення поїзної моделі

Підонтологія	Призначення підонтології	Визначення (нормативне/ розроблене)	Аксіоми OWL
Поїзди на промислової станції	Місцезнаходження поїзда на станції підприємства	поїзд – … состав вагонів з … локомотивами	:train rdf:type owl:Class ; owl:equivalentClass [rdf:type owl:Class ; owl:oneOf (:train1)] ; rdfs:subClassOf [rdf:type owl:Restriction ; owl:onProperty :has_locomotive ; owl:someValuesFrom :locomotive], [rdf:type owl:Restriction ; owl:onProperty :has_locomotive ; owl:minQualifiedCardinality "1"^^xsd:nonNegativeInteger ; owl:onClass :locomotive].
Норми графіку руху поїздів	Перевірка інженером поїзда на відповідність нормативам графіка	Залізниці для кожної дільниці визначають масу поїзда за потужністю локомотива з урахуванням розрахункового ухилу	:hasWeight rdf:type owl:DatatypeProperty ; rdfs:range [rdf:type rdfs:Datatype ; owl:onDatatype xsd:integer ; owl:withRestrictions ([xsd:minInclusive 400] [xsd:maxInclusive 3400])].
Поїзна модель	Додавання поїзда до поїзну моделі шляхом формалізації знань відповідної бази даних поїздів компанії, а також вагових норм	modelTrain EquivalentTo: train and has_weight some ≤3400 and hasTocation some modelTrack	trainModel:model_train rdf:type owl:Class ; owl:equivalentClass [owl:intersectionOf (enterpriseTrains:train [rdf:type owl:Restriction ; owl:onProperty train_model:hasLocation ; owl:someValuesFrom stationModel:modelTrack] [rdf:type owl:Restriction ; owl:onProperty trainNormatives:hasWeight ; owl:someValuesFrom xsd:integer]) ; rdf:type owl:Class].

При внесенні екземпляра вантажу підприємства в онтологію, якщо йому присвоєно правильний клас небезпеки і вагон, вантаж автоматично буде класифікований різонером (reasoner) у відправочну модель Укрзалізниці.

З точки зору визначення, поняття «вантаж» зв'язане відношенням «мати клас» зі значенням класу і «перевозитися» з вагоном. Існує також, наприклад, теорема: якщо вантаж знаходиться у вагоні, виділеному йому ППНВ [230], то він належить до відправочної моделі Укрзалізниці.

Таблиця 3.4 Онтологічне забезпечення відправочної моделі

Підонтологія	Призначення підонтології	Визначення (нормативне/ розроблене)	Аксіоми OWL
Вантажі підприємства	Знаходження вантажів в базі даних підприємства	Небезпечний вантаж – речовини …, які … віднесено до одного з класів небезпечних речовин	:has_class rdf:type owl:DatatypeProperty ; rdfs:range [rdf:type rdfs:Datatype ; owl:oneOf [rdf:type rdf:List ; rdf:first "1"^^xsd:int ; rdf:rest [rdf:type rdf:List ; rdf:first "2"^^xsd:int ; rdf:rest [rdf:type rdf:List ; rdf:first "3"^^xsd:int ; rdf:rest rdf:nil]]]].
Правила перевезення небезпечних вантажів	Перевірка прийомо-здавальником відповідності вантажу вагону	Найменування вантажу Алкілдиметиламіну оксид Рід вагона, тип контейнера КВ – криті вагони; УК – універсальні контейнери	:some_freight rdf:type owl:Class ; rdfs:subClassOf [rdf:type owl:Class ; owl:unionOf ([rdf:type owl:Restriction ; owl:onProperty :carried_by; owl:someValuesFrom :normative_boxcar] [rdf:type owl:Restriction ; owl:onProperty :carried_by; owl:someValuesFrom :normative_container])]
Відправочна модель	Приймання вантажу шляхом формалізації обмежень на відповідність вантажу вагону, а також бази даних підприємства	modelFreight EquivalentTo: enterpriseFreight and hasLocation some modelBoxcar	freightModel:modelFreight rdf:type owl:Class ; owl:equivalentClass [owl:intersectionOf (enterpriseFreight:gas [rdf:type owl:Restriction ; owl:onProperty normativeFreight:carriedBy; owl:someValuesFrom carModel:model_car]) ; rdf:type owl:Class].

Висновки до третього розділу

Підвищення безпеки поїздів може бути досягнуто за рахунок підвищення достовірності інформації в інформаційних системах за рахунок інтеграції різнорідних джерел даних. Розроблено моделі онтології джерел різних форматів даних, онтології залізничної інфраструктури та методу їх зв’язування, які дозволяють виконувати інтеграцію таблиць профілю колії та відомості колій і здійснювати семантичну перевірку і валідацію даних, а також побудову висновків на основі нормативної документації залізниць. Як приклад розглядається узгодження інформаційного забезпечення під час введення в експлуатацію під'їзної колії, оглядів і капітальних ремонтів. Виконано перевірку онтологічного забезпечення засобами різонерів Pellet та Hermit Protégé і отримано висновок про узгодженість схеми.

Онтологія залізничної інфраструктури використовується для контролю даних паспорта під'їзної колії та попереджень про обмеження швидкості колії загального користування відповідно до нормативної документації. Онтології джерел даних відіграють допоміжну роль.

Розроблено модульну онтологію допустимих швидкостей на ділянці залізничної колії з використанням композиції відношень. Дана онтологія дає можливість виконувати інтеграцію таблиць характеристик колії та наказу про допустимі швидкості та контролювати узгодженість швидкостей і характеристик залізничної інфраструктури у відповідних системах з ПТЕ [204] і ДБН [168].

Розроблено методи інтеграції даних таблиць огляду та ремонту колій та заявок на видачу та відміну попереджень та перевірки узгодженості обмежень швидкості руху поїздів у зв’язку з несправністю колії та нормативних документів залізниць шляхом агрегування правил, що дозволяють підвищити рівень безпеки на залізничному транспорті, оскільки перевіряють таблиці та виключають можливість зняття обмежень до виконання ремонтних робіт.

Розроблено базову модель та онтології моделей АСК ВП УЗ-Є, які дають можливість отримати логічний висновок у вигляді колії, вагона, поїзда та вантажу, що відповідає стандартам Укрзалізниці.

За результатами розділу, опубліковано роботи [176, 178, 180].

РОЗДІЛ 4
ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ РОЗРОБЛЕНОГО ОНТОЛОГІЧНОГО ЗАБЕЗПЕЧЕННЯ

Розроблені онтології використовуються для перевірки узгодженості даних реконструйованих ділянок України, Болгарії та Німеччини з нормативним забезпеченням швидкісних перевезень.

У Болгарії реконструюють ділянку «Plovdiv-Burgas» для збільшення швидкості руху поїздів до 160 км/год [148], а ділянку німецьких залізниць Берлін-Дрезден – до 200 км/год. Ділянка залізничної лінії, прилеглої до міста Дрездена, в даний час забезпечує швидкість максимум 120-160 км/год і потребує подальшої реконструкції [50].

В Україні залізнична інфраструктура забезпечує рух поїздів зі швидкістю 160 км/год, а швидкісна залізнична мережа змодельована в [99].

Метою цього етапу є перевірка узгодженості даних опису залізничної інфраструктури України та країн Євросоюзу та відповідних нормативних документів на основі онтологічних засобів, що сприяє підвищенню безпеки руху поїздів.

Перевірка узгодженості даних інформаційних систем і нормативної документації здійснюється теоретичним і експериментальним шляхом. Експеримент проводиться за планом:

1) підготовка даних:

a) національних залізничних нормативних документів:

– Railways-Construction and Operating Regulations («Eisenbahn-Bau- und Betriebsordnung» (EBO)) Німеччини формату PDF [52];

– «Розпорядження про проектування та будівництво залізничних колій» (Наредба за проектиране и строителство на железопътни линии) Болгарії формату MS Word [124];

– Державні будівельні норми України формату PDF [168];

b) Директив Євросоюзу формату PDF:

– Commission Regulation (EU) No 1299/2014 [27];

– Directive 2008/57/EC [55];

c) опису інфраструктури:

– файлів railML формату XML радіусів кривих залізниць ділянки «Coswig-ESig G» німецьких залізниць [51];

– таблиці Network Statement формату PDF ухилів ділянки «Plovdiv-Burgas» болгарських залізниць [112], як в [197];

– креслення поздовжнього профілю колії у форматі PDF радіусів кривих і ухилів колії ділянки «Маріуполь Порт-Волноваха» залізниць України [46], як у [57];

d) розрахунок допустимого радіуса формату XLS;

2) розробка онтологічного забезпечення:

a) модулів схеми онтології відповідно до Рисунку 4.1;

b) перетворення даних у екземпляри онтології;

c) імпортування в онтології всіх розроблених модулів за процедурою Рисунку 4.1;

3) виконання експерименту:

a) запитом SPARQL отримуються елементи інфраструктури, які мають характеристики, що не відповідають обмеженням нормативних документів для очікуваних швидкостей;

4) аналіз отриманих результатів:

a) оцінка онтології з використанням OntOlogy Pitfall Scanner (OOPS!) [128];

b) оцінка адекватності результатів експертом в предметній області [152];

c) розрахунок показника Accuracy [37] розробленої онтології.

Онтологія розробляється модульним способом. Модулі відповідають трьом рівням абстракції (Таблиця 4.1). Нумерація онтологій Рисунку 4.1 і Таблиці 4.1 узгоджена. Абстрактний рівень представлений словниками. На першому рівні конкретизації словник доповнюється даними або правилами. На другому рівні об'єднуються правила і дані, наприклад правила Commission Regulation (EU) No 1299/2014 застосовуються до даних опису залізничної інфраструктури України, Болгарії та Німеччини.

Таблиця 4.1 Модулі онтологій джерел та залізничної інфраструктури відповідно до рівня їх абстракції

Рівень	Онтологія	
	Джерел	Залізничної інфраструктури
Абстрактний рівень	1	5
Перший рівень конкретизації	2a, 2b, 3a, 3b	6a, 6b, 7a, 7b
Другий рівень конкретики	4a, 4b	8a, 8b

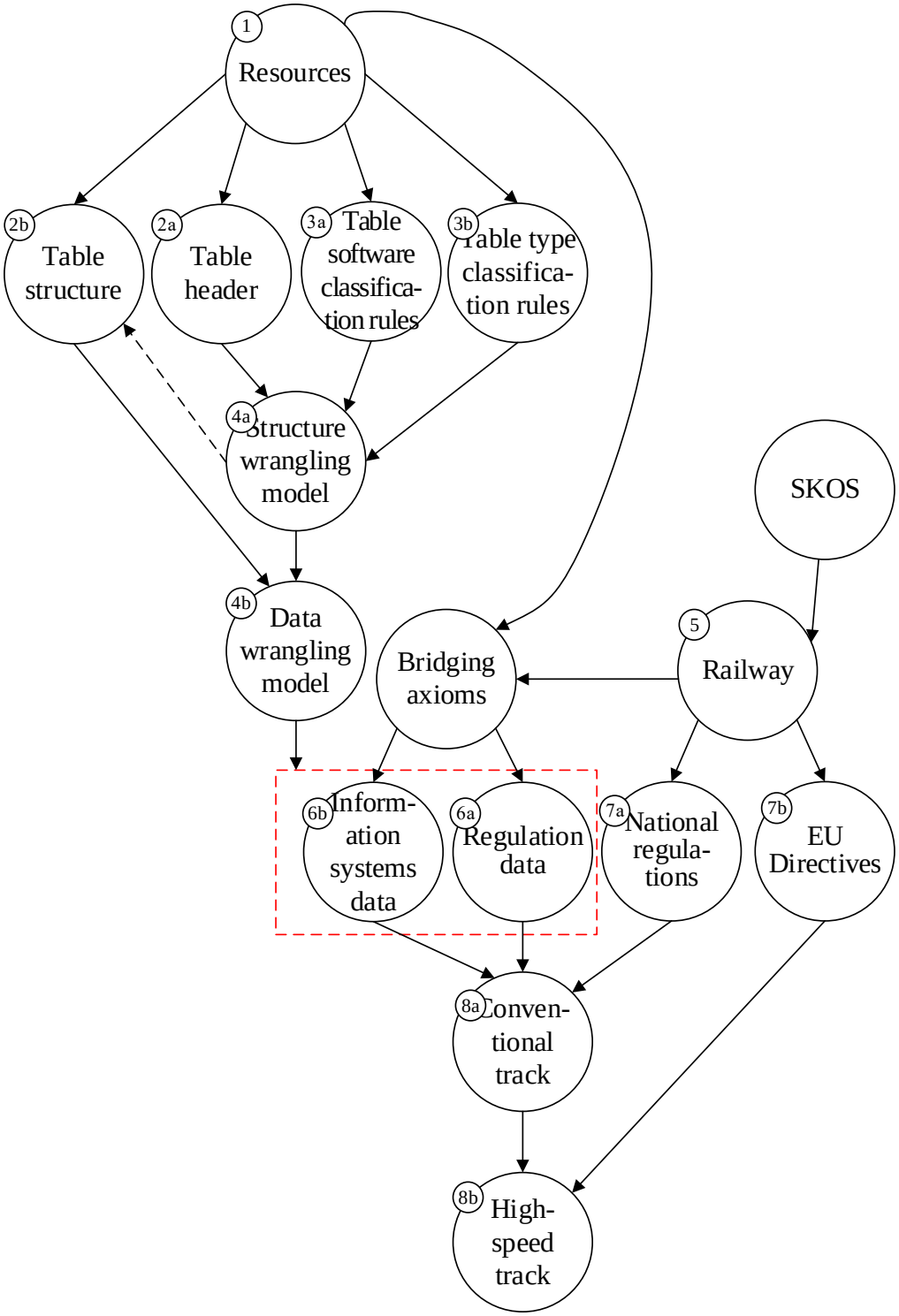


Рисунок 4.1 Модульність онтології залізничної інфраструктури

Онтології конкретної моделі джерел з даними розробляються окремо для кожного типу таблиць у вигляді двох модулів 2a, 2b. Спочатку вручну описується заголовок таблиці і скрипт її перетворення. Потім вони зв'язуються і включається різонер (reasoner) для класифікації таблиці відповідно до її колонок і програми заповнення. Якщо онтологія узгоджена, запитом SPARQL отримується скрипт, який імпортується разом зі словником і таблицею у OpenRefine, для перетворення даних таблиці на екземпляри онтології.

Узагальнено, план експерименту відповідає методології розвитку онтології extreme [37] з етапами розробки компетентносних питань, онтології та тестування з використанням запиту SPARQL. Тому розділ структурований наступним чином. Описується підготовка даних. Розроблюється кожен з модулів онтології в порядку extreme [37] (модуль, тестування). Експеримент закінчується оцінкою результатів і висновками.

4.1 Підготовка даних експерименту

Експеримент проводиться на основі реальних даних інформаційних систем та нормативних документів, отриманих з відкритих джерел (Таблиця 4.2). Видобування даних здійснюється методами присвоєння симентичних ролей в Inception для текстів, оптичного розпізнавання тексту (optical character recognition) в FineReader для креслень, і видобування даних для таблиць формату PDF в Tabula.

Таблиця 4.2 Джерела даних для експерименту

Джерело даних	Країна	Веб-сайт
Опис інфраструктури	Україна [46]	scbist.com
	Болгарія [112]	rail-infra.bg
	Німеччина [51]	railML.org
Нормативні документи	Україна [168]	dbn.co.ua
	Болгарія [124]	rail-infra.bg
	Німеччина [52]	gesetze-im-internet.de
	Євросоюз [55, 27]	eur-lex.europa.eu

Семантичне анотування виконується за допомогою тегів на основі словника онтології залізничної інфраструктури. Вхідними даними Inception є теги і тексти інструкцій, а вихідними – таблиця TSV, придатна для перетворення OpenRefine в

екземпляри онтології. Анотування дозволяє забезпечити зв'язок між текстами нормативної документації та онтології (Рисунок 4.2 – 4.3).

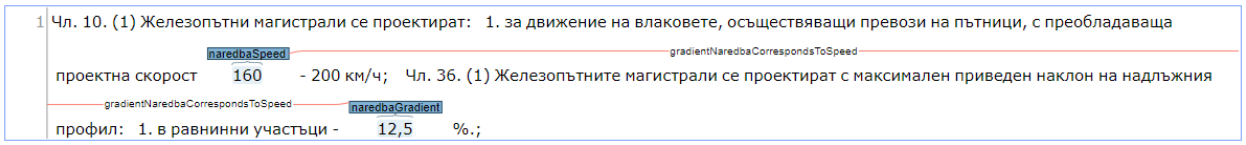


Рисунок 4.2 Семантичне анотування «Наказу про проектування та будівництво залізничних колій» Болгарії

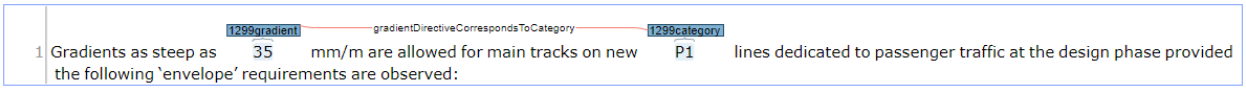


Рисунок 4.3 Семантичне анотування Commission Regulation (EU) No 1299/2014 Євросоюзу

Частина даних була підготовлена вручну. Таблиця розрахунку допустимого радіусу німецьких і швидкісних українських залізниць за формулою (1) ЕВО [52] розроблена в Excel (Таблиця 4.3).

$$R = \frac{k \cdot V^2}{h + \Delta h} \quad (4.1)$$

де h – підвищення рейки. Прийнято 180 мм [27];

Δh – недостатнє підвищення рейки. Прийнято 153 мм для колії шириною 1435 мм [27] і 163 мм для колії шириною 1520 мм [145];

k – коефіцієнт, який враховує ширину колії. Прийнято 11,8 для колії шириною 1435 мм [52] і 12,5 для колії шириною 1520 мм [64];

V – швидкість поїзда;

R – допустимий радіус кривих.

Таблиця 4.3 Розрахунок допустимих радіусів кривих в залежності від швидкості руху поїздів

Радіус кривої повороту R, м	Коефіцієнт ширини колії k	Швикість поїзда V, км/час	Підвищення рейки h, мм	Недостатнє підвищення рейки Δh, мм
908	11,8	160	180	153
1149	11,8	180	180	153
1418	11,8	200	180	153
1458	12,5	200	180	163

4.2 Реалізація конкретної з даними та конкретної другого рівня онтології джерел

Онтологія конкретної моделі джерел даних розроблена у вигляді шести онтологій формату RDF в OpenRefine.

Таблиці представлені в нормативних документах, документах, що описують інфраструктуру, анотаціях текстів. Дані про допустимий радіус для швидкісних і звичайних залізниць України представлені в таблицях ДБН [168] «Значення найменшого радіуса кривих» і «Показників категорій залізничних ліній», а про категорії ліній на залізницях Європейського Союзу – в таблиці «Performance parameters for passenger traffic» Commission Regulation (EU) No 1299/2014.

Перетворення таблиць ДБН [168], розрахунку Excel, railML, Network Statement, поздовжнього профілю колії Commission Regulation (EU) No 1299/2014 та анотації до інструкцій означає, що таблиці анотуються поняттями онтології абстрактної моделі джерел даних. В OpenRefine додатково генеруються колонки зі швидкістю, яку можна перевірити для всіх таблиць з описом колій, а також ідентифікаторами елементів профілю ділянки «Маріуполь Порт-Волноваха» Укрзалізниці.

Модуль онтології тестується на прикладі ділянки «Plovdiv-Burgas» Болгарії. Після експорту даних з OpenRefine в Protege, наприклад, виконується запит «які значення знаходяться в стовпці таблиці «Interstation Section?» (Рисунок 4.4).

Онтологія конкретної моделі джерел розроблена в Protégé, де зв'язуються між собою екземпляри скриптів і таблиць в онтології. Різонер (reasoner) використовує обмеження, щоб перевірити придатність скрипту для обробки таблиці. Якщо онтологія узгоджена, то, наприклад, виконується запит «які таблиці є вхідними даними скрипту?». Запити до фактів, отриманими шляхом логічного висновку, виконуються за допомогою вкладки DL Query Protégé або Snap SPARQL [88] Entailment Regime, що використовується в розділі.

Модуль онтології тестується на прикладі таблиці ДБН [168] «Показників категорій залізничних ліній». На Рисунку 4.5 продемонстровано запит для визначення типу таблиці.

No	Interstation section	Road	Longitudinal gradient (‰)	Beginning of the gradient
1	Brusartsi – Lom		-11.50 ‰	17,550 km
	8th Railway Line Plovdiv – Stara Zagora – Burgas			
1	Plovdiv -> Filipovo		-10.00 ‰	4,160 km
2	Filipovo -> Skutare		-3.00 ‰	12,260 km
3	Por Iztok -> Trakia	1	-2.00 ‰	5,660 km
4	Por Iztok -> Trakia	2	-2.20 ‰	5,784 km
5	Trakia -> Skutare	1	5.30 ‰	9,278 km
6	Trakia -> Skutare	2	6.40 ‰	9,295 km
7	Skutare -> Manole		2.40 ‰	20,900 km
8	Manole -> Belozem		-4.50 ‰	22,420 km
9	Belozem -> Orizovo		5.00 ‰	41,970 km
10	Orizovo -> Cherna Gora		10.00 ‰	44,340 km
11	Cherna Gora -> Chirpan		10.00 ‰	54,860 km
12	Chirpan -> Svoboda		13.92 ‰	65,102 km
13	Svoboda -> Mihaylovo		-13.00 ‰	72,220 km
14	Mihaylovo -> Kaloyanovets	1	9.5‰	85+620
15	Mihaylovo -> Kaloyanovets	2	9.5‰	85+620
16	Kaloyanovets -> Stara Zagora	1	4.80 ‰	93,672 km
17	Kaloyanovets -> Stara Zagora	2	4.80 ‰	93,672 km

SPARQL query:

```
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 table: <http://www.semanticweb.org/larisk0/ontologies/2022/2/table#>

SELECT ?columnHeader ?value
        WHERE {?columnHeader table:naming ?y.
                ?columnHeader table:isRepresentedByLiteral "in
terstationSection".
                ?y table:isRepresentedByLiteral ?value}
```

columnHeader	value
identifier2	"Plovdiv -> Filipovo"
identifier2	"Filipovo -> Skutare"
identifier2	"Por Iztok -> Trakia"
identifier2	"Por Iztok -> Trakia"
identifier2	"Trakia -> Skutare"

Рисунок 4.4 Запит до онтології таблиці «Maximum gradients along the railway lines of SE NRIC» Network Statement ділянки «Plovdiv-Burgas» Болгарії

4.3 Реалізація конкретної з даними та конкретної другого рівня онтології залізничної інфраструктури

Онтологія конкретної моделі залізничної інфраструктури з даними розроблена у вигляді 12 модулів в Protégé для кожного типу таблиці. У модуль імпортується конкретна онтологія джерел і мостова онтологія.

Таблиця 5.1 – Категорії залізничних ліній залежно від умов експлуатації

Категорія залізничних ліній	Призначення залізниць	Розрахункова річна приведена вантажонапруженість (нетто* у вантажному напрямку) на 10-й рік експлуатації, млн ткм/км	Розміри руху вантажних, пасажирських і приміських поїздів на 10-й рік експлуатації (пар приведених поїздів на добу)**	Максимальна швидкість руху пасажирських поїздів, км/год
Швидкісні	Залізничні магістральні лінії	Незалежно від вантажонапруженості	Незалежно від розмірів руху	200
I	Залізничні магістральні лінії	Більше 50	Більше 80	160
II	Залізничні магістральні лінії	Більше 30 до 50 включно	Більше 60 до 80 включно	140
III	Залізничні магістральні лінії	Більше 20 до 30 включно	Більше 40 до 60 включно	120
IV	Залізничні магістральні лінії	Більше 10 до 20 включно	Більше 25 до 40 включно	100
V	Залізничні лінії	Більше 3 до 10 включно	Більше 15 до 25 включно	80
VI	Залізничні лінії	До 3 включно	Більше 10 до 15 включно	до 80
VII	Залізничні лінії	До 3 включно	До 10 включно	до 60
	Внутрішньостанційні з'єднувальні*** та під'їзні колії	Незалежно від вантажонапруженості	Незалежно від розмірів руху	

SPARQL Query | Snap SPARQL Query

Snap SPARQL Query:

```
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 model1: <http://www.semanticweb.org/larisk0/ontologies/2022/5/model1#>
PREFIX table: <http://www.semanticweb.org/larisk0/ontologies/2022/2/table#>
SELECT ?table ?hasColumn  WHERE { ?table rdf:type model1:_SBNCategoryTable.
                                  ?x table:hasPart ?y.
                                  ?y table:hasElement ?z.
                                  ?z rdf:type model1:identifier.
                                  ?z table:isRepresentedByLiteral ?hasColumn.}
```

Execute

?table	?hasColumn
table:SBNCategoryTable	speed
table:SBNCategoryTable	passengerTrafficVolume
table:SBNCategoryTable	freightTrafficDensity
table:SBNCategoryTable	railwayFunction
table:SBNCategoryTable	lineCategory

Рисунок 4.5 Запит SPARQL з визначенням типу таблиці ДБН «Показників категорій залізничних ліній»

Модулі словника залізничної інфраструктури та мостової онтології тестуються на прикладі ділянки «Coswig-ESig G» Німеччини. Виконується запит SPARQL «які радіуси кривих на ділянки колії?» (Рисунок 4.6).

Онтологія конкретної моделі залізничної інфраструктури розроблена в Protégé. Різонер (reasoner) використовує логічні визначення для перевірки узгодженості характеристик швидкості і колії.

Модуль онтології тестується на прикладі ділянки «Маріполь Порт-Волноваха». Запитом SPARQL отримуються радіуси кривих колій, які не відповідають обмеженням для високошвидкісних магістралей.

Для ділянки «Маріуполь Порт-Волноваха» на Рисунку 4.7 а) отримано криві, що не відповідають умовам руху зі швидкістю 200 км/год. Перевірка узгодженості даних опису інфраструктури та нормативних документів проводиться на основі таблиць розрахунку радіусів для швидкісного руху. Прийнято наступні припущення. Оскільки в Україні немає рухомого складу, який розвиває швидкість 200 км/год, швидкісним рухом вважається 180 км/год, на відміну від зазначеного в ДБН [168]. Високошвидкісним вважається рух від 200 км/год для реконструйованих ліній [55] з характеристиками колії, розрахованими колії шириною 1540 мм [145], [64].

Онтологію можна використовувати для швидкості 200 км/год та звичайних залізниць (наприклад, 160 км/год). На Рисунку 4.7 б) отримано криві, що не відповідають умовам руху поїздів зі швидкістю 160 км / год. Перевірка узгодженості даних опису інфраструктури та нормативних документів здійснюється на підставі ДБН [168] «Значення найменшого радіуса кривих» та «Показників категорій залізничних ліній». Порівняння виконується для дуже складних умов.

4.4Оцінка розробленої онтології залізничної інфраструктури

Оцінка – останній етап розвитку онтології – виконується двома способами: вручну експертом предметної області та автоматично веб-застосунком.

Визначення якості онтології виконується за допомогою застосунка OOPS (OntOlogy Pitfall Scanner!) [151] (Рисунок 4.8), як, наприклад, в [13]. Після імпорту в

сканер онтології були виявлені незначні недоліки, імовірно, пов'язані з наявністю точок в ідентифікаторах значень, наприклад, «value2.1».

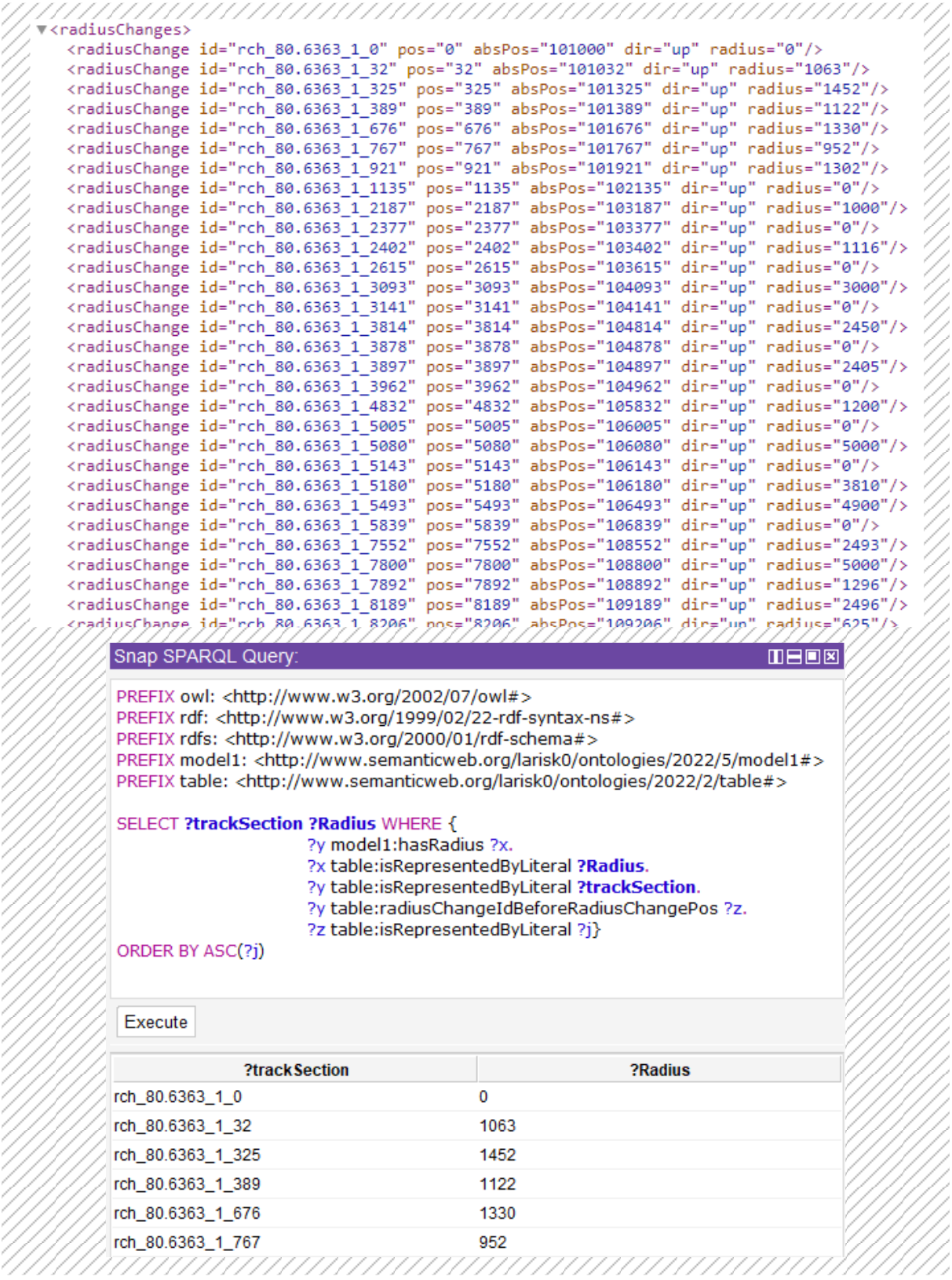


Рисунок 4.6 Запит SPARQL для визначення радіусів кривих ділянки "Coswig-ESig G" Німеччини

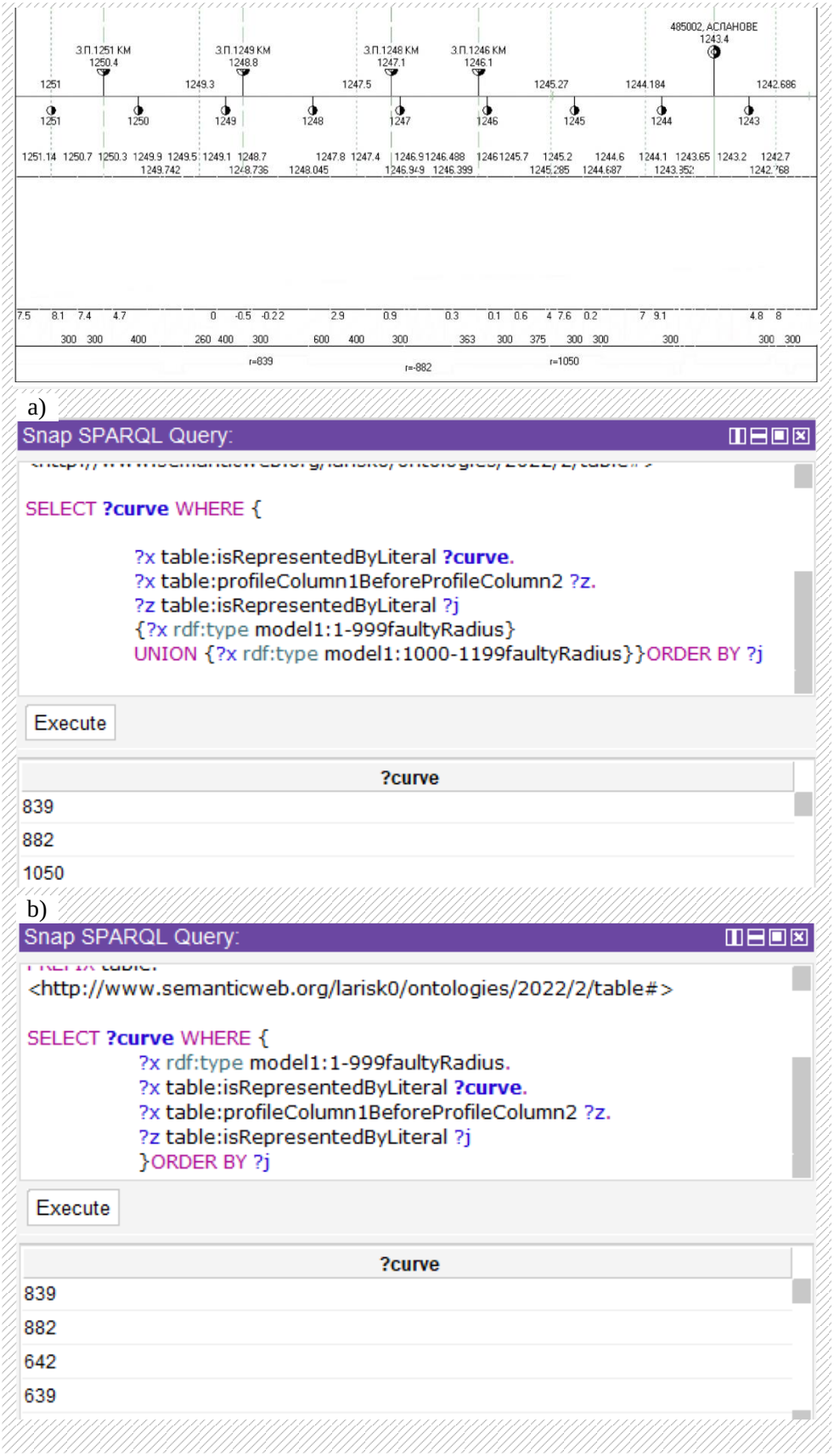


Рисунок 4.7 Запит SPARQL з визначенням кривих колії, що не відповідають умовам руху поїздів зі швидкістю 200 км/год та 160 км/год ділянки «Маріполь Порт-Волноваха»

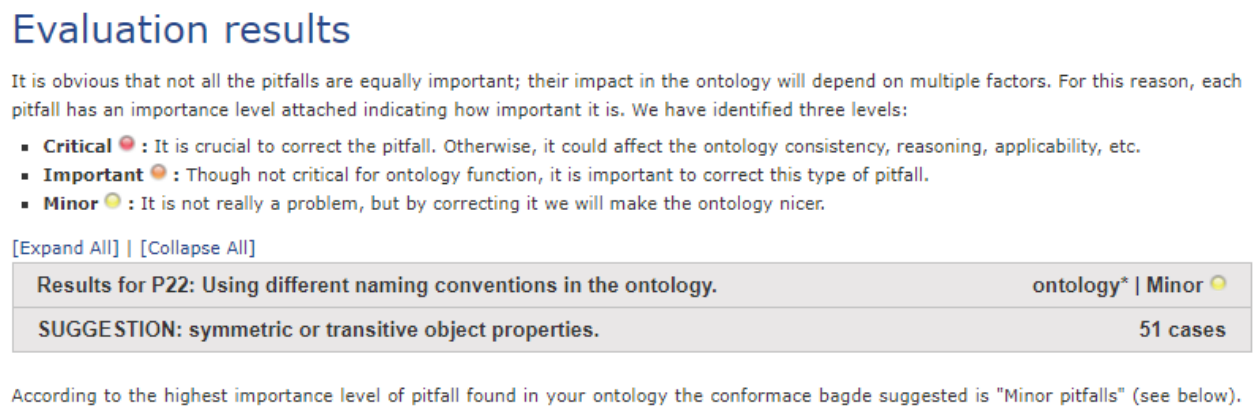


Рисунок 4.8 оцінка розробленої онтології в OntOlogy Pitfall Scanner!

Виконання онтологією вимог, покладених на неї під час розробки, визначається за допомогою тестових запитів SPARQL [152] з подальшою оцінкою адекватності результатів експертом в предметній області. У той же час, виконується «регресивне тестування» (regression testing) [152] – перевірка працездатності модулів, об'єднаних в одну онтологію. Розроблено анкету з прикладами запитань поданих для оцінювання онтології в Таблиці 4.4.

Таблиця 4.4 Анкета для оцінки онтології експертом в предметній області

Твердження	Вірно/невірно
На Рисунок 4.4 отримані значення стовпців «Interstation section» опису ділянки «Plovdiv-Burgas»	
Опис таблиці на Рисунок 4.5 відповідає ДБН [168] «Показники категорій залізничних ліній»	
На Рисунок 4.6 отримані радіуси кривих ділянки «Coswig-ESig G»	
Криві Рисунок 4.7 ділянки «Маріуполь Порт-Волноваха» радіусами 839 м і 882 м не відповідають умовам швидкості 200 км/год	
Крива Рисунок 4.7 ділянки «Маріуполь Порт-Волноваха» радіусом 1050 м відповідає умовам швидкості 160 км/год.	

Оцінка всіх аксіом виконується за допомогою показника Accuracy [37] відношення кількості аксіом до оцінки експерта «невірно» до загальної кількості аксіом.

Розроблена база знань містить близько 3000 аксіом і 500 екземплярів. З огляду на неможливість оцінки всього об’єму експерту надано наступну частину аксіом. Визначення типів таблиць інформаційних систем і нормативного забезпечення (таблиці ДБН [168], розрахунку Excel, railML, Network Statement і Commission Regulation (EU) No 1299/2014) як на Рисунок 4.5– 6 аксіом. Визначення узгодженості

швидкості та радіусів кривих ділянки «Маріуполь Порт-Волноваха» – 23 аксіоми, швидкості та ухилів ділянки «Plovdiv-Burgas» – 50 аксіом. З них всі 79 аксіом оцінені експертом з області як вірні.

Висновки до четвертого розділу

У розділі представлено експериментальну перевірку застосовності підходу запропонованого у дисертаційній роботі. Експерименти проводилися з використанням відкритих даних опублікованих для ділянок залізниці «Маріуполь Порт-Волноваха» України, у Network Statement «Plovdiv-Burgas» Болгарії і прикладів застосування відкритого стандарту railML «Coswig-ESig G» Німеччина.

Оцінка онтології виконана за допомогою сканера «OOPS!» і експерним шляхом. Всі 79 аксіом онтології оцінені експертом як вірні, а сканером знайдені лише помірні недоліки онтології.

У підсумку, згідно з оцінками експериментів, можна стверджувати, що прототип заснований на онтології коректно виконує інтеграцію та перевірку узгодженості даних. Крім того у розділі представлені джерела та нормативні документи, що слугували вихідними даними, що сприяє прозорості експерименту.

За результатами розділу, опублікована робота [174].

ЗАГАЛЬНІ ВИСНОВКИ

У дисертації досліджується актуальна науково-практична задача обробки даних та перевірки їх узгодженості на залізничному транспорті. В результаті дослідження були отримані наступні основні науково-практичні результати:

1. Виконано аналіз існуючих онтологічних розробок, що показав, що онтології на залізничному транспорті Європейського Союзу використовуються для інтеграції даних опису інфраструктури, розкладу поїздів та інших. Встановлено, що нормативному забезпеченню процесу перевезення приділяється недостатньо уваги.
2. Формалізовано онтологічні елементи табличного представлення знань та їх зв'язки у вигляді графічної нотації формальної мови. Запропоновано метод формування складних понять. Концептуалізація проведена на основі глибокої деталізації відношень між різними поняттями, що дозволило моделювати структури даних на основі таблиць онтологічними засобами.
3. Розроблено метод семантичної перевірки, що виконується на основі моделі багаторівневої конкретизації та інтеграції онтологій джерел і залізничної інфраструктури, який дозволяє поєднувати роз’єднані онтології джерел та залізничної інфраструктури держав Євросоюзу та України.
4. Запропоновано метод автоматизації видобування і перетворення даних різнорідних документів, описання станційних колій, що полегшує трудомісткий та схильний до помилок процес ручного заповнення онтологій екземплярами.
5. Запропоновано підхід формалізації обмежень нормативно-правової документації методами семантичного анотування природно-мовних текстів, для створення зв’язку між аксіомами онтології та текстом інструкцій, що надає можливість автоматизувати інтеграцію інформаційних систем залізничного транспорту з інструктивними документами.
6. Розроблено моделі онтології джерел інформаційного забезпечення різних форматів даних, що використовуються при введенні в експлуатацію під'їзної колії, оглядах та капітальних ремонтах, онтології під’їзної колії та способу

їх зв'язування. Вони дозволяють виконувати інтеграцію таблиць профілю колії та відомості колій і здійснювати семантичний контроль і валідацію даних, а також побудову висновків на основі нормативної документації залізниць.

7. Розроблено модульну онтологію допустимих швидкостей на ділянці залізничної колії (справної та несправної) з використанням композиції відношень, що дозволяє виконувати інтеграцію таблиць інформаційного забезпечення та контролювати узгодженість швидкостей і характеристик залізничної інфраструктури;
8. Розроблені онтологічні визначення залізничної колії, вагона, поїзда і вантажу нормативної документації для поєднання моделей АСК ВП УЗ-Є. Розроблено базову модель та онтології, що дозволяє інтегрувати інформаційні системи залізничних підсистем: станційну, вагонну, поїзну, відправочну.
9. Вперше формалізовано процедуру формування онтологій залізничного домену засобами конструктивно-продукційного моделювання, що дозволяє автоматизувати їх генерацію. Розроблено дев’ять конструкторів: від формування таблиці бази даних до словника, екземплярів онтології та інтегрованої бази знань.
10. Придатність запропонованого підходу підтверджена програмним шляхом за допомогою висновку різонера Protégé про узгодженість онтологій, та експериментальним – на прикладі ділянок «Маріуполь порт-Волноваха» залізниці України, «Plovdiv-Burgas» Болгарії та «Coswig-ESig G» Німеччини та відповідного нормативного забезпечення.
11. Практичне значення підтверджується впровадженням в компанії railML.org® Дрезден, Німеччина та навчальному процесі в Українському державному університеті науки і технологій.
12. Виконані дослідження з інтеграції та перевірки узгодженість даних різнопланових інформаційних систем залізничного транспорту та нормативної документації сприяють підвищенню безпеки руху поїздів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abbes, S. B., Scheuermann, A., Meilender, T., & d'Aquin, M. (2012, July). Characterizing modular ontologies. In 7th International Conference on Formal Ontologies in Information Systems-FOIs$_2$012 (pp. 13-25).
2. AGROVOC Multilingual Thesaurus (April 1, 2022). Vocabulary information. Retrieved May 4, 2022, from https://agrovoc.fao.org/browse/agrovoc/en/
3. de Aguiar, C. Z., de Almeida Falbo, R., & Souza, V. E. S. (2018, October). Ontological Representation of Relational Databases. In ONTOBRAS (pp. 140-151).
4. Almendros-Jiménez, J. M. (2012, August). Validation of XML Documents with SWRL. In International Conference on Availability, Reliability, and Security (pp. 44-57). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-32498-7_4
5. An, J., & Park, Y. B. (2018). Methodology for automatic ontology generation using database schema information. Mobile Information Systems, 2018.
6. d’Aquin, M., Haase, P., Rudolph, S., Euzenat, J., Zimmermann, A., Dzbor, M.,... & Gómez-Pérez, J. (2008). NeOn formalisms for modularization: syntax, semantics (Vol. 1). algebra. Technical report.
7. Arp, R., Smith, B., & Spear, AD (2015). Building ontologies with basic formal ontology. Mit Press.
8. Ashurst, J. L., & Collins, J. E. (2003). Gene annotation: prediction and testing. Annual Review of Genomics and Human Genetics, 4(1), 69-88.
9. Asooja, K., Bordea, G., Vulcu, G., O'Brien, L., Espinoza, A., Abi-Lahoud, E., & Butler, T. (2015). Semantic annotation of finance regulatory text using multilabel classification. LeDA-SWAn (to appear, 2015), 8.
10. Balakrishnan, R., Harris, M. A., Huntley, R., Van Auken, K., & Cherry, J. M. (2013). A guide to best practices for Gene Ontology (GO) manual annotation. Database, 2013.
11. Bao, J. (2007). Representing and reasoning with modular ontologies. Iowa State University. https://doi.org/10.31274/rtd-180813-17045

12. Bard, J., Rhee, S. Y., & Ashburner, M. (2005). An ontology for cell types. Genome biology, 6(2), 1-5.
13. Beden, S., Cao, Q., & Beckmann, A. (2021). SCRO: A Domain Ontology for Describing Steel Cold Rolling Processes towards Industry 4.0. Information, 12(8), 304. https://doi.org/10.3390/info12080304
14. Benvenuti, F., Diamantini, C., Potena, D., & Storti, E. (2017). An ontology-based framework to support performance monitoring in public transport systems. Transportation Research Part C: Emerging Technologies, 81, 188-208.
15. Bischof, S., & Schenner, G. (2021, October). Rail Topology Ontology: A Rail Infrastructure Base Ontology. In International Semantic Web Conference (pp. 597-612). Springer, Cham. https://doi.org/10.1007/978-3-030-88361-4_35
16. Boer, A., Hoekstra, R., Winkels, R., Engers, T. V., & Willaert, F. (2002, September). Proposal for a dutch legal xml standard. In International Conference on Electronic Government (pp. 142-149). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-46138-8_22
17. Caceres, P., Sierra-Alonso, A., Cuesta, C.E., & Vela, B. (2020). Improving urban mobility by defining a smart data integration platform. IEEE Access, 8, 204094-204113. https://doi.org/10.1109/access.2020.3033584
18. Calvanese, D., Cogrel, B., Komla-Ebri, S., Lanti, D., Rezk, M., & Xiao, G. (2015, May). How to stay ontop of your data: Databases, ontologies and more. In European Semantic Web Conference (pp. 20-25). Springer, Cham. https://doi.org/10.1007/978-3-319-25639-9_4
19. Calvanese, D., Gal, A., Haba, N., Lanti, D., Montali, M., Mosca, A., & Shraga, R. (2021, June). ADaMaP: Automatic Alignment of Relational Data Sources Using Mapping Patterns. In International Conference on Advanced Information Systems Engineering (pp. 193-209). Springer, Cham. https://doi.org/10.1007/978-3-030-79382-1_12
20. Ceci, M., & Gangemi, A. (2016). An OWL ontology library representing judicial interpretations. Semantic Web, 7(3), 229-253. https://doi.org/10.3233/sw-140146

21. Cellfie (January 28). Retrieved May 4, 2022, from https://github.com/protegeproject/cellfie-plugin
22. CEN European reference data model for public transport information (n.d.). Retrieved May 4, 2022, from https://www.transmodel-cen.eu/
23. Ceusters, W., & Smith, B. (2015). Aboutness: Towards foundations for the information artifact ontology.
24. Chaves-Fraga, D., Pozo-Gilo, L., Toledo, J., Ruckhaus, E., & Corcho, O. (2020, January). Morph-CSV: Virtual Knowledge Graph Access for Tabular Data. In ISWC (Demos/Industry). https://doi.org/10.3233/sw-210432
25. Ciccarese, P., & Peroni, S. (2014). The Collections Ontology: creating and handling collections in OWL 2 DL frameworks. Semantic Web, 5(6), 515-529. https://doi.org/10.3233/sw-130121
26. Cimiano, P., & Völker, J. (2005). A framework for ontology learning and data-driven change discovery. In Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB) (pp. 227-238).
27. Commission Regulation (EU) No 1299/2014 of 18 November 2014 on the technical specifications for interoperability relating to the ‘infrastructure’ subsystem of the rail system in the European Union Text with EEA relevance (June 16, 2019). Retrieved June 21, 2022, from https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv%3AOJ.L_.2014.356.01.0001.01.ENG
28. Constantin, A., Peroni, S., Pettifer, S., Shotton, D., & Vitali, F. (2016). The document components ontology (DoCO). Semantic web, 7(2), 167-181.
29. CORDIS EU research results (1 February 2006). A knowledge based platform of services for supporting medical-clinical management of heart failure withing eldery population. Retrieved May 27, 2022, from https://cordis.europa.eu/project/id/027107
30. Corsar, D., Markovic, M., Edwards, P., & Nelson, JD (2015, October). The transport disruption ontology. In International Semantic Web Conference (pp. 329-336). Springer, Cham.

31. Cota, G. (2020). Best practices for implementing fair vocabularies and ontologies on the web. Applications and Practices in Ontology Design, Extraction, and Reasoning, 49, 39. https://doi.org/10.3233/ssw200034

32. Crispino, G. (2003). Une plate-forme informatique de l'Exploration Contextuelle: cmodélisation, architecture et réalisation (ContextO): application au filtrage sémantique de textes (Doctoral dissertation, Paris 4).

33. Cruz, C., & Nicolle, C. (2008). Ontology Enrichment and Automatic Population From XML Data. ODBIS, 2008, 17-20.

34. CSVW Namespace Vocabulary Terms (06 June 2017). Retrieved May 4, 2022, from https://www.w3.org/ns/csvw

35. Cunningham, H., Tablan, V., Roberts, A., Bontcheva K. (2013) Getting More Out of Biomedical Documents with GATE's Full Lifecycle Open Source Text Analytics. PLoS Comput Biol 9(2): e1002854. doi:10.1371/journal.pcbi.1002854 — http://tinyurl.com/gate-life-sci/

36. Daconta, MC, Obrst, LJ, & Smith, KT (2003). The Semantic Web: a guide to the future of XML, Web services, and knowledge management. John Wiley & Sons.

37. McDaniel, M., Storey, V. C., & Sugumaran, V. (2018). Assessing the quality of domain ontologies: Metrics and an automated ranking system. Data & Knowledge Engineering, 115, 32-47. https://doi.org/10.1016/j.datak.2018.02.001

38. Debruyne, C., & McGlinn, K. (2021). Reusable SHACL Constraint Components for Validating Geospatial Linked Data. In Proceedings of the 4th International Workshop of Geospatial Linked Data. CEUR-WS.

39. Diamantini, C., Potena, D., & Storti, E. (2016). SemPI: A semantic framework for the collaborative construction and maintenance of a shared dictionary of performance indicators. Future Generation Computer Systems, 54, 352-365. https://doi.org/10.1016/j.future.2015.04.011

40. Diamantopoulos, T., Roth, M., Symeonidis, A., & Klein, E. (2017). Software requirements as an application domain for natural language processing. Language Resources and Evaluation, 51(2), 495-524. https://doi.org/10.1007/s10579-017-9381-z

41. Ding, LY, Zhong, BT, Wu, S., & Luo, HB (2016). Construction risk knowledge management in BIM using ontology and semantic web technology. Safety science, 87, 202-213. https://doi.org/10.1016/j.ssci.2016.04.008

42. Dinşoreanu, M., Salomie, I., & Pop, C. B. (2011). Integrated System for Developing Semantically-Enhanced Archive Econtent. Revista Română de Informatică şi Automatică, 21(4), 67.

43. Distinto, I., d'Aquin, M., & Motta, E. (2016). LOTED2: An ontology of European public procurement notices. Semantic Web, 7(3), 267-293. https://doi.org/10.3233/sw-140151

44. DOLCE: Descriptive Ontology for Linguistic and Cognitive Engineering (n.d.). Retrieved May 4, 2022, from http://www.loa.istc.cnr.it/dolce/overview.html

45. Doerr, M., Gradmann, S., Hennicke, S., Isaac, A., Meghini, C., & Van de Sompel, H. (2010, August). The europeana data model (edm). In World Library and Information Congress: 76th IFLA general conference and assembly (Vol. 10, p. 15).

46. Donetsk railway track profiles. 2020 (September 25, 2021). Retrieved June 21, 2022, from http://scbist.com/dokumenty-ukrzal-znic/54546-dokumenty-doneckoi-zheleznoi-dorogi.html

47. Duarte, B. B., de Castro Leal, A. L., de Almeida Falbo, R., Guizzardi, G., Guizzardi, R. S., & Souza, V. E. S. (2018). Ontological foundations for software requirements with a focus on requirements at runtime. Applied Ontology, 13(2), 73-105. https://doi.org/10.3233/ao-180197

48. Duarte, BB, Falbo, RA, Guizzardi, G., Guizzardi, RS, & Souza, VE (2018, October). Towards an ontology of software defects, errors and failures. In International Conference on Conceptual Modeling (pp. 349-362). Springer, Cham. https://doi.org/10.1007/978-3-030-00847-5_25

49. Dutta, B., & DeBellis, M. (2020). CODO: an ontology for collection and analysis of COVID-19 data. arXiv preprint arXiv:2009.01210. https://doi.org/10.5220/0010112500760085

50. Deutsche Bahn 2020 Integrated Report (2020). Retrieved June 21, 2022, from https://ir.deutschebahn.com/fileadmin/Englisch/2020e/Berichte/DB_IB20_e_web_01.pdf
51. East Saxony railway network by FBS (n.d.). Retrieved June 21, 2022, from https://www.railml.org/en/user/exampledata.html
52. Eisenbahn-Bau- und Betriebsordnung (EBO) (n.d.). Retrieved June 21, 2022, from https://www.gesetze-im-internet.de/ebo/EBO.pdf
53. Elizarov, AM, Lipachev, EK, Nevzorova, OA, & Solov'ev, VD (2014, August). Methods and means for semantic structuring of electronic mathematical documents. In Doklady Mathematics (Vol. 90, No. 1, pp. 521-524). Pleiades Publishing. https://doi.org/10.1134/s1064562414050275
54. Enterprise Integration Laboratory – EIL (n.d.). TOVE Ontologies. Retrieved May 4, 2022, from http://www.eil.utoronto.ca/theory/enterprise-modelling/tove/
55. EUR-Lex (17 June 2008). DIRECTIVE 2008/57/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL on theinteroperability of the rail system within the Community. Retrieved May 27, 2022, from https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=celex:32008L0057
56. European Rail Infrastructure Managers (n.d.). The Register of Infrastructure. Retrieved May 4, 2022, from https://eimrail.org/document/rinf/
57. Fadeev V.S., Shtanov O.V., Paladin N.M., Pavlushko G.D., Konakov A.V., Emelyanov E.N. Russia. Patent. 150269, MPK E 01 B 11/54, LLC NTC Information Technologies, No. 2014123170/11
58. Falco, R., Gangemi, A., Peroni, S., Shotton, D., & Vitali, F. (2014, May). Modelling OWL ontologies with Graffoo. In *European Semantic Web Conference* (pp. 320-325). Springer, Cham. https://doi.org/10.1007/978-3-319-11955-7_42
59. Fernández-López, M., Gomez-Pérez, A., & Juristo, N. (1997). Methontology: from ontological art towards ontological engineering.
60. Fiorentini, X., Gambino, I., Liang, V. C., Rachuri, S., Mani, M., Nistir, C. B.,... & Turner, J. M. (2007). An ontology for assembly representation.
61. Fonseca, J. M. D. S. (2014). Converting ontologies into DSLs (Doctoral dissertation).

62. Furini, F., Rai, R., Smith, B., Colombo, G., & Krovi, V. (2016, August). Development of a manufacturing ontology for functionally graded materials. In International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (Vol. 50084, p. V01Bt$_0$2A030). American Society of Mechanical Engineers. https://doi.org/10.1115/detc2016-59964
63. Fürst, F., & Trichet, F. (2006, October). Heavy ontology engineering. In OTM Confederate International Conferences" On the Move to Meaningful Internet Systems" (pp. 38-39). Springer, Berlin, Heidelberg
64. Gailienė, I., Gedaminskas, M., & Laurinavičius, A. (2018). Approach to rational calculation of superelevation in dual gauge track. Transport, 33(3), 699-706. https://doi.org/10.3846/transport.2018.1577
65. Gangemi, A., Peroni, S., Shotton, D., & Vitali, F. (2017). The publishing workflow ontology (PWO). Semantic Web, 8(5), 703-718. https://doi.org/10.3233/sw-160230
66. Ganino, G., Lembo, D., Mecella, M., & Scafoglieri, F. (2018). Ontology population for open – source intelligence: A GATE – based solution. Software: Practice and Experience, 48(12), 2302-2330. https://doi.org/10.1002/spe.2640
67. Gayo, JEL, Prud'hommeaux, E., Solbrig, H.R., & Boneva, I. (2017). Validating and describing linked data portals using shapes. CoRR, abs/1701.08924.
68. GeoSPARQL – A Geographic Query Language for RDF Data (nd). Retrieved May 4, 2022, from https://www.ogc.org/standards/geosparql
69. Gene Ontology Consortium Wiki (n.d.). Cross Product Guide. Retrieved May 27, 2022, from http://wiki.geneontology.org/index.php/Cross_Product_Guide
70. Giunchiglia F., Zaihrayeu I. (2009) Lightweight Ontologies. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_1314
71. Globa, L., Savchuk, Z., Vasylenko, O., & Siemens, E. (2021). QOS of data networks analyzing based on the fuzzy knowledge base. In International Conference Infocommunications–Present and Future (pp. 130-149). Springer, Cham.

72. Gómez Carrasco, C. (2016). High performance computing techniques applied to the design of complex railway infrastructures.

73. Gómez-Pérez A (1994) From Knowledge Based Systems to Knowledge Sharing Technology: Evaluation and Assessment. Knowledge Systems Laboratory, Stanford University, California. http://www-ksl.stanford.edu/KSL_Abstracts/KSL-94–73.html

74. Gruber, TR (1993). A translation approach to portable ontology specifications. Knowledge acquisition, 5(2), 199-220. https://doi.org/10.1006/knac.1993.1008

75. Guizzardi, G. (2005). Ontological foundations for structural conceptual models.

76. Guarino, N. (1997, July). Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In International Summer School on Information Extraction (pp. 139-170). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-63438-x_8

77. Guarino, N., & Giaretta, P. (1995). Ontologies and knowledge bases. Towards very large knowledge bases, 1-2.

78. Guizzardi, G., de Almeida Falbo, R., & Guizzardi, RS (2008, February). Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE Software Process Ontology. In CIbSE (pp. 127-140).

79. Hagedorn, T. (2018). Supporting Engineering Design of Additively Manufactured Medical Devices with Knowledge Management Through Ontologies. DOI: https://doi.org/10.7275/11326512.0

80. Hall, J. A. (2015). Accounting information systems. Cengage Learning.

81. Hernández, E. G., & Piulachs, J. M. (2005, September). Application of the Dublin Core format for automatic metadata generation and extraction. In International Conference on Dublin Core and Metadata Applications (pp. 213-216).

82. Hitzler, P., & Krisnadhi, A. (2018). A tutorial on modular ontology modeling with ontology design patterns: The cooking recipes ontology. arXiv preprint arXiv:1808.08433.

83. Hoekstra, R. (2010, June). Representing Social Reality in OWL 2. In OWLED (Vol. 614).

84. Hoekstra R. (n.d.). N-Ary Relation Pattern (OWL 2). Retrieved May 27, 2022, from http://ontologydesignpatterns.org/wiki/Submissions:N-Ary_Relation_Pattern_(OWL_2)

85. Hoekstra, R., Breuker, J., Di Bello, M., & Boer, A. (2007). The LKIF Core Ontology of Basic Legal Concepts. LOAIT, 321, 43-63.

86. Holzinger, W., Krüpl, B., & Herzog, M. (2006, November). Using ontologies for extracting product features from web pages. In International semantic web conference (pp. 286-299). Springer, Berlin, Heidelberg. https://doi.org/10.1007/11926078_21

87. Horridge, M., Jupp, S., Moulton, G., Rector, A., Stevens, R., & Wroe, C. (2009). A practical guide to building owl ontologies using protégé 4 and co-ode tools edition1. 2. The university of Manchester, 107.

88. Horridge, M., & Musen, M. (2015, October). Snap-SPARQL: a java framework for working with SPARQL and OWL. In International Experiences and Directions Workshop on OWL (pp. 154-165). Springer, Cham. https://doi.org/10.1007/978-3-319-33245-1_16

89. Humphreys, C. M., McLean, S., Schatschneider, S., Millat, T., Henstra, A. M., Annan, F. J.,... & Minton, N. P. (2015). Whole genome sequence and manual annotation of Clostridium autoethanogenum, an industrially relevant bacterium. BMC genomics, 16(1), 1-10.

90. IFOPT, "Identification of Fixed Objects in Public Transport" Standard CEN/TC 278, EN 28701, European Committee for Standardization, 2012.

91. InteGRail (n.d.). Railway Domain Ontology. Retrieved May 4, 2022, from http://www.integrail.eu/documents/fs02.pdf

92. Jakus, G., Milutinovic, V., Omerovic, S., & Tomazic, S. (2013). Concepts, ontologies, and knowledge representations. https://doi.org/10.1007/978-1-4614-7822-5_4

93. Jena Ontology API (n.d.). Retrieved May 4, 2022, from https://jena.apache.org/documentation/ontology/

94. Jovic, A., Gamberger, D., & Krstacic, G. (2011). Heart failure ontology. Bio Algorithms Med Syst., 7(2), 101-110.

95. Julien, N. (2012). What We Know About Wikipedia: A Review of the Literature Analyzing the Project(s). https://doi.org/10.2139/ssrn.2053597

96. Karthikeyan, S., de Herrera, A. G. S., Doctor, F., & Mirza, A. (2021). An OCR Post-Correction Approach Using Deep Learning for Processing Medical Reports. IEEE Transactions on Circuits and Systems for Video Technology, 32(5), 2574-2581.

97. Katsumi, M., & Fox, M. (2018). Ontologies for transportation research: A survey. Transportation Research Part C: Emerging Technologies, 89, 53-82. https://doi.org/10.1016/j.trc.2018.01.023

98. Katsumi, M., & Fox, M. iCity Transportation Planning Suite of Ontologies.

99. Kirpa, G. N., Bosov, A. A., & Korzhenevich I. P. (2004) On the high-speed network on the railways of Ukraine. Science and progress of transport. 4. 103-109.

100. Klein, M. (2001). XML, RDF, and relatives. IEEE Intelligent Systems, 16(2), 26-28. https://doi.org/10.1109/5254.920596

101. Klie, J. C., Bugert, M., Boullosa, B., de Castilho, R. E., & Gurevych, I. (2018, August). The inception platform: Machine-assisted and knowledge-oriented interactive annotation. In Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations (pp. 5-9).

102. Knoblock, CA, & Szekely, P. (2015). Exploiting semantics for big data integration. AI Magazine, 36(1), 25-38. https://doi.org/10.1609/aimag.v36i1.2565

103. Krima, S., Barbau, R., Fiorentini, X., Sudarsan, R., & Sriram, R.D. (2009). Ontostep: OWL-DL ontology for step. National Institute of Standards and Technology, NISTIR, 7561.

104. Lališ, A., Bolčeková, S., & Štumbauer, O. (2020). Ontology-based reliability analysis of aircraft engine lubrication system. Transportation Research Procedia, 51, 37-45. https://doi.org/10.1016/j.trpro.2020.11.006

105. Lewis, R. (2015). A semantic approach to railway data integration and decision support (Doctoral dissertation, University of Birmingham).

106. Loebe, F. (2006, November). Requirements for Logical Modules. In WoMO.

107. Loza Mencía, E., & Fürnkranz, J. (2010). Efficient multilabel classification algorithms for large-scale problems in the legal domain. In Semantic Processing of Legal Texts (pp. 192-215). Springer, Berlin, Heidelberg.
108. Lytvyn, V., Vysotska, V., Wojcik, W., & Dosyn, D. (2017). A method of construction of automated basic ontology. Computational linguistics andintelligent systems (COLINS 2017).
109. mate-tools (n.d.) Retrieved June 21, 2022, from https://code.google.com/archive/p/mate-tools/
110. Ma, C., & Molnár, B. (2022). Ontology learning from relational database: Opportunities for semantic information integration. Vietnam Journal of Computer Science, 9(01), 31-57.
111. Malone, J., Brown, A., Lister, AL, Ison, J., Hull, D., Parkinson, H., & Stevens, R. (2014). The Software Ontology (SWO): a resource for reproducibility in biomedical data analysis, curation and digital preservation. Journal of biomedical semantics, 5(1), 1-13. https://doi.org/10.1186/2041-1480-5-25
112. Maximum gradients along the railway lines of SE NRIC. URL: https://www.rail-infra.bg/upload/1779/Annex%2025_042019.pdf (дата звернення: 19.06.2022).
113. McInerney, S. J., Khakipoor, B., Garner, A. M., Houette, T., Unsworth, C. K., Rupp, A.,... & Niewiarowski, P. H. (2018). E2BMO: Facilitating User Interaction with a BioMimetic Ontology via Semantic Translation and Interface Design. Designs, 2(4), 53. https://doi.org/10.3390/designs2040053
114. McGuinness, D. L. (2001). Description Logics Emerge from Ivory Towers. Description Logics, 49.
115. Meehan, TF, Masci, AM, Abdulla, A., Cowell, LG, Blake, JA, Mungall, CJ, & Diehl, AD (2011). Logical development of the cell ontology. BMC bioinformatics, 12(1), 1-12.
116. De Meester, B., Seymoens, T., Dimou, A., & Verborgh, R. (2020). Implementation-independent function reuse. Future Generation Computer Systems, 110, 946-959. https://doi.org/10.1016/j.future.2019.10.006

117. Milosevic, N., Gregson, C., Hernandez, R., & Nenadic, G. (2016, June). Disentangling the structure of tables in scientific literature. In International Conference on Applications of Natural Language to Information Systems (pp. 162-174). Springer, Cham. https://doi.org/10.1007/978-3-319-41754-7_14

118. Morris, C., & Easton, J. (2018). Use of ontology for data integration in a degraded mode signalling system. WIT Transactions on Ecology and the Environment, 181, 215-223. https://doi.org/10.2495/cr180191

119. Morris, C., Easton, J., & Roberts, C. (2015). Ontology in the rail domain: the railway core ontologies. In Proceedings of the 7th International Conference on Knowledge Engineering and Ontology Development (KEOD 2015), Lisbon, Portugal (pp. 1175-1181). https://doi.org/10.5220/0005613702850290

120. Mouromtsev, D. I., Shilin, I. A., Pliukhin, D. A., Baimuratov, I. R., & Rezeda, R. K. (2021). Building knowledge graphs of regulatory documentation based on semantic modeling and automatic term extraction. Journal Scientific and Technical Of Information Technologies, Mechanics and Optics, 132(2), 256. https://doi.org/10.17586/2226-1494-2021-21-2-256-266

121. Mungall, CJ, Torniai, C., Gkoutos, GV, Lewis, SE, & Haendel, MA (2012). Uberon, an integrative multi-species anatomy ontology. Genome biology, 13(1), 1-20. https://doi.org/10.1186/gb-2012-13-1-r5

122. Munnelly, G. (2020). Entity Linking for Text Based Digital Cultural Heritage Collections (Doctoral dissertation, TRINITY COLLEGE DUBLIN).

123. Nanda, R., Siragusa, G., Di Caro, L., Theobald, M., Boella, G., Robaldo, L., & Costamagna, F. (2017, December). Concept Recognition in European and National Law. In JURIX (pp. 193-198).

124. Naredba № 55 ot$_2$9 yanuari 2004 g. Za proektirane i Stroitelstvo na zhelezopatni linii, Zhelezopatni gari, zhelezopatni prelezi i drugi Elementi ot zhelezopatnata infrastruktura (January 29, 2004). Retrieved June 21, 2022, from https://www.rail-infra.bg/upload/192/Naredba_55.pdf

125. Navigli, R., & Velardi, P. (2008). From Glossaries to Ontologies: Extracting Semantic Structure from Textual Definitions. Ontology Learning and Population, 167, 71-87.
126. NCI Thesaurus (NCIt) (n.d.). Retrieved May 4, 2022, from https://ncithesaurus.nci.nih.gov/ncitbrowser/
127. Neches, R., Fikes, R. E., Finin, T., Gruber, T., Patil, R., Senator, T., & Swartout, W. R. (1991). Enabling technology for knowledge sharing. AI magazine, 12(3), 36-36.
128. Neumann, M., King, D., Beltagy, I., & Ammar, W. (2019). ScispaCy: fast and robust models for biomedical natural language processing. arXiv preprint arXiv:1902.07669.
129. Noy, N. F., & McGuinness, D. L. (2001). Ontology development$_1$01: A guide to creating your first ontology.
130. Oberle, D., Grimm, S., & Staab, S. (2009). An ontology for software. In Handbook on ontologies (pp. 383-402). Springer, Berlin, Heidelberg.
131. Ogren, P. (2006, June). Knowtator: a protégé plug-in for annotated corpus construction. In Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Demonstrations (pp. 273-275).
132. Oliveira, D., & Pesquita, C. (2018). Improving the interoperability of biomedical ontologies with compound alignments. Journal of biomedical semantics, 9(1), 1-13. https://doi.org/10.1186/s13326-017-0171-8
133. de Oliveira Bringuente, A. C., de Almeida Falbo, R., & Guizzardi, G. (2011). Using a foundational ontology for reengineering a software process ontology. Journal of Information and Data Management, 2(3), 511-511.
134. Ontologies and semantic annotation (n.d.) Retrieved June 21, 2022, from https://gate.ac.uk/sale/talks/gate-course-may10/track-3/module-10-ontologies/ontologies.pdf
135. OntoMathPro: A Hub for Math Linking Open Data (LOD) (4 Aug 2017). Retrieved May 4, 2022, from https://github.com/CLLKazan/OntoMathPro

136. Osumi-Sutherland, D. (2017). Cell ontology in an age of data-driven cell classification. BMC bioinformatics, 18(17), 35-42. https://doi.org/10.1186/s12859-017-1980-6

137. Ovcharuk Iryna, & Boklah Yevhen. (2020). Information Systems in Rail Transport: Development and Prospects. Digital platform: information technologies in the sociocultural sphere, 3(2), 170–182. https://doi.org/10.31866/2617-796x.3.2.2020.220594

138. OWLAPI (April 16, 2022). OWL API main repository. Retrieved May 4, 2022, from https://github.com/owlcs/owlapi

139. owlready2 (n.d.). Retrieved May 4, 2022, from https://owlready2.readthedocs.io/en/latest/mixing_python_owl.html

140. O'connor, MJ, Halaschek-Wiener, C., & Musen, MA (2010, November). Mapping master: a flexible approach for mapping spreadsheets to OWL. In International Semantic Web Conference (pp. 194-208). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-17749-1_13

141. Palmirani, M., & Vitali, F. (2012) Legislative XML: Principles and Technical Tools.

142. Panagiotopoulos, P., Gionis, G., Psarras, J., & Askounis, D. (2011). Supporting public decision making in policy deliberations: an ontological approach. Operational Research, 11(3), 281-298. https://doi.org/10.1007/s12351-010-0081-3

143. Panov, P., Džeroski, S., & Soldatova, L. N. (2010). Representing entities in the OntoDM data mining ontology. In Inductive Databases and Constraint-Based Data Mining (pp. 27-58). Springer, New York, NY. https://doi.org/10.1007/978-1-4419-7738-0_2

144. Panov, P., Soldatova, LN, & Džeroski, S. (2016). Generic ontology of datatypes. Information Sciences, 329, 900-920. https://doi.org/10.1016/j.ins.2015.08.006

145. Patlasov, O. M., & Paylasov, Y. O. (2016). Unification of the cant and maximum values for cant deficiency. Technologijos ir menas. 6. 111–115

146. Pauwels, P., Van Deursen, D., Verstraeten, R., De Roo, J., De Meyer, R., Van de Walle, R., & Van Campenhout, J. (2011). A semantic rule checking environment for building performance checking. Automation in construction, 20(5), 506-518. https://doi.org/10.1016/j.autcon.2010.11.017

147. Peroni, S. (November 29, 2010). The Error Ontology Making constraints on ontology resources. Retrieved May 4, 2022, from https://sparontologies.github.io/error/current/error.html

148. Plovdiv-Burgas Railway Line Phase Two Rehabilitation Project, Bulgaria (May 25, 2021). Retrieved June 21, 2022, from https://www.railway-technology.com/projects/plovdiv-burgas-phase-two-railway-line-rehabilitation-project

149. Plu, J., & Scharffe, F. (2012, May). Publishing and linking transport data on the web: Extended version. In Proceedings of the First International Workshop on Open Data (pp. 62-69). https://doi.org/10.1145/2422604.2422614

150. Popova, M., Globa, L., & Novogrudska, R. (2021). Multilevel Ontologies for Big Data Analysis and Processing. In Proceedings of the 9th International Conference on Applied Innovations in IT (pp. 41-53).

151. Poveda-Villalón, M., Gómez-Pérez, A., & Suárez-Figueroa, M. C. (2014). Oops!(ontology pitfall scanner!): An on-line tool for ontology evaluation.

152. Presutti, V., Daga, E., Gangemi, A., & Blomqvist, E. (2009, October). eXtreme design with content ontology design patterns. In Proc. Workshop on Ontology Patterns (pp. 83-97).

153. Quirino, G. K., Barcellos, M. P., & Falbo, R. A. (2017, November). OPL-ML: A modeling language for representing ontology pattern languages. In International Conference on Conceptual Modeling (pp. 187-201). Springer, Cham.

154. Rosén, G. (2019). Analysis of Tabula: A PDF-Table extraction tool.

155. R2RML: RDB to RDF Mapping Language (27 September 2012). Retrieved May 4, 2022, from https://www.w3.org/TR/r2rml/

156. railML (n.d.). Retrieved May 4, 2022, from https://www.railml.org/

157. railML2.4nor – Norway's Railway Data Exchange Format (8 January 2021). Retrieved May 4, 2022, from https://www.jernbanedirektoratet.no/railml
158. RailNet Europe (n.d.). network statements. Retrieved May 4, 2022, from https://rne.eu/organization/network-statements/
159. RailTopoModel(nd). Retrieved May 4, 2022, from https://www.railtopomodel.org
160. Rector A. (2005). When and Why to use a Classifier. Retrieved May 27, 2022, from https://protege.stanford.edu/conference/2005/slides/5.2_Rector_Why_classify_Protege_workshop_2005-v2.pdf
161. Rector, A. L. (2003, October). Modularisation of domain ontologies implemented in description logics and related formalisms including OWL. In Proceedings of the 2nd international conference on Knowledge capture (pp. 121-128). https://doi.org/10.1145/945645.945664
162. Rector, A., Aranguren, M.E. (2003). normalization. Retrieved May 4, 2022, from http://ontologydesignpatterns.org/wiki/Submissions:Normalization
163. Rogushina, J. V. (2019). Means and methods of the unstructured data analysis. Problems in programming, (1), 57-77.
164. Rogushina, J., & Gladun, A. (2021). Task Thesaurus as a Tool for Modeling of User Information Needs. In New Perspectives on Enterprise Decision-Making Applying Artificial Intelligence Techniques (pp. 385-403). Springer, Cham.
165. Roman, D., Alexiev, V., Paniagua, J., Elvesæter, B., von Zernichow, BM, Soylu, A., & Taggart, C. (2022). The euBusinessGraph ontology: A lightweight ontology for harmonizing basic company information. Semantic Web, 13(1), 41-68. https://doi.org/10.3233/sw-210424
166. Ruy, F. B., Guizzardi, G., Falbo, R. A., Reginato, C. C., & Santos, V. A. (2017). From reference ontologies to ontology patterns and back. Data & Knowledge Engineering, 109, 41-69.
167. Schreiber, G., Wielinga, B., & Jansweijer, W. (1995, August). The KACTUS view on the 'O'word. In IJCAI workshop on basic ontological issues in knowledge sharing (Vol. 20, p. 21).

168. SBN B.2.3-19-2008. Railway transport facilities 1520 mm track design standards (29895) (2008) Retrieved June 21, 2022, from https://dnaop.com/html/29895/doc-%D0%94%D0%91%D0%9D_%D0%92.2.3-19-2008

169. Shah, NH, Jonquet, C., Chiang, AP, Butte, AJ, Chen, R., & Musen, MA (2009, February). Ontology-driven indexing of public datasets for translational bioinformatics. In BMC bioinformatics (Vol. 10, No. 2, pp. 1-10). BioMedCentral. https://doi.org/10.1186/1471-2105-10-s2-s1

170. Shape Expressions Language 2.1 (8 October 2019). Retrieved May 4, 2022, from https://shex.io/shex-semantics/index.html

171. Shapes Constraint Language (SHACL) (20 July 2017). Retrieved May 4, 2022, from https://www.w3.org/TR/shacl/

172. Shynkarenko, V., & Kuropiatnyk, O. (2018). Constructive model of the natural language. Acta Cybernetica, 23(4), 995-1015. https://doi.org/10.14232/actacyb.23.4.2018.2

173. Shynkarenko, V., & Zhuchyi, L. (2022) Constructive-synthesizing modeling of ontological document management support for the railway train speed restrictions. Science and Transport Progress, (2 (98)), 59-68. DOI: https://doi.org/10.15802/stp2022/268001

174. Shynkarenko, V., Zhuchyi, L., (2022). Ontological suitability analysis of railway tracks for high-speed traffic. Computer systems and information technologies, (3), 11-21. DOI: https://doi.org/10.31891/csit-2022-3-2

175. Shynkarenko, V. I., & Zhuchyi, L. I. (2022). Ontological support of metallurgical and machine-building technological processes. In Informacijni tehnologiї v metalurgiї ta mashinobuduvanni (p. 178-180). Дніпро

176. Shynkarenko, V., & Zhuchyi, L. (2022). Semantic Checking of Different Type Information Sources About Permitted Speeds in Railway Transport. CEUR Workshop Proceedings. 2022. Vol.: International Conference on Computational Linguistics and Intelligent Systems. Vol.: Main Conference, COLINS2022, 2022.

177. Shynkarenko, V., & Zhuchyi, L. (2021). Semantic checking of train speed limit warnings. In Modern information and communication technologies on a transport, in industry and education (p. 142). Дніпро
178. Shynkarenko, V., & Zhuchyi, L. (2021). Ontological Harmonization of Railway Transport Information Systems. COLINS. (pp. 541–554)
179. Shynkarenko, V., Zhuchyi, L., & Ivanov, O. (2021, September). Conceptualization of the tabular representation of knowledge. In 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT) (Vol. 2, pp. 248-251). IEEE. DOI: 10.1109/CSIT52700.2021.9648761.
180. Shynkarenko, V., Zhuchyi, L., & Ivanov, O. (2022) Ontology-Based Semantic Checking of Data in Railway Infrastructure Information Systems. Foundations of Computing and Decision Sciences, 47(3), 291-319.
181. Skalozub, V., Ilman, V., & Shynkarenko, V. (2017). Development of ontological support of constructive-synthesizing modeling of information systems. Eastern-European Journal of Enterprise Technologies, 6(4(90), 58–69. https://doi.org/10.15587/1729-4061.2017.119497
182. Skalozub, V., Ilman, V., & Shynkarenko, V. (2018). Ontological support formation for constructive-synthesizing modeling of information systems development processes. Eastern-European Journal of Enterprise Technologies, 5(4(95), 55–63. https://doi.org/10.15587/1729-4061.2018.143968
183. Skevakis, G. (2011). EUROMUSE: A web-based system for the management of MUSEum objects and their interoperability with EUROpeana (Doctoral dissertation, Technical University of Crete).
184. SKOS Simple Knowledge Organization System (August$_1$8, 2009). reference. Retrieved May 4, 2022, from https://www.w3.org/TR/skos-reference/
185. State Regulatory Acts on Labor Protection Legislation (February 20, 1987). NPAOP 63.21-1.14-87 (45248). Retrieved May 4, 2022, from https://dnaop.com/html/45248/doc-npaop-6321-114-87-pravila-tehniki-bezopasnosti-pri-ekspluatacii-kontaktnoj-seti-elektrificirovannyh-zheleznyh-dorog-i-ustrojstv-

186. Strategy JSC Ukrzaliznytsia for 2019–2023 (n.d.). Retrieved May 4, 2022, from https://www.uz.gov.ua/files/file/%D0%A1%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D1%96%D1%8F-4-Typography.pdf
187. Suárez-Figueroa, MC, Gomez-Pérez, A., & Fernández-López, M. (2012). The NeOn methodology for ontology engineering. In Ontology engineering in a networked world (pp. 9-34). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-24794-1_2
188. Szekely, P., Knoblock, CA, Gupta, S., Taheriyan, M., & Wu, B. (2011, November). Exploiting semantics of web services for geospatial data fusion. In Proceedings of the 1st ACM SIGSPATIAL International Workshop on Spatial Semantics and Ontologies (pp. 32-39). https://doi.org/10.1145/2068976.206898
189. Szekely, P., Knoblock, CA, Yang, F., Zhu, X., Fink, EE, Allen, R., & Goodlander, G. (2013, May). Connecting the smithsonian american art to the linked data cloud. In Extended Semantic Web Conference (pp. 593-607). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-38288-8_40
190. Tabula (June 4, 2018). Retrieved May 4, 2022, from https://tabula.technology/
191. The OBO Foundry (n.d.). Relation Ontology. Retrieved May 27, 2022, from https://obofoundry.org/ontology/ro.html
192. The RDF Data Cube Vocabulary (January 16, 2014). Retrieved May 4, 2022, from https://www.w3.org/TR/vocab-data-cube/
193. Thongkrau, T., & Lalitrojwong, P. (2012). Ontopop: An ontology population system for the semantic web. IEICE TRANSACTIONS on Information and Systems, 95(4), 921-931.
194. TopBraid SHACL API (January 11, 2022). Retrieved May 4, 2022, from https://github.com/TopQuadrant/shacl
195. Trade union of railway makers and transport builders of Ukraine (n.d.). Metodychni rekomendatsii z avtomatyzovanoi vydachi ta vidminy poperedzhen na poizd Retrieved May 4, 2022, from http://zalp.org.ua/content/view/513/198/lang,ukrainian/

196. Tutcher, J. (2016). Development of semantic data models to support data interoperability in the rail industry (Doctoral dissertation, University of Birmingham).

197. Verstichel, S., Ongenae, F., Loeve, L., Vermeulen, F., Dings, P., Dhoedt, B., & De Turck, F. (2011). Efficient data integration in the railway domain through an ontology-based methodology. Transportation Research Part C: Emerging Technologies, 19(4), 617-643. https://doi.org/10.1016/j.trc.2010.10.003

198. Ukrainian Railways Joint Stock Company (n.d.). Interactive map of railway history of Ukraine. Retrieved May 4, 2022, from https://www.uz.gov.ua/about/technical_and_social_policy/retro_sector/

199. Ukrainian Railways Joint Stock Company (n.d.). open data. Retrieved May 4, 2022, from https://www.uz.gov.ua/opendata/

200. Ukrainian Railways Joint Stock Company (n.d.). Osobam z invalidistiu. Retrieved May 4, 2022, from https://www.uz.gov.ua/passengers/persons_with_disabilities/

201. Ukrainian Railways Joint Stock Company (November 26, 2020). Ukrzaliznytsia maie namir perevozyty poizdamy ne menshe 50% vnutrishnikh turystiv. Retrieved May 4, 2022, from https://www.uz.gov.ua/press_center/up_to_date_topic/page-4/529731/

202. Umiliacchi, P., Lane, D., Romano, F., & SpA, A. (2011, May). Predictive maintenance of railway subsystems using an Ontology based modelling approach. In Proceedings of 9th world conference on railway research, May (pp. 22-26).

203. Verkhovna Rada of Ukraine (August$_3$1, 2005). Pro zatverdzhennia Instruktsii z rukhu poizdiv i manevrovoi roboty na zaliznytsiakh Ukrainy. Retrieved May 4, 2022, from https://zakon.rada.gov.ua/rada/show/v0507650-05#Text

204. Verkhovna Rada of Ukraine (December 20, 1996). Pro zatverdzhennia Pravyl tekhnichnoi ekspluatatsii zaliznyts Ukrainy. Retrieved May 4, 2022, from https://zakon.rada.gov.ua/laws/show/z0050-97#Text

205. Villegas, M., & Bel, N. (2015). PAROLE/SIMPLE 'lemon'ontology and lexicons. Semantic Web, 6(4), 363-369.

206. Vrandečić, D., Völker, J., Haase, P., Tran Duc, T., & Cimiano, P. (2006). A metamodel for annotations of ontology elements in owl dl. In Meta-modelling and ontologies–Proceedings of the 2nd Workshop on Meta-Modelling–WoMM 2006. Gesellschaft für Informatik e. V.

207. W3C Semantic Web Interest Group (n.d.). Basic Geo (WGS84 lat/long) Vocabulary. Retrieved May 4, 2022, from https://www.w3.org/2003/01/geo/

208. W3C (26 March 2020). Time Ontology in OWL. Retrieved May 27, 2022, from https://www.w3.org/TR/owl-time/

209. Web vocabulary for e-commerce (n.d.). Retrieved May 4, 2022, from https://www.heppnetz.de/projects/goodrelations/

210. Xie, H., & Shen, W. (2006, May). Ontology as a mechanism for application integration and knowledge sharing in collaborative design: a review. In 2006 10th International Conference on Computer Supported Cooperative Work in Design (pp. 1-7). IEEE. https://doi.org/10.1109/cscwd.2006.253214

211. Zedlitz, J., & Luttenberger, N. (2014). Conceptual modelling in UML and OWL-2. International Journal on Advances in Software, 7(1), 182-196.

212. Zhang, S., Boukamp, F., & Teizer, J. (2015). Ontology-based semantic modeling of construction safety knowledge: Towards automated safety planning for job hazard analysis (JHA). Automation in Construction, 52, 29-41.

213. Zhao, L., Ichise, R., Mita, S., & Sasaki, Y. (2014, November). An ontology-based intelligent speed adaptation system for autonomous cars. In Joint International Semantic Technology Conference (pp. 397-413). Springer, Cham. https://doi.org/10.1007/978-3-319-15615-6_30

214. Zheng, J., Harris, MR, Masci, AM, Lin, Y., Hero, A., Smith, B., & He, Y. (2016). The Ontology of Biological and Clinical Statistics (OBCS) for standardized and reproducible statistical analysis. Journal of Biomedical Semantics, 7(1), 1-13. https://doi.org/10.1186/s13326-016-0100-2

215. Zhuchyi, L. I. (2022). Ontological Support for Harmonization and Integration of Ukrzaliznytsia Information Systems Data. Science and Transport Progress, (1 (97)), 32-49. DOI: https://doi.org/10.15802/stp2022/265335

216. Zhuchyi, L. I. Ontologies in the transportation context (2022). Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті. (с. 115) Дніпро

217. Великодний, В. В., Ковдря, Д. В., & Цейтлін, С. Ю. (2017). 10 років розвитку інформаційних технологій залізничної галузі. Залізничний транспорт України, (1), 16-23.

218. Жучий, Л. І. (2020). Розвиток онтологій виробництва та транспорту. Інформаційні технології в металургії та машинобудуванні (р. 152-157). Дніпро

219. Жучий, Л. І. (2018). Концептуальне та онтологічне моделювання залізничного транспорту. Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті (р. 90). Дніпро

220. Жучий, Л. И. (2019). Способы представления онтологий железнодорожного транспорта. Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті (р. 79). Дніпро

221. Жучий, Л. И., & Шинкаренко, В.И. (2020). Сравнительный анализ предметно-ориентированных языков. Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті (с. 84). Дніпро

222. Жучий, Л. И., & Шинкаренко, В.И. (2020). Формализация глоссария онтологическими средствами. Сучасні інформаційні та комунікаційні технології на транспорті, в промисловості та освіті (р. 83). Дніпро

223. Жучий, Л. І., Шинкаренко, В.І., & Іванов, О.П. (2020). Аналіз розвитку онтологій залізничного транспорту. Проблеми та перспективи розвитку залізничного транспорту (р. 377). Дніпро

224. Козаченко, Д. М., Березовий, М. І., Малашкін, В. В., Арбузов, М. А., & Сковрон, І. Я. (2017). Розробка структури типового паспорту під'їзних залізничних колій. Транспортные системы и технологии перевозок, (14), 42-49.

225. Литвин, В. В., Вовнянка, Р. В., & Досин, Д. Г. (2014). Комп'ютерна система автоматизованої розбудови базової онтології CROCUS. Electrotechnic and Computer Systems, (13 (89)), 135-143.

226. Інструкція з забезпечення безпеки руху поїздів при виконанні колійних робіт на залізницях України (2012) Retrieved June 21, 2022, from http://scbist.com/dokumenty-ukrzal-znic/27101-cp-0273-nstrukc-ya-z-zabezpechennya-bezpeki-ruhu-po-zd-v-pri-vikonann-kol-inih-rob-t-na-zal-znicyah-ukra-ni.html

227. Інструкція з улаштування та утримання колії залізниць України (2012) Retrieved June 21, 2022, from http://scbist.com/dokumenty-ukrzal-znic/22091-cp-0269-nstrukc-ya-z-ulashtuvannya-ta-utrimannya-kol-zal-znic-ukra-ni.html

228. Інструкція зі складання графіка руху поїздів на залізницях України, затверджена наказом Укрзалізниці (2002). Retrieved June 21, 2022, from http://scbist.com/dokumenty-ukrzal-znic/21627-cd-0040-nstrukc-ya-z-skladannya-graf-ka-ruhu-po-zd-v-na-zal-znicyah-ukra-ni-zatverdzhena-nakazom-ukrzal-znic-v-d-05-04-2002-170-c.html

229. Организация сотрудничества железных дорог (январь 2009). Анализ параметров, являющихся определяющими для сохранения технической и эксплуатационной совместимости железнодорожной системы колеи 1520 мм и 1435 мм на границе СНГ-ЕС подсистема: инфраструктура. Путь и путевое хозяйство. Retrieved May 27, 2022, from https://osjd.org/api/media/resources/47?action=download

230. Правила перевезення небезпечних вантажів (2014). Retrieved June 21, 2022, from https://www.uz.gov.ua/cargo_transportation/legal_documents/terms_of_freight/page-3/264636/

231. Про затвердження Правил експлуатації власних вантажних вагонів (2015). Retrieved June 21, 2022, from https://zakon.rada.gov.ua/laws/show/z0168-15#Text

232. Про затвердження Правил технічної експлуатації залізничного транспорту промислових підприємств (2010). Retrieved June 21, 2022, from https://zakon.rada.gov.ua/laws/show/z0237-10#Text

233. Шинкаренко, В. И., & Ильман, В. М. (2014). Конструктивно-продукционные структуры и их грамматические интерпретации. II. Уточняющие преобразования. Кибернетика и системный анализ.

Додаток А

Конструктивно-продукційне моделювання документообігу обмежень швидкості руху поїзда

А.1 Породжувальний конструктор формування таблиці csv структури попередження про обмеження швидкості та концептів табличного представлення знань

Тут і далі наведено лише мету конструктора, його початкові умови, вхідні дані, виконавці, правила підстановки та додаткові операції, умови завершення та вихідні дані. Спеціалізація та інтерпретація конструктора виконана аналогічно «конструктору, що породжує формування таблиці бази даних попереджень про обмеження швидкості».

Ціль конструктора – формування таблиці csv структури бланка попереджень та концептів табличного представлення знань. Таблиця CSV відрізняється від бази даних тим, що має опціональний заголовок, роздільник (кома) і символ кінця рядка і не має індексів.

Семантика Λ_{tabl} включає поняття: rowCnt – лічильник рядків, elementCnt – лічильник стовпців.

Початкові умови: t_0 =1, t_1 =0, t_2 =0, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =0,

Нетермінали : TABLE, TUPLE1, LIT, TUPLE, ELEMENt_1, ELEMENT.

Початковий нетермінал – TABLE.

Термінали: eof, eol, value, element, кількість стовпців (b), кількість рядків (f).

Вхідні дані – структура форми попередження ДУ-61 з ІРП та модель табличного представлення знань.

Обмеження задаються атрибутами правил підстановки.

Умова завершення – всі відношення підстановки недоступні.

Вихідні дані – таблиці CSV зі структурою форми попередження ДУ-61 та концептами моделі табличного представлення знань (таблиця 2).

Таблиця Б. 1 Таблиці csv структури ДУ-61 ІРП та концептів табличного представлення знань

Структура ДУ-61 ІДП	Концепти табличного представлення знань
place, speed eol 1,3 eol	class, objectProperty, datatypeProperty eol value, halement, isRepresentedByLiteral eol identifier, hasPart, eol tuple, naming, eol table,, eol

Виконавець – текстовий редактор та зовнішній або програмний виконавець для імпорту.

Правила підстановки:

$s_1 = \langle {}_{LIT}b,{}_{LIT}f\, TABLE\ {}_{t_0}\!\rightarrow TUPLE1 \circ {}_{LIT}b,{}_{LIT}f\, TABLE\rangle,$

$g_1 = \langle @(LIT \downarrow b \downarrow TABLE), @(LIT \downarrow f \downarrow TABLE), := (t_1, 1), := (t_0, 0), := (t_2, 0)\rangle.$

Правило s_1. " TABLE " замінюється на " TUPLE 1 ∘TABLE ".

Зовнішнім виконавцем вводяться атрибути таблиці кількість стовпців (b = 2), кількість рядків (f = 2).

t_1 присвоюється 1. Тому правило s_2 стає доступним.

t_0 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_1 стає недоступним. Правило s_1 має бути недоступним, тому що в ланцюжку залишився нетермінал TABLE.

t_2 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_3 стає недоступним.

Оскільки доступно правило s_2 переходимо до правила s_2.

t_0 =0, t_1 =1, t_2 =0, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =0.

$s_2 = \langle TUPLE1\ {}_{t_1}\!\rightarrow TUPLE \circ TUPLE1\rangle,$

$g_2 = \langle +(rowCnt, rowCnt, 1), := (t_1, 0),$

$:= (t_6, 1), \#(rowCnt, f, := (t_4, 1))\rangle.$

Правило s_2. " TUPLE 1" замінюється на " TUPLE ∘TUPLE 1 ".

Збільшується лічильник rowCnt.

t_1 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_2 стає недоступним.

Якщо лічильник rowCnt дорівнює f, t_4 присвоюється 1.

t_6 присвоюється 1. Тому правило s_4 стає доступним.

Оскільки доступно правило s_4 переходимо до правила s_4.

t_0 =0, t_1 =0, t_2 =0, t_3 =0, t_4 =0, t_5 =0, t_6 =1, t_7 =0.

Продовжимо розглядати приклад реалізації:

$s_4 = \langle TUPLE \ _{t_6}\!\!\rightarrow \ _{LIT}value\, element \circ ',' \circ ELEMENT1\rangle,$

$g_4 = \langle @(LIT \downarrow value \downarrow element)\ ,$

$:= (elementCnt, 1), := (t_6, 0), := (t_7, 1), \#\big(elementCnt, b, := (t_7, 0)\big),$

$\#(elementCnt, b, := (t_2, 1)),\ \#\big(elementCnt, b, := (t_1, 1)\big),$

$\#(elementCnt, b\ AND\ rowCnt, f := (t_5, 1))\rangle.$

Правило s_4. " TUPLE " замінюється на "елемент ∘', ' ∘ELEMENt_1 ".

Зовнішнім виконавцем вводиться літерал значення елемента таблиці.

Лічильник elementCnt присвоюється 1.

t_6 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_4 стає недоступним.

t_7 присвоюється 1. Тому правило s_5 стає доступним.

Якщо лічильник elementCnt дорівнює кількість стовпців (b) початкової умови, то t_7 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_5 стає недоступним.

Якщо лічильник elementCnt дорівнює кількість стовпців (b) початкової умови, то t_2 присвоюється 1. Правило s_3 стає доступним.

Якщо лічильник elementCnt дорівнює кількість стовпців (b) початкової умови, то t_1 присвоюється 1. Правило s_2 стає доступним.

Якщо лічильник elementCnt дорівнює кількість стовпців (b) початкової умови і лічильник rowCnt дорівнює кількість рядків (f), t_5 присвоюється 1.

Таким чином, є три випадки поведінки конструктивної системи.

Переходять до правила s_2 якщо у таблиці 1 стовпець

t_0 =0, t_1 =0, t_2 =0, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =1.

або перехід до правила s_5 якщо у таблиці 2 і більше стовпців

t_0 =0, t_1 =1, t_2 =1, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =0.

Або перехід до правила s_3 якщо лічильник elementCnt дорівнює кількість стовпців (b) і лічильник rowCnt дорівнює кількість рядків (f).

t_0 =0, t_1 =1, t_2 =1, t_3 =0, t_4 =0, t_5 =1, t_6 =0, t_7 =0.

$$s_5 = \langle ELEMENT1 \ {}_{t_7}\!\rightarrow \ {}_{LIT}value\, element \circ', ' \circ ELEMENT1\rangle,$$

$$g_5 = \langle @(LIT \downarrow value \downarrow element), \quad +(elementCnt, elementCnt, 1),$$
$$\#\big(elementCnt, b, \coloneqq (t_7, 0)\big),$$
$$\#(elementCnt, b, \coloneqq (t_2, 1)), \#\big(elementCnt, b, \coloneqq (t_1, 1)\big),$$
$$\#(elementCnt, b\ AND\ rowCnt, f \coloneqq (t_5, 1))\rangle.$$

Правило s_5. «ELEMENt$_1$» замінюється на "element ∘', ' ∘ELEMENt$_1$ ".

Зовнішнім виконавцем вводиться елемент таблиці.

Збільшується лічильник elementCnt.

Якщо лічильник elementCnt дорівнює кількість стовпців (b) початкової умови, то t_7 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_5 стає недоступним.

Якщо лічильник elementCnt дорівнює кількість стовпців (b) початкової умови, то t_2 присвоюється 1. Правило s_3 стає доступним.

Якщо лічильник elementCnt дорівнює кількість стовпців (b) початкової умови, то t_1 присвоюється 1. Правило s_2 стає доступним.

Якщо лічильник elementCnt дорівнює кількість стовпців (b) початкової умови і лічильник rowCnt дорівнює кількість рядків (f), t_5 присвоюється 1.

Таким чином, є три випадки поведінки конструктивної системи.

Переходять до правила s_2 якщо у таблиці 2 стовпця

t_0 =0, t_1 =0, t_2 =0, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =1.

або перехід до правила s_5, якщо в таблиці 3 і більше стовпців.

t_0 =0, t_1 =1, t_2 =1, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =0

або перехід до правила s_3 якщо лічильник elementCnt дорівнює кількість стовпців (b) і лічильник rowCnt дорівнює кількість рядків (f).

t_0 =0, t_1 =1, t_2 =1, t_3 =0, t_4 =0, t_5 =1, t_6 =0, t_7 =0.

А.2 Перетворюючий конструктор зі структури таблиці CSV структури попередження ДУ-61 в метадані таблиці CSV

Будемо називати метаданими таблиці концепти опису таблиці розробки словника онтології джерел.

Мета конструктора – перетворити таблицю структури ДУ-61 на таблицю метаданих ДУ-61.

Семантика Λ_{tabl}включає поняття: header, header1 – імена стовпців таблиці, propPart – частина відношення, що пов'язує відношенням порядку стовпці таблиці, property – відношення, що зв'язує відношенням порядку стовпці таблиці, а також поняття онтологій конструкторів, що породжують.

Початкові умови: $t_0 = 1$, $t_1 = 0$, $t_2 = 0$, $t_3 = 0$, $t_4 = 0$,

Нетермінали : A, B, C, D, E.

Початковий нетермінал – A.

Термінали : element, value, eol, ','.

Вхідні дані – таблиця csv структури бланка ДУ-61 (вихідні дані «конструктора, що породжує формування таблиці csv структури попередження про обмеження швидкості»):

place, speed eol

1,3 eol eof

Виконавець – внутрішній – текстовий редактор та зовнішній або програмний виконавець для імпорту та введення назви таблиці.

Обмеження задаються атрибутами правил підстановки.

Умова завершення – всі елементи таблиці csv структури попередження ДУ-61 перетворені на метадані.

Вихідні дані – таблиця csv метаданих структури бланка ДУ-61 :

Клас, об'єктProperty, datatypeProperty eol

ду61Таблиця,, eol

ду61ЗаголовнийКортеж,, eol

місцеІдентифікатор,, eol

ШвидкістьІдентифікатор, місцеПередШвидкість, eol eof

Правила підстановки:

$$s_1 = \langle A \ {}_{t_0}\!\rightarrow \ {}_{class}value\,element \circ ',' \circ {}_{objectProperty}value\,element \circ ',' \circ {}_{datatypeProperty}value\,element \circ eol \circ {}_{LIT}value\,element \circ ',' \circ {}_{\varepsilon}value\,element \circ ',' \circ {}_{\varepsilon}value\,element \circ eol \circ B \rangle,$$

$$g_1 = \langle @(table\ name), \cdot (LIT \downarrow value \downarrow element, table\ name, 'table'), := (t_1, 1) \rangle.$$

Правило s_1. « A » замінюється на «element∘ ',' ∘element ∘',' ∘element∘ eol ∘element ∘',' ∘element ∘',' ∘element ∘eol ∘B ». Серед них у перших трьох елементів вже є значення заголовків таблиці метаданих " class ", " objectProperty ", " datatypeProperty ". Четвертому елементу буде надано літерал значення операцією конкатенації правила s_1. Два останні елементи мають літерал значення ε.

Зовнішнім виконавцем вводиться ім'я таблиці.

Літералу значення четвертого елемента таблиці присвоюється результат конкатенації імені таблиці та рядка «таблиця».

t_1 присвоюється 1. Тому правило s_2 стає доступним.

$t_0 = 1$, $t_1 = 1$, $t_2 = 0$, $t_3 = 0$, $t_4 = 0$, $t_5 = 0$, $t_6 = 0$, $t_7 = 0$, $t_8 = 0$, $t_9 = 0$.

$$s_2 = \langle B \ {}_{t_1}\!\rightarrow \ {}_{LIT}value\,element \circ ',' \circ$$

$_{\varepsilon value}element \circ ',' \circ \ _{\varepsilon value}element \circ eol \circ C\rangle$,

$g_2 = \langle \cdot (LIT \downarrow value \downarrow element, table\ name, 'headerTuple'), := (t_2, 1)$ $@(LIT \downarrow value \downarrow terminal, header)\rangle$.

Правило s_2. «B» замінюється на «елемент ∘','∘ element ∘',' ∘element ∘EOL ∘C». Серед них першому елементу буде надано літерал значення операцією конкатенації правила s_2. Два останні елементи мають літерал значення ε.

Літералу значення першого елемента таблиці присвоюється результат конкатенації імені таблиці та рядка headerTuple.

t_2 присвоюється 1. Тому правило s_3 стає доступним.

Виконується операція читання @ літерала значення терміналу ланцюжка таблиці вхідних даних у змінну header.

t_0 =1, t_1 =1, t_2 =1, t_3 =0, t_4 =0, t_5 = 0, t_6 = 0, t_7 = 0, t_8 = 0, t_9 = 0.

$s_3 = \langle$C $_{t_2}\rightarrow \ _{LIT value}element \circ ',' \circ$

$_{\varepsilon value}element \circ ',' \circ \ _{\varepsilon value}element \circ eol \circ D\rangle$,

$g_3 = \langle := (\cdot (header, 'identifier'), LIT \downarrow value \downarrow element), := (t_5, 1),$

$:= (t_3, 1), @(terminal), @(header1, LIT \downarrow value \downarrow$

$terminal), \#(terminal, eol, := (t_5, 0))\rangle$.

Правило s_3. «C» замінюється на «element∘ ',' ∘element ∘',' ∘element∘ eol ∘D». Серед них першому елементу буде надано літерал значення операцією конкатенації правила s_3. Два останні елементи мають літерал значення ε.

Літералу значення елемента присвоюється результат конкатенації змінної header і строки 'identifier'

t_3 присвоюється 1. Тому правило s_6 стає доступним.

t_3 присвоюється 1. Тому правило s_4 стає доступним.

Виконується операція читання @ терміналу ланцюжка таблиці вхідних даних для переміщення каретки з комою на елемент.

Змінній header1 присвоюється літерал значення терміналу ланцюжка таблиці вхідних даних (деякий заголовок таблиці).

Якщо операцією читання @ прочитано термінал eol, то t_3 присвоюється 0. Виконання конструктора завершується. Оскільки метадані таблиці формуються згідно з її шапкою.

Якщо операцією читання @ прочитано термінал eol, то t_7 присвоюється 1 для видалення нетермінала D.

Таким чином, є два випадки поведінки конструктивної системи.

Переходять до правила s_4 якщо прочитано черговий елемент шапки таблиці

$t_0=1$, $t_1=1$, $t_2=1$, $t_3=1$, $t_4=0$, $t_5=0$, $t_6=0$, $t_7=0$, $t_8=0$, $t_9=0$.

або завершують обробку таблиці якщо досягнуто терміналу eol ланцюжка таблиці вхідних даних, тобто оброблено всю шапку

$t_0=1$, $t_1=1$, $t_2=1$, $t_3=0$, $t_4=0$, $t_5=0$, $t_6=0$, $t_7=0$, $t_8=0$, $t_9=0$.

Продовжимо розглядати приклад реалізації:

$$s_4 = \langle D \ {}_{t_3}\!\rightarrow \ {}_{LIT}value\, element \circ ',' \circ E \rangle,$$

$$g_4 = \langle := (LIT \downarrow value \downarrow element, \cdot (header1, 'identifier')),$$

$$:= (t_4, 1), @(terminal),$$

$$\#(terminal, eol, := (t_5, 0)), \#(terminal, eol, := (t_9, 1))\rangle.$$

Правило s_4. «D» замінюється на «element ∘','∘ E».

Літералу значення елемента присвоюється результат конкатенації header 1 і строки 'identifier'.

t_4 присвоюється 1. Тому правило s_4 стає доступним.

Якщо операцією читання @ прочитано термінал eol, то t_5 присвоюється 0. Виконання конструктора завершується. Оскільки метадані таблиці формуються згідно з її шапкою.

Якщо операцією читання @ прочитано термінал eol, то t_9 присвоюється 1 для видалення нетермінала F.

t_0 =1, t_1 =1, t_2 =1, t_3 =1, t_4 =1, t_5 = 0, t_6 = 0, t_7 = 0, t_8 = 0, t_9 = 0.

$$s_5 = \langle E \ {}_{t_4}\!\rightarrow \ {}_{LIT}value\, element \circ ',' \circ \ {}_{\varepsilon}value\, element \circ eol \circ F \rangle,$$

$$g_5 = \langle \cdot (propPart, header,' before'),$$

$$\cdot (LIT \downarrow value \downarrow element, propPart, header1),$$

$$:= (t_5, 1), @(LIT \downarrow value \downarrow terminal, header),$$

$$\#\big(terminal, eol, := (t_5, 0)\big), := (header, header1)\ \rangle.$$

Правило s_5. «E» замінюється на « element ∘ ',' ∘ element ∘ eol ∘З ». Серед них першому елементу буде надано літерал значення операцією конкатенації правила s_3. Другий елемент має літерал значення ε.

Змінній propPart присвоюється результат конкатенації змінної header (першого з двох прочитаних заголовків) та рядка «перед».

Літералу значення елемента присвоюється результат конкатенації змінної propPart і змінної header 2 (другий із двох прочитаних заголовків).

t_5 присвоюється 1. Тому правило s_6 стає доступним.

Виконується операція читання @ терміналу ланцюжка таблиці вхідних даних для переміщення каретки з комою на елемент.

Перемінний header 1 присвоюється літерал значення терміналу ланцюжка таблиці вхідних даних (деякий заголовок таблиці).

Якщо операцією читання @ прочитано термінал eol, то t_5 присвоюється 0. Виконання конструктора завершується.

Якщо операцією читання @ прочитано термінал eol, то t_9 присвоюється 1. Тому правило s_7 стає доступним.

Змінній header присвоюється значення header для формування відношення « Стовпець » наступної пари стовпців.

Таким чином, є два випадки поведінки конструктивної системи.

Переходять до правила s_5 якщо прочитано черговий елемент шапки таблиці

t_0 =1, t_1 =1, t_2 =1, t_3 =1, t_4 =1, t_5 = 1, t_6 = 0, t_7 = 0, t_8 = 0, t_9 = 0.

або завершують обробку таблиці якщо досягнуто терміналу eol ланцюжка таблиці вхідних даних, тобто оброблено всю шапку

$t_0 = 1$, $t_1 = 1$, $t_2 = 1$, $t_3 = 1$, $t_4 = 1$, $t_5 = 0$, $t_6 = 0$, $t_7 = 0$, $t_8 = 0$, $t_9 = 1$.

$$s_6 = \left\langle F \ {}_{t_5}\!\rightarrow {}_{LIT}value\, element \circ ',' \circ {}_{\varepsilon}value\, element \circ ',' \circ {}_{\varepsilon}value\, element \circ eol \circ D \right\rangle,$$

$$g_6 = \langle := (\cdot (header, 'identifier'), LIT \downarrow value \downarrow element), := (t_3, 1), \#(terminal, eol, := (t_3, 0)) \rangle.$$

Правило s_6. «C» замінюється на «element∘ ',' ∘element ∘',' ∘element∘ eol ∘D». Серед них першому елементу буде надано літерал значення операцією конкатенації правила s_3. Два останні елементи мають літерал значення ε.

Виконується операція читання @ терміналу ланцюжка таблиці вхідних даних для переміщення каретки з комою на елемент.

Виконується операція читання літерала значення терміналу ланцюжка таблиці вхідних даних у змінну header.

Літералу значення елемента присвоюється результат конкатенації змінної header і рядки ' identifier '

t_3 присвоюється 1. Тому правило s_4 стає доступним.

Якщо операцією читання @ прочитано термінал eol, то t_3 присвоюється 0. Виконання конструктора завершується. Оскільки метадані таблиці формуються згідно з її шапкою.

Якщо операцією читання @ прочитано термінал eol, то t_6 присвоюється 1. Тому правило s_7 стає доступним.

Таким чином, є два випадки поведінки конструктивної системи.

Переходять до правила s_4 якщо прочитано черговий елемент шапки таблиці

$t_0 = 1$, $t_1 = 1$, $t_2 = 1$, $t_3 = 1$, $t_4 = 0$, $t_5 = 0$, $t_6 = 0$, $t_7 = 0$, $t_8 = 0$, $t_9 = 0$.

або завершують обробку таблиці якщо досягнуто терміналу eol ланцюжка таблиці вхідних даних, тобто оброблено всю шапку

$t_0 = 1$, $t_1 = 1$, $t_2 = 1$, $t_3 = 0$, $t_4 = 0$, $t_5 = 0$, $t_6 = 0$, $t_7 = 0$, $t_8 = 0$, $t_9 = 0$.

$s_7 = \langle C \ _{t_6}\rightarrow \varepsilon; D \ _{t_7}\rightarrow \varepsilon; E \ _{t_8}\rightarrow \varepsilon; \ F \ _{t_9}\rightarrow \varepsilon \rangle$,

$g_7 = \langle \varepsilon \rangle$.

А.3 Перетворюючий конструктор з таблиці бази даних попереджень ДУ-61 у загальне представлення таблиць OpenRefine

Мета конструктора – змінити індекс рядка на число.

Початкові умови rowCnt = 0, prevIndex = "рядок", $t_0 = 1$, $t_1 = 0$, початковий нетермінал – A.

Вхідні дані: таблиця баз даних попереджень про обмеження швидкості (Таблиця 2.4).

Вводяться наступні операції над атрибутами:

1) операція нерівності $\neq (a, b)$ повертає істину якщо $a \neq b$;

2) $\#'(a, b)$операція b виконується якщо операція a повертає істину.

Семантика Λ_{tabl}включає поняття: значення, заголовок, індекс, rowCnt = 0 – лічильник рядків, $prevIndex = \emptyset$ – індекс попереднього елемента таблиці, а також поняття онтологій конструкторів, що породжують, $temp$ – змінна для зберігання терміналу ланцюжка вхідних даних.

Нетермінали : A, B.

Початковий нетермінал – A.

Термінали : element, header.

Виконавець – внутрішній OpenRefine та зовнішній або програмний виконавець для імпорту.

Обмеження задаються атрибутами правил підстановки.

Умова завершення – таблиця бланка попередження ДУ-61 перетворена на аналогічну, що має формат загального представлення.

Вихідні дані: таблиця попередження про обмеження швидкості загального формату.

Правила підстановки:

$s_1 = \langle A \ {}_{t_0}\!\rightarrow header \circ B \ \rangle$,

$g_1 = \langle @(LIT \downarrow value \downarrow terminal, LIT \downarrow value \downarrow header)$,

$@(terminal, temp), \#\big(temp, element, := (t_2, 1)\big), \#(temp, header, := (t_1, 1))\rangle$.

Правило s_1. «A» замінюється на « header ∘ B ».

Виконується читання операцією @літерала значення терміналу ланцюжка таблиці бланка ДУ-61 вхідних даних до літералу значення заголовка таблиці загального представлення.

Виконується читання операцією @ланцюжка вхідних даних змінну temp.

Якщо temp є елементом, то t_2 присвоюється 1. Правило s_3 стає доступним.

Якщо temp є заголовком, то t_1 присвоюється 1. Правило s_2 стає доступним.

Таким чином, є два випадки поведінки конструктивної системи.

Переходять до правила s_2 якщо temp «заголовок».

t_0 =1, t_1 =1, t_2 =0, t_3 =0, t_4 =0.

Або переходять до правила s_3 якщо temp «елемент»

t_0 =1, t_1 =0, t_2 =1, t_3 =0, t_4 =0.

$s_2 = \langle B \ {}_{t_1}\!\rightarrow header \circ B\rangle$,

$g_2 = \langle := (LIT \downarrow value \downarrow header, LIT \downarrow value \downarrow$

$temp), @(terminal, temp)$,

$\#\big(temp, element, := (t_2, 1)\big), \#(temp, element, := (t_1, 0))\rangle$.

Правило s_2. «B» замінюється на « header ∘ B ».

Виконується привласнення літерала значення змінної temp до літералу значення заголовка таблиці загального представлення.

Виконується читання операцією @ наступного символу ланцюжка вхідних даних у змінну temp.

Якщо temp є елементом, то t_2 присвоюється 1. Правило s_3 стає доступним.

Якщо temp є елементом, то t_1 присвоюється 0. Правило s_2 стає недоступним.

Таким чином, є два випадки поведінки конструктивної системи.

Повторно виконують правило s_2 якщо temp «заголовок»

t_0 =1, t_1 =1, t_2 =0, t_3 =0, t_4 =0.

Або переходять до правила s_3 якщо temp «елемент»

t_0 =1, t_1 =0, t_2 =1, t_3 =0, t_4 =0.

$s_3 = \langle B \ {}_{t_2}\!\rightarrow element \circ B \rangle$,

$g_3 = \langle \coloneqq (\ LIT \downarrow value \downarrow element, LIT \downarrow value \downarrow temp)$,

$\#'(\neq (prevIndex, LIT \downarrow index \downarrow temp)\ , +(rowCnt, rowCnt, 1)\big)$,

$\coloneqq (prevIndex, LIT \downarrow index \downarrow temp), \coloneqq (t_2, 0), \coloneqq (t_3, 1)\rangle$.

Правило s_3. «B» замінюється на « element ∘ B ».

Якщо літерал індексу терміналу ланцюжка таблиці бланка ДУ-61 вхідних даних не дорівнює pre vIndex (початковими умовами встановлено присвоєння prevIndex = ∅), то значення лічильника rowCnt збільшується на 1. При першому виконанні правила індекс терміналу порівнюється на рівність зі рядком, rowCnt збільшується на одиницю зі значенням «0» і стає 1.

Змінної prevIndex присвоюється літерал індексу терміналу ланцюжка таблиці бланка ДУ-61 вхідних даних.

t_2 присвоюється 0. Тому правило s_3 стає недоступним.

t_3 присвоюється 1. Тому правило s_4 стає доступним.

t_0 =1, t_1 =0, t_2 =0, t_3 =1, t_4 =0.

$s_4 = \langle LIT \downarrow index \downarrow element \ {}_{t_3}\!\rightarrow rowCnt \rangle$,

$g_4 = \langle \coloneqq (t_2, 1), \coloneqq (t_3, 0), @(terminal, temp), \#\big(temp, eof, \coloneqq (t_2, 0)\big), \#\big(temp, eof, \coloneqq (t_5, 1)\big)\rangle$.

Правило s_3. Літерал індексу елемента таблиці замінюється на rowCnt.

t_2 присвоюється 1. Правило s_1 стає доступним.

t_3 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_2 стає недоступним.

Читання наступного символу ланцюжка вхідних даних

Якщо прочитаний операцією @термінал дорівнює eof, то t_2 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_3 стає недоступним. Виконання конструктора завершується.

Якщо прочитаний операцією @термінал дорівнює eof, то t_4 присвоюється 1. Правило s_5 видалення символу B стає доступним.

Таким чином, є два випадки поведінки конструктивної системи.

Переходять до правила s_3 якщо temp «елемент»

t_0 =1, t_1 =0, t_2 =1, t_3 =0, t_4 =0.

Або завершують виконання конструктора, якщо temp – це eof.

t_0 =1, t_1 =0, t_2 =0, t_3 =0, t_4 =1.

$s_5 = \langle B \ _{t_4}\rightarrow eof \rangle$,

$g_5 = \langle \varepsilon \rangle$.

А.4 Перетворюючий конструктор з таблиць csv метаданих структури попередження ДУ-61 та концептів табличного представлення знань у загальне представлення таблиць OpenRefine

Мета конструктора:

1) позначити заголовний кортеж;

2) виключити роздільники та символ перекладу рядка;

3) додати індекси;

Вводяться такі операції:

1) $\rho()$– Повернутися до початку конструкції;

2) $Y(terminal, attribute)$– додає терміналу атрибут

3) $AND(a, b)$– повертає істину, якщо операції aі bповертають істину.

Семантика Λ_{tabl}включає поняття: $rowCnt = 1$ – лічильник рядків, $columnCnt$ – лічильник стовпців, $temp$ – змінна для зберігання терміналу ланцюжка вхідних даних, а також поняття онтологій конструкторів, що породжують.

Початкові умови $rowCnt$= $columnCnt$= 0, $t_0 = 1$, $t_1 = 0$, $t_2 = 0$, $t_3 = 0$,

Вхідні дані: таблиці csv метаданих структури попередження ДУ-61 та концептів табличного представлення знань.

Клас, об'єктProperty, datatypeProperty eol

ду61Таблиця,, eol

ду61ЗаголовнийКортеж,, eol

місцеІдентифікатор,, eol

ШвидкістьІдентифікатор, місцеПередШвидкість, eol eof

Нетермінали : A, B.

Початковий нетермінал – A.

Термінали : header, element, value, index.

Виконавець: внутрішній – OpenRefine та зовнішній чи програмний виконавець для імпорту.

Обмеження задаються атрибутами правил підстановки.

Умова завершення – всі елементи таблиці csv метаданих структури попередження ДУ-61 та концептів табличного представлення знань перетворені на таблицю, що має формат загального представлення.

Вихідні дані – таблиці загального представлення.

Правила підстановки:

$s_1 = \langle A \ {}_{t_0}\!\rightarrow header \circ B \rangle$,

$g_1 = \langle @(lit \downarrow value \downarrow terminal, lit \downarrow value \downarrow header)$,

$+(columnCnt, 1) \coloneqq (t_1, 1)\rangle$.

Правило s_1. «A» замінюється на « header ∘ B ».

Виконується операція читання @ літерала значення терміналу ланцюжка таблиці вхідних даних літералу значення заголовка таблиці загального формату.

t_1 присвоюється 1. Правило s_2 стає доступним.

Збільшується лічильник стовпців columnCnt.

t_0 =1, t_1 =1, t_2 =0, t_3 =0, t_4 =0, t_5 =0.

Розглянемо приклад реалізації:

$s_2 = \langle B \ {}_{t_1}\!\rightarrow header \circ B \rangle$,

$g_2 = \langle @(terminal), @(lit \downarrow value \downarrow terminal, lit \downarrow value \downarrow header),$
$+(columnCnt, 1), \#(columnCnt, 3, \coloneqq (t_1, 0)),$
$\#\big(columnCnt, 3, \coloneqq (t_2, 1)\big), \#(columnCnt, 3, \coloneqq (columnCnt, 0))\rangle$.

Правило s_2. « B » замінюється на « header ∘ B ».

Виконується операція читання @ терміналу ланцюжка таблиці вхідних даних для перестановки каретки з комою на елемент таблиці.

Виконується операція читання @ літерала значення терміналу ланцюжка таблиці вхідних даних літералу значення заголовка таблиці загального формату.

Збільшується лічильник стовпців columnCnt.

Якщо лічильник columnCnt дорівнює 3 (бо у сформованих таблицях csv три стовпці: «клас», «властивість даних» та «об'єктна властивість»), t_1 присвоюється 0. Правило s_2 стає недоступним.

Якщо лічильник columnCnt дорівнює 3, t_2 присвоюється 1. Правило s_3 стає доступним.

Якщо лічильник columnCnt дорівнює 3, columnCnt присвоюється 0.

Таким чином, є два випадки поведінки конструктивної системи.

Переходять до правила s_2 якщо columnCnt не дорівнює 3

t_0 =1, t_1 =1, t_2 =0, t_3 =0, t_4 =0, t_5 =0.

Або переходять до правила s_3 якщо columnCnt дорівнює 3.

t_0 =1, t_1 =0, t_2 =1, t_3 =0, t_4 =0, t_5 =0.

$s_3 = \langle B \ _{t_2}\rightarrow element \circ B\rangle$,

$g_3 = \langle @(terminal), @(lit \downarrow value \downarrow terminal, lit \downarrow value \downarrow element),$
$+(columnCnt, 1), Y(element, index)\#(columnCnt, 4, +(rowCnt, rowCnt, 1)),$
$\#(columnCnt, 4, \coloneqq (columnCnt, 1)), \coloneqq (t_2, 0),$
$\coloneqq (t_3, 1), \#\big(terminal, eof, \coloneqq (t_3, 0)\big), \#(terminal, eof, \coloneqq (t_4, 1)))\rangle$.

Правило s_3. « B » замінюється на « header ∘ B ».

Виконується операція читання @ терміналу ланцюжка таблиці вхідних даних для перестановки каретки із символу «комою» або eol на елемент таблиці.

Виконується операція читання @ літерала значення терміналу ланцюжка таблиці вхідних даних літералу значення елемента таблиці загального формату.

Збільшується лічильник стовпців columnCnt.

Елементу таблиці додається операцією Y атрибут індекс.

Якщо лічильник columnCnt дорівнює 4, лічильник rowCnt збільшується.

Якщо лічильник columnCnt дорівнює 4, columnCnt присвоюється 1.

t_2 присвоюється 0. Правило s_3 стає недоступним.

t_3 присвоюється 1. Правило s_4 стає доступним.

Якщо прочитано символ eof, то t_3 присвоюється 0.

Якщо прочитаний символ eof, то t_4 присвоюється 1. Для видалення та елемента, згенерованого при читанні eof.

Таким чином, є два випадки поведінки конструктивної системи.

Переходять до правила s_4 якщо прочитаний не eof

t_0 =1, t_1 =0, t_2 =0, t_3 =1, t_4 =0

Або переходять до правила s_4, якщо прочитано eof.

t_0 =1, t_1 =0, t_2 =0, t_3 =0, t_4 =1

$s_3 = \langle LIT \downarrow index \downarrow terminal \ _{t_3}\rightarrow rowCnt\rangle,$

$g_3 = \langle \coloneqq (t_2, 1), \coloneqq (t_3, 0)\rangle.$

Правило s_3. Літерал атрибута індексу терміналу замінюється на rowCnt.

t_0 присвоюється 1. Тому правило s_1 стає доступним.

t_2 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_3 стає недоступним.

t_0 =1, t_1 =0, t_2 =1, t_3 =0, t_4 =0

$s_4 = \langle element \circ B \ _{t_4}\rightarrow eof\rangle,$

$g_4 = \langle \varepsilon\rangle.$

Правило s_4. В замінюється на eof.

Термінал елемента зі значенням eof замінюється на ε.

Завершується виконання конструктора.

Продовжимо розглядати приклад реалізації:

А.5 Перетворюючий конструктор із загального представлення таблиць OpenRefine у словник онтології обмеження швидкості руху поїздів

Онтологія OWL складається з трійок (об'єкт предикат суб'єкт). Де об'єкт – це ресурс (не властивість) із типом чи ні нього, предикат – це ресурс типу «властивість», а суб'єкт – це ресурс (не властивість) чи літерал.

Мета конструктора: згенерувати префікси словника; перетворити дані стовпця «клас» на класи словника; перетворити дані стовпця « objectProperty » у ObjectProperty словника; перетворити дані стовпця « datatypeProperty » у DatatypeProperty словника.

Виконавець – внутрішній RDF Extension OpenRefine та зовнішній або програмний виконавець для імпорту.

Семантика Λ_{tabl}включає поняття: змінні class, objProp, dataProp для зберігання класів, об'єктних та властивостей даних відповідно, flag = 2 – змінна

для зберігання індексу відношення підстановки правила, а також поняття онтологій конструкторів, що породжують.

Вхідні дані: метадані таблиці структури бланка попередження про обмеження швидкості та таблиця концептів моделі табличного представлення знань (Таблиця Б. 2).

Таблиця Б. 2 метадані таблиці структури бланка попередження про обмеження швидкості та таблиця концептів моделі табличного представлення знань

Клас	objectProperty	datatypeProperty
ду61Таблиця	ε	ε
ду61ЗаголовнийКортеж	ε	ε
місцеІдентифікатор	ε	ε
ШвидкістьІдентифікатор	місцеПередШвидкість	ε

eof

Вводиться операція читання @̃()аналогічна @(), служить для вказівки елемент таблиці в певному стовпці. Початкові умови: t_0 =1, t_1 = 0, t_2 = 0, t_3 = 0, t_4 = 0, t_5 = 0, t_6 = 0, t_7 = 0, t_8 = 0 Нетермінали : A, B, LIT. Початковий нетермінал – A. Термінали: eof – кінець файлу, element. Обмеження задаються атрибутами правил підстановки. Умова завершення – всі елементи таблиць метаданих структури попередження ДУ-61 та концептів табличного представлення перетворені на словник онтології. Вихідні дані представлені на Рисунках Б. 1 та Б. 2:

```
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix table:  <http://example.org/table#> .

table:du61table a  owl:Class .
table:du61headerTuple a  owl:Class .
table:placeIdentifier a  owl:Class .
table:speedIdentifier a  owl:Class .
table:placeBeforeSpeed a    owl:ObjectProperty .
```

Рисунок Б. 1 Реалізація «перетворюючого конструктора із загального представлення таблиць OpenRefine у словник онтології» для структури таблиці ДУ-61

```
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix table:  <http://example.org/table#> .

table:table  a  owl:Class .
table:hasElement  a    owl:ObjectProperty .
table:isRepresentedByLiteral  a     owl:DatatypeProperty .
```

Рисунок Б. 2 Реалізація «перетворюючого конструктора із загального представлення таблиць OpenRefine у словник онтології» для концептів табличного представлення знань

Правила підстановки:

$$s_1 = \langle A \ {}_{t_0}\!\rightarrow {}_{@prefix}value\,element \circ {}_{owl:}value\,element \circ {}_{<http://www.w3.org/2002/07/owl\#>}value\,element \circ {}_{.}value\,element \circ eol \circ {}_{@prefix}value\,element \circ {}_{rdf:}value\,element \circ {}_{<http://www.w3.org/1999/02/22-rdf-syntax-ns\#>}value\,element \circ {}_{.}value\,element \circ eol \circ {}_{@prefix}value\,element \circ {}_{table:}value\,element \circ {}_{<http://example.org/table\#>}value\,element \circ {}_{.}value\,element \circ \circ eol \circ B \rangle,$$

$g_1 = \langle := (t_1, 1) \rangle$.

Правило s_1. A замінюється на префікси онтології та термінали префіксів та нетермінал B.

t_1 присвоюється 1. Тому правило s_2 стає доступним.

t_0 =1, t_1 =1, t_2 =0, t_3 =0, t_4 =0, t_5 =0, t_6 =1, t_7 =0, t_8 = 0

$s_2 = \langle \varepsilon \ {}_{t_1}\!\rightarrow \varepsilon \rangle$,

$g_2 = \langle := (t_1, 0), @(terminal), @(terminal), @(terminal), \#(flag, 2, t_2, 1), \#(flag, 3, t_3, 1), \#(flag, 4, t_4, 1) \rangle$.

Правило s_2'. Використовується для "пропуску" заголовкового кортежу таблиці. Таблиця містить три стовпці, а правило три операції читання для переміщення каретки на рядок зі значеннями.

t_1 присвоюється 0. Тому правило s_2 стає недоступним.

Якщо змінній flag = 2, t_2 присвоюється 1, то правило s_3 стає доступним.

Якщо змінній flag = 3, t_3 присвоюється 1, то правило s_4 стає доступним.

Якщо змінній flag = 3, t_4 присвоюється 1, то правило s_5 стає доступним.

t_0 =1, t_1 =0, t_2 =1, t_3 =0, t_4 =0, t_5 =0, t_6 =1, t_7 =0, t_8 = 0

$s_3 = \langle B\ {}_{t_2}\!\rightarrow element \circ {}_{a}value\, element \circ {}_{owl:Class}value\, element \circ {}_{.}value\, element \circ eol \circ B \rangle$.

$g_3 = \langle \widetilde{@}(LIT \downarrow value \downarrow terminal, class), \#(lit \downarrow value \downarrow terminal, \varepsilon, t_7, 1), \cdot (LIT \downarrow value \downarrow element, ' table:' , class), @(terminal), @(terminal), \#(terminal, eof, \rho()), \#(terminal, eof, := (t_1, 1)), \#(terminal, eof, := (t_2, 0)), \#(terminal, eof, := (flag, 3)), \#(terminal, eof, := (t_5, 1)) \rangle$.

Правило s_3. замінюється на термінали element, « a owl : Class.» і нетермінали B і C. Де "owl:" – це простір імен схеми онтологій owl.

Виконується читання літералу значення терміналу ланцюжка вхідних даних операцією $\widetilde{@}$ у змінну class.

Якщо літерал значення терміналу дорівнює ε, то t_8 присвоюється 1. Тому правило s_7 стає доступним. Необхідно читати наступний рядок у цьому стовпці.

Літералу значення елемента присвоюється результат конкатенації «table:» та змінної class. Де «table:» – це простір імен онтології.

Для переводу каретки двічі виконується операція читання @(), щоб читати операцією $\widetilde{@}()$в першому стовпці таблиць метаданих і концептів табличного представлення знань. Тому що перший стовпець цих таблиць має назву «клас».

Якщо термінал є eof, виконується операція $\rho()$повернення до початку ланцюжка таблиці вхідних даних.

Якщо термінал є eof, то t_1 присвоюється 1. Тому правило s_2 стає доступним. Приступають до генерації об'єктних відношень онтології.

Якщо термінал є eof то змінної flag присвоюється 3.

Якщо термінал є eof, то t_2 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_3 стає недоступним.

Якщо термінал є eof, то t_5 присвоюється для видалення елемента, згенерованого при читанні eof. Тому правило s_6 стає доступним. Завершується виконання генерації класів словника онтології.

Таким чином, є два випадки поведінки конструктивної системи.

Ітеративне виконання правила s_2

$t_0 = 1, t_1 = 0, t_2 = 1, t_3 = 0, t_4 = 0, t_5 = 0, t_6 = 0, t_7 = 0, t_8 = 0$

або перехід до правила s_4 коли досягнуто символу eof і здійснено повернення до початку ланцюжка.

$t_0 = 1, t_1 = 1, t_2 = 0, t_3 = 0, t_4 = 0, t_5 = 1, t_6 = 0, t_7 = 0, t_8 = 0$

$$s_4 = \langle B \ {}_{t_3}\!\rightarrow element \circ {}_{a}value\, element \circ {}_{owl{:}ObjectProperty}value\, element \circ {}_{.value}element \circ eol \circ B\rangle,$$

$$g_4 = \langle @(terminal), \widetilde{@}(LIT \downarrow value \downarrow terminal, objProp),$$
$$\#(lit \downarrow value \downarrow terminal, \varepsilon, t_7, 1),$$
$$\cdot (LIT \downarrow value \downarrow element, ' table:', objProp), @(terminal), @(terminal),$$
$$\#(terminal, eof, \rho()), \#\big(terminal, eof, := (t_1, 1)\big),$$
$$\#\big(terminal, eof, := (t_3, 0)\big),$$
$$\#\big(terminal, eof, := (flag, 4)\big), \#\big(terminal, eof, := (t_5, 1)\big)\rangle.$$

Правило s_4. Замінюється на термінали element, « a owl : Class.» та нетермінал B. _

Для переводу каретки виконується операція для читання @таблиці вхідних даних.

Виконується читання терміналу операцією @в змінну objProp тому, що другий стовпець таблиці вхідних даних має назву «об'єктна властивість».

Якщо літерал значення терміналу дорівнює ε, то t_8 присвоюється 1. Тому правило s_7 стає доступним. Необхідно читати наступний рядок у цьому стовпці.

Літералу значення елемента присвоюється результат конкатенації «table:» та змінної objProp.

Для перекладу каретки один раз виконується операція читання @,щоб робити читання операцією @у другому стовпці таблиць метаданих та концептів табличного представлення знань. Тому що в другому стовпчику цих таблиць має назву «об'єктна властивість».

Якщо термінал є EOF, виконується операція $\rho()$повернення до початку ланцюжка таблиці вхідних даних.

Якщо термінал є EOF, то t_1 присвоюється 1. Тому правило s_2 стає доступним. Приступають до генерації об'єктних властивостей онтології.

Якщо термінал є EOF то змінної flag присвоюється 4.

Якщо термінал є EOF, то t_5 присвоюється для видалення елемента, згенерованого при читанні eof. Тому правило s_6 стає доступним.

Таким чином, є два випадки поведінки конструктивної системи.

Ітеративне виконання правила s_3

$t_0=1$, $t_1=0$, $t_2=1$, $t_3=1$, $t_4=0$, $t_5=0$, $t_6=1$, $t_7=0$, $t_8=0$

або перехід до правила s_4 коли досягнуто символ EOF і здійснено повернення до початку ланцюжка.

$t_0=1$, $t_1=1$, $t_2=1$, $t_3=0$, $t_4=0$, $t_5=1$, $t_6=0$, $t_{7=}0$, $t_{8=}0$.

$$s_5 = \langle B \ {}_{t_4}\!\rightarrow element \circ 'a' \circ 'owl{:}\, DatatypeProperty' \circ '.' \circ eol \circ B \rangle,$$

$$g_5 = \langle @(terminal), @(terminal), \tilde{@}(terminal, datProp),$$

$$\#(lit \downarrow value \downarrow terminal, \varepsilon, t_8, 1),$$

$$\cdot (LIT \downarrow value \downarrow element, ' table{:}' \ , datProp)_{,,} \#\big(terminal, EOF, := (t_6, 1)\big)\rangle.$$

Правило s_5. Замінюється на термінали element «a owl:DatatypeProperty.» та нетермінал B.

Для переводу каретки виконується дві операції читання @ таблиці вхідних даних.

Виконується читання терміналу операцією @̃в змінну datProp тому, що третій стовпець таблиці вхідних даних має назву «властивість даних».

Якщо літерал значення терміналу дорівнює ε, то t_8 присвоюється 1. Тому правило s_7 стає доступним. Необхідно читати наступний рядок у цьому стовпці.

Літералу значення елемента присвоюється результат конкатенації «table:» та змінної datProp.

Якщо термінал є eof, то t_7 присвоюється 1. Тому правило s_6 стає доступним. Завершується виконання конструктора.

t_0 =1, t_1 =1, t_2 =1, t_3 =1, t_4 =0, t_5 =0, t_6 =1, t_7 =1, t_8 = 0.

$$s_6 = \langle element \circ' a' \circ' owl{:}\, Class' \circ '.' \circ eol \circ B \ {}_{t_5}\!\rightarrow B;,$$

$$element \circ' a' \circ' owl{:}\, DatatypeProperty' \circ '.' \circ eol \circ B \ {}_{t_7}\!\rightarrow \varepsilon\rangle.$$

$$g_6 = \langle \varepsilon \rangle.$$

Правило s_6. t_1 присвоюється 1. Тому правило s_2 стає доступним.

Конструктивна операція:

$$s_7 = \langle \tilde{\delta}(terminal) \ {}_{t_8}\!\rightarrow \delta(terminal) \circ \delta(terminal) \circ \tilde{\delta}(terminal)\rangle,$$

$$g_7 = \langle \#(\neq (lit \downarrow value \downarrow terminal, \varepsilon), t_8, 0), \#(terminal, eof, t_7, 0)\rangle.$$

Правило s_7. Конструктивна операція використовується для читання терміналів ланцюжка таблиці вхідних даних у певному стовпці до тих пір, поки не зустрінеться термінал з літералом значення не рівним ε. Правило виконується рекурсивно доки не зустріне термінал з літералом значення не рівним ε.

Необхідно «пропустити» два термінали та прочитати новий термінал у цьому ж стовпці.

А.6 Перетворюючий конструктор із загального представлення таблиць OpenRefine в екземпляри онтології

Мета конструктора – перетворення табличних у дані RDF.

Семантика Λ_{tabl}включає поняття: idCnt =0, idCnt1=1, colCnt =0, colCnt1=0, colCnt2=0, colCnt3=1 – лічильники стовпців таблиці,, idLitPart – частина значень назви стовпця таблиці, rowCnt =1, rowCnt$_1$, rowCnt2, rowCnt3 – лічильники рядків таблиці, valuePart1, valuePart2, value1Part1, value Part2, value2Part1, value2Part2, value3Part1, value3Part1 4 Part1, value4Part2 – частини екземплярів представляють назви стовпців таблиці, valueLitPart – частина значень елементів таблиці, temp – змінна зберігання елементів таблиці, лічильники n = -1, k =0, і поняття онтологій конструкторів.

Початкові умови: t_0 =1, t_1 =0, t_2 =1, t_3 =1, t_4 =1, t_5 =1, t_6 =0, t_7 =0, t_8 =0, t_9 = 1, t_{10} = 1, t_{11} = 1, t_{12} = 0

Вихідні дані представлені на Рисунку Б. 4.

```
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix table:  <http://example.org/table#> .

table:tableDU61   a        table:table .
table:tableDU61 table:hasPart table:tuple1 .
table:tableDU61 table:hasPart table:tuple2 .
table:tuple0   a        table:tuple .
table:tuple1   a        table:tuple .

table:tuple0 table:hasElement  table:identifier1 .
table:identifier1   a        table:identifier .
table:identifier1 table:isrepresentedbyliteral “place” .
table:identifier1 table:naming table:value1 .

table:tuple1 table:hasElement  table:value1.1 .
table:value1.1   a        table:value .
table:value1.1 table:isrepresentedbyliteral “60” .


table:tuple0 table:hasElement  table:identifier2 .
table:identifier2   a        table:identifier .
table:identifier2 table:isrepresentedbyliteral “speed” .
table:identifier2 table:naming table:value2.1 .


table:tuple1 table:hasElement  ta-ble:value2.1 .
table:value2.1   a        table:value .
table:value2.1 table:isrepresentedbyliteral “23km” .


table:identifier1 table:placeBeforeSpeed table:identifier2 .
table:value1 table:placeBeforeSpeed table:value2 . #end of file
```

Рисунок Б. 3 Реалізація «перетворюючого конструктора із загального представлення таблиць OpenRefine в екземпляри онтології» для ДУ-61

Вхідні дані – бланк попередження про обмеження швидкості (Таблиця 2.4). Початковий нетермінал – A.

Термінали: id, id1, id2, id3 – екземпляри, що представляють назви стовпців таблиці, idlit – значення назви стовпця таблиці, value, value1, value2, value3, value4 – екземпляри, що представляють елементи таблиці, valuelit – значення елементів таблиці, tuple кінець рядка, eof – кінець файлу.

Виконавець – внутрішній RDF Extension OpenRefine та зовнішній або програмний виконавець для імпорту та введення відношень.

Вводиться операція $AND(a, b)$ – повертає істину, якщо операції aі b повертають істину.

Обмеження: всі класи та відношення належать словнику; задаються атрибутами правил підстановки.

Умова завершення – всі елементи таблиці бланка попередження ДУ-61 перетворено на екземпляри онтології. Правила підстановки:

$$s_1 = \Big\langle A \ {}_{t_0}\!\rightarrow {}_{@prefix}value\,element \circ {}_{owl:}value\,element \circ$$

$${}_{<http://www.w3.org/2002/07/owl\#>}value\,element$$

$${}_{.}value\,element \circ eol \circ {}_{@prefix}value\,element \circ$$

$${}_{rdf:}value\,element \circ {}_{<http://www.w3.org/1999/02/22-rdf-syntax-ns\#>}value\,element \circ$$

$${}_{.}value\,element \circ eol \circ$$

$${}_{@prefix}value\,element \circ {}_{table:}value\,element \circ$$

$${}_{<http://\ example.org/table\#>}value\,element \circ {}_{.}value\,element \circ$$

$${}_{table:tuple0}value\,element \circ {}_{a}value\,element \circ {}_{table:tuple}value\,element \circ {}_{.}value\,element \circ eol$$

$${}_{table:table}value\,element \circ {}_{a}value\,element \circ {}_{table:table}value\,element \circ {}_{.}value\,element \circ eol$$

$$\circ B \circ {}_{\#oef}value\,element\Big\rangle.$$

$$g_1 = \langle \coloneqq (t_1, 1), @(terminal, tmp) \rangle.$$

Правило s_1. А замінюється на префікси онтології, нетермінал і коментар з позначенням кінця файлу # eof.

t_1 присвоюється 1. Тому правило s_2 стає доступним.

Виконується операція читання @терміналу ланцюжка вхідних даних змінну temp.

t_0 =1, t_1 =1, t_2 =0, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =0, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 0, t_{12} = 0

$$s_2 = \langle B \ {}_{t_1}\!\rightarrow {}_{table:tuple0}value\,element \circ,$$

$${}_{table:hasElement}value\,element \circ id \circ '.' \circ {}_{eol}value\,element \circ$$

$$id \circ {}_{a}value\,element \circ {}_{tabale:identificator}value\,element \circ {}_{.}value\,element \circ eol \circ B\rangle.$$

$$g_2 = \langle +(idCnt, idCnt, 1),$$

$$\cdot (lit \downarrow value \downarrow id,' table{:}\, identifier', idCnt), @(terminal, temp),$$

$$\#(\neq (temp, header), \coloneqq (t_1, 0)),$$

$$\#(\neq (temp, header), \coloneqq (t_6, 1)), \#(\neq (temp, header), \rho())\rangle.$$

Правило s_2. У замінюється на « table : tuple 0 table : hasElement » термінал id, точку, eol, і нетермінал.

Збільшується лічильник idCnt.

Літералу значення терміналу id присвоюється результат конкатенації « table: identifier » та idCnt.

Виконується операція читання @ наступного терміналу ланцюжка вхідних даних у змінну temp.

Якщо temp не є заголовком, то t_1 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_2 стає недоступним.

Якщо temp не є заголовком, то t_2 присвоюється 1. Тому правило s_3 стає доступним.

Якщо temp не є заголовком, виконується операція $\rho()$повернення до читання початку ланцюжка таблиці вхідних даних.

Таким чином, є два випадки поведінки конструктивної системи.

Послідовне виконання правила s_2 для всіх заголовків таблиці

$t_0 = 1, t_1 = 1, t_2 = 0, t_3 = 0, t_4 = 0, t_5 = 0, t_6 = 0, t_7 = 0, t_8 = 0, t_9 = 0, t_{10} = 0, t_{11} = 0, t_{12} = 0$

або перехід до правила s_3, якщо заголовки закінчилися.

$t_0 = 1, t_1 = 0, t_2 = 1, t_3 = 0, t_4 = 0, t_5 = 0, t_6 = 0, t_7 = 0, t_8 = 0, t_9 = 0, t_{10} = 0, t_{11} = 0, t_{12} = 0$

$$s_3 = \langle B \xrightarrow[t_2]{} id1 \circ {}_{table:isRepresentedByLitertal}value\ element \circ idlit \circ {}_{value}element \circ eol \circ B \rangle,$$

$$g_3 = \langle @(terminal, temp), +(idCnt1, idCnt1, 1), \#(idCnt1, idCnt, t_2, 0),$$
$$\#(idCnt1, idCnt, t_3, 1, \#(idCnt1, idCnt, \rho()),$$
$$\cdot (lit \lrcorner value \lrcorner id1, ' table: identifier', idCnt1),$$
$$\cdot (idLitPart, '"', LIT \lrcorner value \lrcorner temp), \cdot (lit \lrcorner value \lrcorner idlit, idlitPart, '"') \rangle.$$

Правило s_3. В замінюється на id 1, table : isRepresentedByLiteral idlit, точку, символ завершення рядка eol та нетермінал B.

Виконується операція читання@ терміналу ланцюжка вхідних даних у змінну temp.

Збільшується лічильник $idCnt_1$.

Якщо лічильник $idCnt_1$ дорівнює idCnt, t_2 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_3 стає недоступним.

Якщо лічильник $idCnt_1$ дорівнює idCnt, t_3 присвоюється 1. Тому правило s_4 стає доступним.

Якщо лічильник $idCnt_1$ дорівнює idCnt, виконується операція $\rho()$повернення до читання початку ланцюжка таблиці вхідних даних.

Літералу значення терміналу id 1 присвоюється результат конкатенації «table:identifier» та $idCnt_1$.

Змінній idLitPart присвоюється результат конкатенації «"» (лапки) і літералу значення терміналу.

Літералу значення терміналу idlit присвоюється результат конкатенації idLitPart і «"».

Таким чином, є два випадки поведінки конструктивної системи.

Послідовне виконання правила s_3 для всіх заголовків таблиці

t_0 =1, t_1 =0, t_2 =1, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =0, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 0, t_{12} = 0

або перехід до правила s_4 якщо $idCnt_1$ дорівнює idCnt.

t_0 =1, t_1 =0, t_2 =0, t_3 =1, t_4 =0, t_5 =0, t_6 =0, t_7 =0, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 0, t_{12} = 0

$$s_4 = \langle B \ {}_{t_3}\!\rightarrow id2 \circ relation \circ id3 \circ \ {}_{.value}element \circ eol \circ B \rangle,$$

$$g_4 = \langle @(terminal), \cdot (lit \downarrow value \downarrow id2, ' table: identifier', idCnt1),$$

$$@(lit \downarrow value \downarrow relation), +(idCnt1, idCnt1,1), @(terminal),$$

$$\cdot (lit \downarrow value \downarrow id3, , ' table: identifier', idCnt), \#(idCnt1, idCnt, t_3, 0),$$

$$\#(idCnt1, idCnt, t_4, 1), -(idCnt1, idCnt1,1)\rangle.$$

Правило s_4. В замінюється на « id 2∘ relation ∘ id 3∘ '.' ∘ eol ∘ B ».

Для переводу каретки наступного символу ланцюжка виконується операція читання @ терміналу ланцюжка вхідних даних.

Літералу значення терміналу id 2 присвоюється результат конкатенації « table : identifier » та лічильника $idCnt_1$.

Зовнішнім виконавцем вводиться вираз зв'язку ідентифікаторів (має вигляд *before* і належить словнику) в літерал значення термінала relation.

Збільшується лічильник $idCnt_1$.

Для переведення каретки на наступний символ ланцюжка виконується операція читання @терміналу ланцюжка вхідних даних.

Літералу значення терміналу id 3 присвоюється результат конкатенації «table:identifier» та лічильника $idCnt_1$.

Якщо лічильник $idCnt_1$ дорівнює idCnt, то t_3 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_4 стає недоступним.

Якщо лічильник $idCnt_1$ дорівнює idCnt, то t_4 присвоюється 1. Тому правило s_5 стає доступним.

Зменшується лічильник $idCnt_1$.

Таким чином, є два випадки поведінки конструктивної системи.

Послідовне виконання правила s_5 для всіх заголовків таблиці

t_0 =1, t_1 =0, t_2 =0, t_3 =1, t_4 =0, t_5 =0, t_6 =0, t_7 =0, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 0, t_{12} = 0

або перехід до правила s6 якщо $idCnt_1$ дорівнює idCnt -1.

t_0 =1, t_1 =0, t_2 =0, t_3 =0, t_4 =1, t_5 =0, t_6 =0, t_7 =0, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 0, t_{12} = 0

$$s_5 = \langle B \ {}_{t_4}\!\rightarrow value \circ {}_{table:isRepresentedByLitertal}\!value\, element \circ valuelit \circ {}_{.value}\!element \circ eol \circ value \circ {}_{a value}\!element \circ {}_{table:value}\!value\, element \circ {}_{.value}\!element \circ eol \circ B \rangle.$$

$$g_5 = \langle @(terminal, temp), +(colCnt, colCnt, 1), \#(colCnt, idCnt - 1, +(rowCnt, rowCnt, 1)), \cdot (valuePart1,' table: value', colCnt), \cdot (valuePart2, valuePart1,' .'),$$
$$\cdot (lit \downarrow value \downarrow value, valuePart2, rowCnt),$$
$$\cdot (valueLitPart,' "', LIT \downarrow value \downarrow temp),$$
$$\cdot (lit \downarrow value \downarrow valuelit, valueLitPart, '"'),$$
$$@(terminal, temp), := (t_4, 0), := (t_5, 1)\rangle.$$

Правило s_5. Генерація трійок з літералами значень таблиці виконується за допомогою двох правил, щоб не згенерувати трійку типу eof isRepresentedByLiteral » і дізнатися кількість рядків у таблиці.

В замінюється на «value ∘ ' table:isRepresentedByLiteral ' ∘valuelit ∘'.' ∘eol ∘B».

Виконується операція читання @ термінала ланцюжка вхідних даних у змінну temp. Збільшується лічильник colCnt.

Змінній $valuePart_1$ присвоюється результат конкатенації «table:value» та лічильника colCnt. Змінній $valuePart_2$ присвоюється результат конкатенації $valuePart_1$ і крапки.

Літералу значення терміналу value присвоюється результат конкатенації $valuePart_2$ і лічильника rowCnt. Змінній valueLitPart присвоюється результат конкатенації лапки та літералу значення терміналу вхідних даних.

Літералу значення терміналу valuelit присвоюється результат конкатенації значенняLitPart і лапки.

Виконується операція читання @ наступного (для правила s_8 буде поточним) терміналу ланцюжка вхідних даних.

t_4 присвоюється 0. Тому правило s_5 стає недоступним.

t_5 присвоюється 1. Тому правило s_8 стає доступним.

$t_0 = 1$, $t_1 = 0$, $t_2 = 0$, $t_3 = 0$, $t_4 = 0$, $t_5 = 1$, $t_6 = 0$, $t_7 = 0$, $t_8 = 0$, $t_9 = 0$, $t_{10} = 0$, $t_{11} = 0$, $t_{12} = 0$

$$s_6 = \langle B \ {}_{t_5}\!\rightarrow value \circ {}_{table:isRepresentedByLiteral}{}^{value}element \circ$$

$$valuelit \circ '.' \circ eol \ \circ$$

$$value \circ {}_{a}{}^{value}element \circ {}_{table:value}{}^{value}element \circ {}_{.}{}^{value}element \circ eol \circ B\rangle.$$

$$g_6 = \langle +(colCnt, colCnt, 1),$$

$$\cdot (valuePart1,' table{:}\, value', colCnt), \cdot (valuePart2, valuePart1,' .'),$$

$$\cdot (lit \downarrow value \downarrow value, valuePart2, rowCnt),$$

$$\cdot (valueLitPart,' "', LIT \downarrow value \downarrow terminal),$$

$$\#\big(colCnt, idCnt, +(rowCnt, rowCnt, 1)\big),$$

$$\cdot (lit \downarrow value \downarrow valuelit, valueLitPart, '"'),$$

$$\#(colCnt, idCnt, := (colCnt, 1)), @(terminal, temp), \#(temp, eof, := (t_5, 0)),$$

$$\#(temp, oef, -(rowCnt, rowCnt, 1)), \#\big(temp, eof, := (t_6, 1)\big), \#(temp, oef, \rho())\rangle.$$

Правило s_6. В замінюється на «value∘ ' table:isRepresentedByLiteral ' ∘valuelit ∘'.' ∘eol ∘B».

Збільшується лічильник colCnt.

Змінній $valuePart_1$ присвоюється результат конкатенації « table:value » та лічильника colCnt.

Змінній $valuePart_2$ присвоюється результат конкатенації valuePart1 і крапки.

Літералу значення терміналу value присвоюється результат конкатенації $valuePart_2$ і лічильника rowCnt.

Змінній valueLitPart присвоюється результат конкатенації лапки та літералу значення терміналу вхідних даних.

Літералу значення терміналу valuelit присвоюється результат конкатенації значенняLitPart і лапки.

Якщо colCnt дорівнює idCnt, то збільшується лічильник rowCnt.

Якщо colCnt дорівнює idCnt то colCnt присвоюється 1.

Виконується операція читання @ наступного (для правила s_8 буде поточним) терміналу ланцюжка вхідних даних змінну temp.

Якщо temp є eof, то t_5 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_6 стає недоступним.

Якщо temp є eof, то t_6 присвоюється 1. Тому правило s_7 стає доступним.

Якщо temp eof, то rowCnt зменшують на одиницю, тому що наступного рядка немає.

Якщо temp є eof, виконується операція $\rho()$повернення до початку читання ланцюжка символів вхідних даних.

Таким чином, є два випадки поведінки конструктивної системи.

Послідовне виконання правила s_6 для всіх значень таблиці

t_0 =1, t_1 =0, t_2 =0, t_3 =0, t_4 =0, t_5 =1, t_6 =0, t_7 =0, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 0, t_{12} = 0

або перехід до правила s_7 якщо досягнуто символу eof.

t_0 =1, t_1 =0, t_2 =0, t_3 =0, t_4 =0, t_5 =0, t_6 =1, t_7 =0, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 0, t_{12} = 0

$$s_7 = \langle B \ {}_{t_6}\!\rightarrow B\rangle,$$

$$g_7 = \langle @(terminal) + (k, k, 1), \#(k, colCnt, t_6, 0), \#(k, colCnt, t_7, 1)\rangle.$$

Правило s_7. Заміна не виконується. Правило використовується для перестановки каретки із заголовного кортежу на кортеж значень.

t_0 =1, t_1 =0, t_2 =0, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =1, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 0, t_{12} = 0

$$s_8 = \langle B \ {}_{t_7}\!\rightarrow tuple \circ {}_{table:hasElement}{}^{value} element \circ value1 \circ$$

$${}_{value} element \circ eol \circ B\rangle.$$

$$g_8 = \langle +(colCnt1, colCnt1,1),,$$

$$\#(AND(\big(colCnt1, colCnt), (rowCnt1, rowCnt)\big), := (t_7, 0)),$$

$$\#(AND\big((colCnt1, colCnt), (rowCnt1, rowCnt)\big), := (t_8, 1)),$$

$$\#(AND((colCnt1, colCnt), (rowCnt1, rowCnt)), \rho()),$$

$$\cdot (value1Part1,' table: value', colCnt1),$$

$\cdot (value1Part2, value1Part1,'.'),$

$\cdot (lit \lrcorner value \lrcorner value1, value1Part2, rowCnt1),$

$\cdot (lit \lrcorner value \lrcorner tuple,' table{:}tuple', rowCnt1),$

$\#(colCnt1, colCnt, +(rowCnt1, rowCnt1,1)),$

$\#(colCnt1, colCnt, \coloneqq (colCnt1,1))\rangle.$

Правило s_8. В замінюється на «tuple ∘' table:hasElement ' ∘VALUE1 ∘'.' ∘eol ∘B».

Збільшується лічильник $colCnt_1$.

Якщо лічильник $colCnt_1$ дорівнює лічильнику colCnt I лічильник $rowCnt_1$ дорівнює лічильнику rowCnt, то то t_7 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_8 стає недоступним.

Якщо лічильник $colCnt_1$ дорівнює лічильнику colCnt I лічильник $rowCnt_1$ дорівнює лічильнику rowCnt, то t_8 присвоюється 1. Тому правило s_9 стає доступним.

Якщо лічильник $colCnt_1$ дорівнює лічильнику colCnt I лічильник $rowCnt_1$ дорівнює лічильнику rowCnt, то виконується операція ρповернення каретки до початку ланцюжка вхідних даних.

Змінній value 1 $Part_1$ присвоюється результат конкатенації «table:value» та лічильника $colCnt_1$. Змінній value 1 $Part_2$ присвоюється результат конкатенації value1 Part1 і крапки. Літералу значення терміналу value1 присвоюється результат конкатенації value1 Part2 та лічильника rowCnt1.

Літералу значення терміналу tuple присвоюється результат конкатенації «table:tuple» та лічильника rowCnt1. Якщо лічильник $colCnt_1$ дорівнює лічильнику colCnt, то лічильник rowCnt1 збільшується.Якщо лічильник $colCnt_1$ дорівнює лічильнику colCnt то $colCnt_1$ присвоюється 1.

Таким чином, є два випадки поведінки конструктивної системи.

Послідовне виконання правила s_8 для всіх значень таблиці

$t_0 = 1, t_1 = 1, t_2 = 0, t_3 = 0, t_4 = 0, t_5 = 0, t_6 = 0, t_7 = 0, t_8 = 0, t_9 = 0, t_{10} = 0, t_{11} = 0, t_{12} = 0$

або перехід до правила s_9 якщо досягнуто символу eof.

$t_0 = 1, t_1 = 1, t_2 = 0, t_3 = 0, t_4 = 0, t_5 = 0, t_6 = 0, t_7 = 0, t_8 = 0, t_9 = 0, t_{10} = 0, t_{11} = 0, t_{12} = 0$

$$s_9 = \langle B \ {}_{t_8}\!\rightarrow id3 \circ {}_{table:naming}\!value\, element \circ value2 \circ '.' \circ eol \circ B \rangle,$$

$$g_9 = \langle +(colCnt2, colCnt2, 1),$$

$$\#(AND((colCnt2, colCnt), (rowCnt2, rowCnt)), t_8, 0),$$

$$\#(AND((colCnt2, colCnt), (rowCnt2, rowCnt)), t_9, 1),$$

$$\cdot (value2Part1, ' table: value', colCnt2), \cdot (value2Part2, value2Part1, ' . '),$$

$$\cdot (lit \lrcorner value \lrcorner value2, value2Part2, rowCnt2),$$

$$\cdot (lit \lrcorner value \lrcorner id3, ' table: identifier', colCnt2),$$

$$\#(colCnt2, colCnt, +(rowCnt2, rowCnt2, 1)),$$

$$\#(colCnt2, colCnt, := (colCnt2, 0))\rangle.$$

Правило s_9. В замінюється на «id3 ∘' table:naming ' ∘value2 ∘'.' ∘eol ∘B».

Збільшується лічильник $colCnt_2$.

Якщо лічильник $colCnt_2$ дорівнює лічильнику colCnt I лічильник $rowCnt_2$ дорівнює лічильнику rowCnt, то t_8 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_9 стає недоступним.

Якщо лічильник $colCnt_2$ дорівнює лічильнику colCnt I лічильник $rowCnt_2$ дорівнює лічильнику rowCnt, то t_9 присвоюється 1. Тому правило s_{10} стає доступним.

Змінній value 2 $Part_1$ присвоюється результат конкатенації «table:value» та лічильника $colCnt_2$.

Змінній value 2 $Part_2$ присвоюється результат конкатенації value 2 $Part_1$ і крапки.

Літералу значення терміналу value 2 присвоюється результат конкатенації value 2 $Part_2$ та лічильника $rowCnt_2$.

Літералу значення терміналу id 3 присвоюється результат конкатенації «table:tuple» та лічильника $rowCnt_2$. Якщо лічильник $colCnt_2$ дорівнює лічильнику colCnt, то лічильник $rowCnt_2$ збільшується. Якщо лічильник $colCnt_2$ дорівнює лічильнику colCnt, то $colCnt_2$ присвоюється 0.

Таким чином, є два випадки поведінки конструктивної системи.

Послідовне виконання правила s_9 для всіх значень таблиці

t_0 =1, t_1 =0, t_2 =0, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =0, t_8 =1, t_9 =0, t_{10} = 0, t_{11} = 0, t_{12} = 0

або перехід до правила s_{10} якщо досягнуто символу eof.

$t_0=1, t_1=0, t_2=0, t_3=0, t_4=0, t_5=0, t_6=0, t_7=0, t_8=0, t_9=1, t_{10}=0, t_{11}=0, t_{12}=0$

$$s_{10} = \langle B\ {}_{t_9}\!\rightarrow\ {}_{table:table}value\ element \circ {}_{table:hasPart}value\ element \circ tuple \circ$$

$${}_{.}value\ element \circ eol \circ$$

$$tuple \circ {}_{a}value\ element \circ {}_{table:tuple}value\ element \circ {}_{.}value\ element \circ eol \circ B\rangle.$$

$$g_{10} = \langle +(n,n,1), \cdot(lit \downarrow value \downarrow value3; table: tuple, n),$$

$$\#(n, rowCnt, t_9, 0), \#(n, rowCnt, t_{10}, 1)\rangle.$$

Правило s_{10}. В замінюється на « table:table ∘' table:hasPart ∘tuple ∘'.' ∘eol ∘B».

Збільшується лічильник n.

Літералу значення терміналу tuple присвоюється результат конкатенації «table:tuple» і n.

Якщо n = rowCnt, то $t_9=0$.

Якщо n = rowCnt, то $t_{10}=1$.

Таким чином, є два випадки поведінки конструктивної системи.

Послідовне виконання правила s_{10} для всіх значень таблиці

$t_0=1, t_1=0, t_2=0, t_3=0, t_4=0, t_5=0, t_6=0, t_7=0, t_8=0, t_9=0, t_{10}=1, t_{11}=0, t_{12}=0$

або перехід до правила s_{11} якщо n = rowCnt.

$t_0=1, t_1=0, t_2=0, t_3=0, t_4=0, t_5=0, t_6=0, t_7=0, t_8=0, t_9=0, t_{10}=1, t_{11}=0, t_{12}=0$

$$s_{11} = \langle B\ {}_{t_{10}}\!\rightarrow value3 \circ reation \circ value4 \circ '.' \circ eol \circ B\rangle,$$

$$g_{11} = \langle \cdot(value3Part1,' table: value', colCnt3),$$

$$\cdot(value3Part2, value2Part1,' .'),$$

$$\cdot(lit \downarrow value \downarrow value3, value3Part2, rowCnt3),$$

$$@(lit \downarrow value \downarrow relation), +(colCnt3, colCnt3, 1),$$

$$\cdot(value2Part1,' table: value', colCnt3), \cdot(value4Part2, value2Part1,' .'),$$

$$\cdot(lit \downarrow value \downarrow value4, value4Part2, rowCnt3),$$

$$+(rowCnt3, rowCnt3, 1), := (t_{11}, 1), := (t_{10}, 0),$$

$$\#(rowCnt3, rowCnt, t_{11}, 0),, \#(rowCnt3, rowCnt, t_{10}, 1).$$

$$\#(rowCnt3, rowCnt, t_{11}, 0),,\#(rowCnt3, rowCnt, t_{10}, 0)$$

$\#(AND((colCnt3, colCnt), (rowCnt3, rowCnt)), t_{12}, 1)\rangle$,

Правило s_{11}. Генерація трійок з порядком значень таблиці виконується з допомогою двох правил, щоб у першому правилі зовнішнім виконавцем вводилося відношення порядку, тоді як у другому генерувалися трійки всім рядків стовпця крім першої.

В замінюється на "value3 ∘relation ∘value4 ∘'."∘ eol ∘B».

Змінній value3Part$_1$ присвоюється результат конкатенації « table:value » та лічильника colCnt$_3$.

Змінній value 3 Part$_2$ присвоюється результат конкатенації value 3 Part1 і крапки.

Літералу значення терміналу value3 присвоюється результат конкатенації value3 Part2 і лічильника rowCnt3.

Зовнішнім виконавцем вводиться відношення порядку до літералу значення терміналу relation.

Збільшується лічильник colCnt$_3$.

Змінній value 4 Part$_1$ присвоюється результат конкатенації « table:value» та лічильника colCnt3.

Змінній value 4 Part2 присвоюється результат конкатенації value4 Part1 і крапки.

Літералу значення терміналу VALUE 4 присвоюється результат конкатенації value 4 Part2 та лічильника rowCnt3.

t_{11} присвоюється 1. Тому правило s_{12} стає доступним.

t_{10} присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_{11} стає недоступним.

Якщо лічильник rowCnt$_3$ дорівнює лічильнику rowCnt, то t_{11} присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_{13} стає недоступним.

Якщо лічильник rowCnt$_3$ дорівнює лічильнику rowCnt, то t_{10} присвоюється 1. Тому правило s_{12} стає доступним.

Якщо лічильник colCnt$_3$ дорівнює лічильнику colCnt I лічильник rowCnt$_3$ дорівнює лічильнику rowCnt, то t_{10} присвоюється 0.

Якщо лічильник $colCnt_3$ дорівнює лічильнику colCnt I лічильник $rowCnt_3$ дорівнює лічильнику rowCnt, то t_{12} присвоюється 1.

Таким чином, є три випадки поведінки конструктивної системи.

Перехід до правила s_{11} та формування трійок для всіх значень таблиці в цьому стовпці

t_0 =1, t_1 =0, t_2 =0, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =0, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 1, t_{12} = 0

або виконання s_{10} якщо у таблиці один рядок.

t_0 =1, t_1 =0, t_2 =0, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =0, t_8 =0, t_9 =0, t_{10} = 1, t_{11} = 0, t_{12} = 0

або завершення виконання конструктора

t_0 =1, t_1 =0, t_2 =0, t_3 =0, t_4 =0, t_5 =0, t_6 =0, t_7 =0, t_8 =0, t_9 =0, t_{10} = 0, t_{11} = 0, t_{12} = 1

$s_{12} = \langle B \ _{t_{11}}\rightarrow value3 \circ relation \circ value4 \circ'.' \circ eol \ \circ B\rangle,$

$g_{12} = \langle \cdot (value3Part1,' table: value', colCnt3),$

$\cdot (value3Part2, value2Part1,' .')$б

$\cdot (lit \downarrow value \downarrow value3, value3Part2, rowCnt3),$

$@(lit \downarrow value \downarrow relation), +(colCnt3, colCnt3,1),$

$\cdot (value4Part1,' table: value', colCnt3), \cdot (value4Part2, value4Part1,' .'),$

$\cdot (lit \downarrow value \downarrow value4, value4Part2, rowCnt3), \#(rowCnt3, rowCnt, t_{11}, 0),$

$\#(AND((colCnt3, colCnt), (rowCnt3, rowCnt)), t_{12}, 1),$

$\#(rowCnt3, rowCnt, t_{11}, 0), \#(rowCnt3, rowCnt, t_{10}, 1)\rangle.$

Правило s_{12}. В замінюється на "value3 ∘relation ∘value4 ∘'." ∘eol ∘B».

Змінній value 3 Part1 присвоюється результат конкатенації «table:value» та лічильника $colCnt_3$.

Змінній value 3 Part2 присвоюється результат конкатенації value3Part1 і крапки.

Нетермінал value3 присвоюється результат конкатенації value 3Part2 і лічильника rowCnt3.

Збільшується лічильник $colCnt_3$.

Змінній value 4 $Part_1$ присвоюється результат конкатенації «table:value» та лічильника $colCnt_3$.

Змінній value 4 $Part_2$ присвоюється результат конкатенації value4Part1 і крапки.

Нетермінал value4 присвоюється результат конкатенації value4Part2 і лічильника rowCnt3.

Якщо лічильник $rowCnt_2$ дорівнює лічильнику rowCnt, то t_{11} присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_{13} стає недоступним.

Якщо лічильник $colCnt_3$ дорівнює лічильнику colCnt I лічильник $rowCnt_3$ дорівнює лічильнику rowCnt, то t_{12} присвоюється 1

Якщо лічильник $rowCnt_2$ дорівнює лічильнику rowCnt, то t_{11} присвоюється 0.

Якщо лічильник $rowCnt_2$ дорівнює лічильнику rowCnt, то t_{10} присвоюється 1. Тому правило s_{12} стає доступним.

Таким чином, є два випадки поведінки конструктивної системи.

Послідовне виконання правила s_{13} для всіх значень таблиці

$t_0 = 1, t_1 = 0, t_2 = 0, t_3 = 0, t_4 = 0, t_5 = 0, t_6 = 0, t_7 = 0, t_8 = 0, t_9 = 0, t_{10} = 0, t_{11} = 1, t_{12} = 0$

або перехід до правила s_{12} якщо досягнуто символ лічильник $rowCnt_2$ дорівнює лічильнику rowCnt.

$t_0 = 1, t_1 = 0, t_2 = 0, t_3 = 0, t_4 = 0, t_5 = 0, t_6 = 0, t_7 = 0, t_8 = 0, t_9 = 0, t_{10} = 1, t_{11} = 0, t_{12} = 0$

$$s_{11} = \langle B_{\ t_{10}} \rightarrow \varepsilon \rangle,$$

$$g_{11} = \langle \varepsilon \rangle.$$

А.7 Перетворюючий конструктор зі словника на правила онтології у структуру таблиці ДУ-61 з ІРП

Згідно з розробленою моделлю [179], таблиця ДУ-61 – це така таблиця, яка містить заголовковий кортеж таблиці ДУ-61. Заголовний кортеж таблиці ДУ-61 – це такий кортеж, який (для усіченого варіанта) містить ідентифікатори місця та швидкості. Ідентифікатор місця – це такий ідентифікатор, який представлений літералом «місце».

Мета конструктора – формування ієрархії схеми онтології з використанням логічних визначень.

Вводиться: операція усічення π(a,b,c). Надає «a» залишок від усічення «b» на «c».

Семантика Λ_{tabl} включає поняття: header1var, header2var – змінні для заголовків таблиці, temp – змінна зберігання прочитаних символів ланцюжка вхідних даних. header1LitPart1, header1LitPart2, header2LitPart1, header2 LitPart2 – частини значень назви стовпця таблиці, а також поняття онтологій конструкторів, що породжують.

Початкові умови: $t_0 = 1$, $t_1 = 0$, $t_2 = 0$, $t_3 = 0$

Нетермінали : A, B.

Початковий нетермінал – A.

Термінали: eol, eof, value, tabletype – вид таблиці, table – таблиця, property2 – відношення для зв'язку таблиці та кортежу, tableheader – заголовний кортеж, tuple – кортеж, property1 – відношення для зв'язку кортежу та елементів таблиці, header1, header – екземпляри, що представляють назви стовпців таблиці, property – відношення для зв'язку елементів таблиці та їх значень, header1lit, header2lit – значення назви стовпця таблиці

Вхідні дані: метадані форми таблиці попередження ДУ-61 з ІРП (Таблиця Б. 2).

Виконавець – внутрішній – Protégé та зовнішній чи програмний виконавець для імпорту.

Обмеження:

1) всі класи та відношення належать словнику;

2) задаються атрибутами правил підстановки.

Умова завершення – всі елементи таблиці метаданих структури попередження ДУ-61 перетворені на правила онтології.

Вихідні дані представлені в Рисунку Б. 4.

Правила підстановки:

$$s_1 = \langle A \ {}_{t_0}\!\rightarrow {}_{@prefix}\!value\, element \circ {}_{owl:}\!value\, element \circ$$

$$<http://www.w3.org/2002/07/owl\#>\!value\, element \circ {}_{value}\!element \circ eol \circ$$

$$_{@prefix}{}^{value}element \circ {}_{rdf:}{}^{value}element \circ$$

$$<http://www.w3org/1999/02/22-rdf-syntax-ns\#>.{}^{value}element \circ$$

$$_{.}{}^{value}element \circ eol \circ {}_{@prefix}{}^{value}element \circ {}_{table:}{}^{value}element \circ$$

$$<http://example.org/table\#>{}^{value}element \circ$$

$$_{.}{}^{value}element \circ eol \circ {}_{<http://example.org/table>}{}^{value}element \circ$$

$$_{rdf:type}{}^{value}element \circ {}_{owl:ontology}{}^{value}element \circ {}_{.}{}^{value}element \circ$$

$$eol \circ tabletype \circ {}_{owl:equivalentClass}{}^{value}element \circ$$

$$_{[}{}^{value}element \circ {}_{owl:intersectionOf}{}^{value}element \circ$$

$$_{(}{}^{value}element \circ table \circ {}_{[}{}^{value}element \circ$$

$$_{rdf:type}{}^{value}element \circ {}_{owl:Restriction}{}^{value}element \circ$$

$$_{;}{}^{value}element \circ {}_{owl:onProperty}{}^{value}element \circ$$

$$property2 \circ {}_{;}{}^{value}element \circ$$

$$_{owl:someValuesFrom}{}^{value}element \circ tableheader \circ$$

$$_{]}{}^{value}element \circ {}_{;}{}^{value}element \circ {}_{)}{}^{value}element \circ$$

$$_{rdf:type}{}^{value}element \circ {}_{owl:Class}{}^{value}element \circ$$

$$_{]}{}^{value}element \circ {}_{.}{}^{value}element \circ eol \circ B\rangle,$$

$$g_1 = \langle := (t_0, 0), := (t_1, 1), @(terminal),$$

$$@(terminal), @(terminal), @(terminal, temp),$$

$$\cdot (lit \downarrow value \downarrow tabletype, ' table:' , lit \downarrow value \downarrow temp).$$

$$:= (lit \downarrow value \downarrow table, ' table: table'),$$

$$:= (lit \downarrow value \downarrow property2, ' table: hasPart'), \widetilde{@}(terminal, temp),$$

$\#(lit \downarrow value \downarrow temp, \varepsilon, t_7, 1),$

$\cdot (lit \downarrow value \downarrow tableheader,' table:' , lit \downarrow value \downarrow temp)\rangle.$

```
@prefix : <http://www.semanticweb.org/larisk0/ontologies/2022/6/untitled-ontology-556#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@base <http://example.org/table> .

<http://example.org/table> rdf:type owl:Ontology .

:hasElement rdf:type owl:ObjectProperty .

:hasPart rdf:type owl:ObjectProperty .


:du61Table          owl:equivalentClass [ owl:intersectionOf ( :table
                                                              [ rdf:type owl:Restriction ;
                                                                owl:onProperty :hasPart ;
                                                                owl:someValuesFrom :du61header
                                                              ]
                                                            ) ;
                                          rdf:type owl:Class
                                        ] .


:du61header           owl:equivalentClass [ owl:intersectionOf ( :tuple
                                                                 [ rdf:type owl:Restriction ;
                                                                   owl:onProperty :hasElement ;
                                                                   owl:someValuesFrom :placeIdentifier
                                                                 ]
                                                                 [ rdf:type owl:Restriction ;
                                                                   owl:onProperty :hasElement ;
                                                                   owl:someValuesFrom :speedIdentifier
                                                                 ]
                                                               ) ;
                                            rdf:type owl:Class
                                          ] .
```

Рисунок Б. 4 Реалізація «перетворюючого конструктора зі словника до правил онтології для структури таблиці ДУ-61 з ІДП» на прикладі правил онтології джерела структури таблиці ДУ-60 ІДП

Правило s_1. А замінюється на префікси онтології, визначення онтології та логічне визначення таблиці бланка ДУ-61 з терміналами table, property 2, tabletype, tableheader та нетерміналом B. У результаті виходить логічне визначення таблиці ДУ-61. t_0 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_1 стає недоступним. t_1 присвоюється 1. Тому правило s_2 стає доступним. Для переміщення каретки з рядка заголовків на рядок із значеннями тричі виконується операція читання @. Виконується операція читання @ терміналу ланцюжка таблиці вхідних даних змінну temp. Літералу значення терміналу tabletype присвоюється результат конкатенації рядка «table:» та літерала значення temp. Літералу значення терміналу table присвоюється строка

«table:table». Літералу значення терміналу property 2 присвоюється строка «table:hasPart». Виконується операція читання @ терміналу ланцюжка таблиці вхідних даних змінну temp. Якщо літерал значення терміналу дорівнює ε, то t_4 присвоюється 1. Тому правило s_5 стає доступним. Необхідно читати наступний рядок у цьому стовпці. Літералу значення терміналу tableheader присвоюється результат конкатенації рядка «table:» та літерала значення прочитаного наступного терміналу ланцюжка таблиці вхідних даних. t_0 =0, t_1 = 1, t_2 =0, t_3= 0.

$$s_2 = \langle B \ {}_{t_1}\!\rightarrow tableheader \circ$$

$${}_{owl:equivalentClass}{}^{value}element \circ \ {}_{[}{}^{value}element \circ \ {}_{owl:intersectionOf}{}^{value}element \circ$$

$${}_{(}{}^{value}element \circ tuple \circ \ {}_{[}{}^{value}element \circ$$

$${}_{rdf:type}{}^{value}element \circ \ {}_{owl:Restriction}{}^{value}element \circ$$

$${}_{;}{}^{value}element \circ \ {}_{owl:onProperty}{}^{value}element \circ property1 \circ \ {}_{;}{}^{value}element \circ$$

$${}_{owl:someValuesFrom}{}^{value}element \circ header1 \circ \ {}_{]}{}^{value}element \circ$$

$${}_{rdf:type}{}^{value}element \circ \ {}_{owl:Restriction}{}^{value}element \circ$$

$${}_{;}{}^{value}element \circ \ {}_{owl:onProperty}{}^{value}element \circ property1 \circ \ {}_{;}{}^{value}element \circ$$

$${}_{owl:someValuesFrom}{}^{value}element \circ header2 \circ \ {}_{]}{}^{value}element \circ$$

$$\circ \ {}_{]}{}^{value}element \circ \ {}_{;}{}^{value}element \circ \ {}_{)}{}^{value}element \circ$$

$${}_{rdf:type}{}^{value}element \circ \ {}_{owl:Class}{}^{value}element \circ \ {}_{]}{}^{value}element \circ \ {}_{.}{}^{value}element \circ$$

$$eol \circ B\rangle,$$

$$g_2 = \langle \coloneqq (t_1, 0), \coloneqq (t_2, 1),$$

$$\cdot (lit \lrcorner value \lrcorner tableheader,' table:' , lit \lrcorner value \lrcorner temp)$$

$$\coloneqq (lit \lrcorner value \lrcorner tuple,' table: tuple'),$$

$$\coloneqq (lit \lrcorner value \lrcorner property1,' table: hasElement'),$$

$\widetilde{@}(terminal, temp)$, $\#(lit \lrcorner value \lrcorner temp, \varepsilon, t_7, 1)$,

$\coloneqq (header1Var, lit \lrcorner value \lrcorner temp)$,

$\cdot (lit \lrcorner value \lrcorner header1,' table:' , lit \lrcorner value \lrcorner temp)$,

$\widetilde{@}(terminal, temp)$, $\#(lit \lrcorner value \lrcorner temp, \varepsilon, t_7, 1)$,

$\coloneqq (header2Var, lit \lrcorner value \lrcorner temp)$,

$\cdot (lit \lrcorner value \lrcorner header2,' table:', lit \lrcorner value \lrcorner temp)\rangle$.

Правило s_2. У замінюється на логічне визначення заголовного кортежу визначення таблиці бланка ДУ-61 з терміналами tuple, property 1 header 1 header 2 і нетермінала B. В результаті генерується логічне визначення заголовного кортежу таблиці попередження про обмеження швидкості.

t_1 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_2 стає недоступним.

t_2 присвоюється 1. Тому правило s_3 стає доступним.

Літералу значення терміналу tuple присвоюється строка "table:tuple".

Літералу значення терміналу property1 присвоюється строка «table:hasElement».

Виконується операція читання $\widetilde{@}()$терміналу ланцюжка таблиці вхідних даних змінну temp.

Якщо літерал значення temp дорівнює ε, то t_4 присвоюється 1. Тому правило s_5 стає доступним. Необхідно читати наступний рядок у цьому стовпці.

Змінній header 1 var присвоюється літерал значення temp.

Літералу значення терміналу header1 присвоюється результат конкатенації строки «table:» та літерала значення temp.

Виконується операція читання$\widetilde{@}()$ наступного терміналу ланцюжка таблиці вхідних даних у змінну temp.

Якщо літерал значення temp дорівнює ε, то t_4 присвоюється 1. Тому правило s_5 стає доступним. Необхідно читати наступний рядок у цьому стовпці. Змінній header2var присвоюється літерал значення temp.

Літералу значення терміналу header 2 присвоюється результат конкатенації рядка « table:» та літерала значення temp.

$t_0 = 0$, $t_1 = 0$, $t_2 = 1$, $t_3 = 0$.

$$s_3 = \langle B \ _{t_2}\rightarrow header1 \circ$$

$$_{owl:equivalentClass}value\,element \circ \ _{[}value\,element \circ \ _{owl:intersectionOf}value\,element \circ$$

$$\ _{(}value\,element \circ \ _{table:identifier}value\,element \circ \ _{[}value\,element \circ$$

$$_{rdf:type}value\,element \circ \ _{owl:Restriction}value\,element \circ$$

$$\ _{;}value\,element \circ \ _{owl:onProperty}value\,element \circ property \circ \ _{;}value\,element \circ$$

$$_{owl:someValuesFrom}value\,element \circ \ _{[}value\,element \circ \ _{rdf:type}value\,element \circ$$

$$_{rdfs:datatype}value\,element \circ \ _{;}value\,element \circ \ _{owl:oneof}value\,element \circ$$

$$\ _{[}value\,element \circ \ _{rdf:type}value\,element \circ \ _{rdf:list}value\,element \circ$$

$$\ _{;}value\,element \circ \ _{rdf:first}value\,element \circ header1lit \circ$$

$$\ _{;}value\,element \circ \ _{rdf:rest}value\,element \circ \ _{rdf:nil}value\,element \circ$$

$$\ _{]}value\,element \circ \ _{]}value\,element \circ \ _{]}value\,element \circ$$

$$\ _{)}value\,element \circ \ _{rdf:type}value\,element \circ \ _{owl:Class}value\,element \circ$$

$$\ _{]}value\,element \circ \ _{.}value\,element \circ eol \circ B\rangle,$$

$$g_3 = \langle \coloneqq (t_2, 0), \coloneqq (t_3, 1), \cdot (lit \downarrow value \downarrow header1, ' table:', header1var),$$

$$\coloneqq (lit \downarrow value \downarrow property, ' table: isRepresentedByLiteral'),$$

$$\pi(header1LitPart1, header1var, 'identifier'),$$

$$\cdot (header1LitPart2, '"', header1LitPart1).$$

$\cdot (lit \downarrow value \downarrow header1lit, header1LitPart2, '"')\rangle$.

Правило s_3. В замінюється на логічне визначення заголовка з терміналами property header 1 header 1 lit і терміналом B. В результаті генерується логічне визначення заголовка «place».

$t_0 = 0$, $t_1 = 0$, $t_2 = 0$, $t_3 = 1$.

$$s_4 = \langle B \ {}_{t_3}\!\rightarrow header2 \circ$$

$${}_{owl:equivalentClass}value\,element \circ \ {}_{[}value\,element \circ \ {}_{owl:intersectionOf}value\,element \circ$$

$${}_{(}value\,element \circ \ {}_{table:identifier}value\,element \circ \ {}_{[}value\,element \circ$$

$${}_{rdf:type}value\,element \circ \ {}_{owl:Restriction}value\,element \circ$$

$${}_{;}value\,element \circ \ {}_{owl:onProperty}value\,element \circ property \circ \ {}_{;}value\,element \circ$$

$${}_{owl:someValuesFrom}value\,element \circ \ {}_{[}value\,element \circ \ {}_{rdf:type}value\,element \circ$$

$${}_{rdfs:datatype}value\,element \circ \ {}_{;}value\,element \circ \ {}_{owl:oneof}value\,element \circ$$

$${}_{[}value\,element \circ \ {}_{rdf:type}value\,element \circ \ {}_{rdf:list}value\,element \circ$$

$${}_{;}value\,element \circ \ {}_{rdf:first}value\,element \circ header2lit \circ$$

$${}_{;}value\,element \circ \ {}_{rdf:rest}value\,element \circ \ {}_{rdf:nil}value\,element \circ$$

$${}_{]}value\,element \circ \ {}_{]}value\,element \circ \ {}_{]}value\,element \circ$$

$${}_{)}value\,element \circ \ {}_{rdf:type}value\,element \circ \ {}_{owl:Class}value\,element \circ$$

$${}_{]}value\,element \circ \ {}_{.}value\,element \circ eol \circ B\rangle,$$

$$g_4 = \langle \coloneqq (t_3, 0), \cdot (lit \downarrow value \downarrow header2, ' table:', header2var)$$

$$\pi(header2LitPart1, header2var, 'identifier'),$$

$$\cdot (hader2LitPart2, '"', header2LitPart1),$$

$$\cdot (lit \downarrow value \downarrow header2lit, header1LitPart2, '"')\rangle.$$

Правило s_4. В замінюється на логічне визначення заголовка з нетерміналами header 2, header 2 lit. В результаті генерується логічне визначення заголовка «speed».

t_3 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_3 стає недоступним.

Літералу значення терміналу header2 присвоюється результат конкатенації строки «table:» та header2var.

Змінній header 2 LitPart1 присвоюється результат усічення header2var на «ідентифікатор».

Змінній header 2 LitPart_2 присвоюється результат конкатенації «"» (лапки) і header2LitPart1.

Літералу значення терміналу header2lit присвоюється результат конкатенації header2LitPart2 та «"» (лапки).

t_0 =0, t_1 =0, t_2 =0, t_3 =0.

Конструктивна операція.

$$s_5 = \langle \tilde{@}(terminal) \ {}_{t_4}\!\rightarrow @(terminal) \circ @(terminal) \circ \tilde{@}(terminal) \rangle,$$

$$g_5 = \langle \#(\neq (lit \downarrow value \downarrow terminal, \varepsilon), t_7, 0) \rangle.$$

Правило s_5. Конструктивна операція використовується для читання терміналів ланцюжка таблиці вхідних даних у певному стовпці до тих пір, поки не зустрінеться термінал з літералом значення не рівним ε. Правило виконується рекурсивно доки не зустріне термінал з літералом значення не рівним ε. Необхідно «пропустити» два термінали та прочитати термінал у цьому ж стовпці.

А.8 Перетворюючий конструктор з екземплярів та правил в онтологію джерела ДУ-61

При імпорті екземплярів формату RDF у схему онтології екземпляри класифікуються як owl:NamedIndividual тому що вони є екземплярами класів OWL (наприклад table:value1.1 a table:значення) і зв'язуються з іншими

екземплярами та літералами за допомогою owl:ObjectProperty та owl:datatypeProperty. Вирази « rdf:type » та «a» еквівалентні.

При імпорті в новий файл схеми онтології та екземплярів необхідно додати тип owl:NamedIndividual кожному екземпляру та додати вираз типу <http://www.semanticweb.org/larisk0/ontologies/2022/1/model> rdf:type owl:Ontology, owl:imports для схеми онтології. Примірники зливаються з порожнім файлом.

Таким чином необхідно спочатку прочитати файл екземплярів і додати кожному тип owl:NamedIndividual, а потім злити схему та перетворений файл екземплярів.

Мета конструктора – перетворення ресурсів RDF на іменовані екземпляри OWL.

Вводиться операція імпорту β(a,b) – імпортує файл «a» файл «b».

Виконавці – зовнішній чи програмний виконавець для імпорту, Protégé.

Обмеження:

1) всі класи та відношення належать словнику;

2) задаються атрибутами правил підстановки.

Семантика Λ_{tabl}включає: temp – змінна для зберігання елементів онтології – лічильник елементів трійки, а також поняття онтологій конструкторів, що породжують.

Умова завершення – всі ресурси RDF перетворені на екземпляри онтології, виконано імпорт схеми онтології.

Початкові умови: $t_0 = 1$, $t_1 = 0$, $t_2 = 0$, $t_3 = 0$, $t_4 = 0$.

Вхідні дані: файли екземплярів та схеми онтології.

Нетермінали : A, B,

Початковий нетермінал – A.

Термінали: element, value.

Вихідні дані: онтологія джерел попереджень про обмеження швидкості.

Правила підстановки:

$s_1 = \langle A\ {}_{t_0}\!\rightarrow B\rangle$,

$g_1 = \langle @(terminal, temp), +(k, k, 1), := (t_1, 1)\rangle$.

Правило s_1. А замінюється конструкція імпорту онтології та нетермінал B. Де «http://example.org/model» – адреса нової онтології, а «http://example.org/table» – адреса імпортованої.

Збільшується лічильник k.

Виконується операція @читання терміналу ланцюжка вхідних даних.

t_1 присвоюється 1. Тому правило s_2 стає доступним.

t_0 =1, t_1 =1, t_2 =0, t_3 =0, t_4 =0.

$s_2 = \langle B\ {}_{t_1}\!\rightarrow element \circ B\rangle$,

$g_2 = \langle +(k, k, 1), \#(k, 5, := (t_4, 1)), \#(k, 5, := (t_1, 0)),$
$:= (lit \downarrow value \downarrow element, lit \downarrow value \downarrow temp),$
$@(terminal, temp), \#(lit \downarrow value \downarrow temp, 'a', := (t_2, 1)),$
$\#(lit \downarrow value \downarrow temp, 'a', := (t_1, 0)),$
$\#(temp, eof, := (t_3, 1)), \#(temp, eof, := (t_1, 0))\rangle$.

Правило s_2. B замінюється на « ∘BCD ∘»

Збільшується лічильник k.

Якщо k дорівнює 4, t_4 присвоюється 1. Правило s_5 стає доступним.

Якщо k дорівнює 4, t_1 присвоюється 0. Правило s_2 стає недоступним.

Літералу значення терміналу element присвоюється літерал значення temp.

Виконується операція @читання наступного терміналу ланцюжка вхідних даних змінну temp.

Якщо літерал temp дорівнює «a» (rdf:type), то t_2 присвоюється 1. Тому правило s3 стає доступним.

Якщо літерал temp дорівнює «a» (rdf:type), то t_1 присвоюється 0. Тому правило s_2 стає недоступним.

Якщо temp дорівнює eof, то t_3 присвоюється 1. Тому правило s_4 стає доступним.

Якщо temp дорівнює eof, то t_1 присвоюється 0. Тому правило s_2 стає недоступним.

Таким чином, є три випадки поведінки конструктивної системи.

Переходять до правила s_5 якщо k = 4

t_0 =1, t_1 =0, t_2 =0, t_3 =0, t_4 =1.

перехід до правила s_3 якщо прочитаний термінал є виразм «a» (rdf:type)

t_0 =1, t_1 =0, t_2 =0, t_3 =1, t_4 =0

або повторне виконання правила s_2

t_0 =1, t_1 =1, t_2 =0, t_3 =0, t_4 =0

$s_3 = \langle B \ _{t_2}\rightarrow \ _{a}{}^{value}element \circ$

$_{owl:NanedIndiwidual}{}^{value}element \circ \ _{;}{}^{value}element \circ B\rangle,$

$g_3 = \langle @(terminal, temp), +(k, k, 1), := (t_1, 1), := (t_2, 0)\rangle.$

Правило s_3. В замінюється на 'a' ∘' owl:NamedIndividual ∘B».

Виконується операція @читання наступного терміналу ланцюжка вхідних даних змінну temp.

t_1 присвоюється 1. Тому правило s_2 стає доступним.

t_2 присвоюється 0. Конструктор має обмеження «якщо t = 0 правило підстановки стає недоступним». Тому правило s_3 стає недоступним.

t_0 =1, t_1 =0, t_2 =0, t_3 =0, t_4 =0

$s_4 = \langle B \ _{t_3}\rightarrow \varepsilon; \rangle,$

$g_4 = \langle \varepsilon \rangle.$

Правило s_4. Використовується для заміни нетерміналів на ε.

$s_5 = \langle B \ _{t_4}\rightarrow eol \ \circ B\rangle,$

$g_5 = \langle +(n, n, 1), \#(n, 3, := (t_5, 1)), @(terminal, temp), := (t_1, 1), := (k, 1), \#(n, 3, := (t_1, 0)), := (t_4, 0)\rangle.$

Правило s_5. В замінюється на eol ∘ B. _

Збільшується лічильник n.

Якщо n = 3 t_5 присвоюється 1.

Виконується операція @читання наступного терміналу ланцюжка вхідних даних змінну temp.

t_1 присвоюється 1. Тому правило s_2 стає доступним.

t_4 присвоюється 0.

k присвоюється 0

Якщо n = 3 t_1 присвоюється 0.

$$s_6 = \langle B \ {}_{t_5}\!\rightarrow {}_{<http://example.org/model>}\!^{value}element \circ$$

$${}_{rdf:type\ owl:Ontology}\!^{value}element \circ \ {}_{,value}element \circ$$

$${}_{owl:imports}\!^{value}element \circ {}_{<http://example.org/table>}\!^{value}element \circ eol \ \circ B\rangle,$$

$$g_6 = \langle := (t_1, 1), := (t_5, 0), \beta(model, table)\rangle.$$

Правило s_6. В замінюється на конструкцію імпорту.

t_1 присвоюється 1. Тому правило s_2 стає доступним.

t_5 присвоюється 0.

Виконується операція імпорту.

Додаток Б

Акти впровадження

railML.org

railML.org e. V.
Altplauen 19h, 01187 Dresden, Germany

Ukrainian State University of Science and Technology
Larysa Zhuchyi
Lazaryana St. 2
Dnipro 49010
Ukraine

Your reference
Your letter
Our reference
Your correspondent Vasco Paul Kolmorgen
Telephone +49 351 – 47 58 29 11
Mobile
E-mail coordination@railml.org

December 14th, 2022

Implementation Certificate

This is to certify that the results of the "Railway data integration and consistency checking by ontology" (Інтеграція та узгодження даних інформаційних систем залізничного транспорту онтологічними засобами) PhD dissertation research of M.Sc. Larysa Zhuchyi are used in railML.org as:

1) Part of a "railML® automated certification" project, where proposed data consistency checking procedures come into use;
2) Part of a "railML® 2 to railML® 3 data transformation" project, where proposed schema bridging mechanisms come into use;
3) Recommendations for prospects of the railML® ontology development.

The usage of the thesis results allows for developing description logic ontologies from the railway view for data integration and consistency checking to facilitate the enhancement of railway transport safety.

Sincerely,

M.Sc. Vasco Paul Kolmorgen
Governance Coordinator

Altplauen 19h • Dresden • Germany
railML.org e.V.
+49 351 47582911
Germany · Allemagne · Германия · ألمانيا

M.Sc. Christian Rahmig
Infrastructure Coordinator

"ЗАТВЕРДЖУЮ"
Перший проректор
Українського державного університету
науки і технологій

_____ д.т.н., проф. Анатолій РАДКЕВИЧ
"04" 05 2023 р.

АКТ
про використання результатів дисертації
"ІНТЕГРАЦІЯ ТА УЗГОДЖЕННЯ ДАНИХ ІНФОРМАЦІЙНИХ СИСТЕМ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ ОНТОЛОГІЧНИМИ ЗАСОБАМИ"
Жучий Лариси Ігорівни,
представленої на здобуття наукового ступеня доктора філософії спеціальності 122 «Комп'ютерні науки»,

Цей акт складений про те, що у навчальному процесі Дніпровського інституту інфраструктури і транспорту Українського державного університету науки і технологій при підготовці магістрів за спеціальністю 121 «Інженерія програмного забезпечення» використовуються наукові та практичні результати, що отримані в дисертації Жучий Л. І., при викладанні дисциплін «Інформаційні системи на залізничному транспорті» та «Інтернет-технології», а також аспірантів за спеціальністю 122 «Комп'ютерні науки» – «Ефективність інформаційних систем та комп'ютерних технологій».

Зав. кафедри
Комп'ютерні інформаційні технології
к.т.н., доцент _____ Вадим ГОРЯЧКІН

Декан факультету
Комп'ютерних технологій і систем
к.т.н., доцент _____ Володимир МАЛОВІЧКО