Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи підвищення ефективності пошуку
в базах знань великої розмірності

(тема)

Виконав:
студент II курсу, групи СПм-19-1
Бахмацький В.С.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: доц. Федорченко В.М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Бахмацькому Владиславу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи підвищення ефективності пошуку в базах знань великої розмірності

затверджена наказом по університету від " 30 " жовтня 2020 р. № 1486Ст

2. Термін подання студентом роботи до екзаменаційної комісії 14 грудня 2020 р.

3. Вхідні дані до роботи Публікації моделей представлення знань в базах даних

4. Перелік питань, що потрібно опрацювати в роботі

Актуальність продукційної моделі знань в інтелектуальних системах, принципи організації баз знань з використанням прецедентів, редукція дерева рішень шляхом усунення повторювання гілок реалізація методів прискорення пошуку рішень в системі технологічного обслуговування касира

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) слайди презентації 12

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Актуальність продукційній моделі знань в	03.11.20-09.11.20	
	інтелектуальних системах		
2	Модель актуальної системи на основі бази	10.11.20-17.11.20	
	знань великої розмірності		
3	Методи підвищення ефективності пошуку в	24.11.20-01.12.20	
	базах знань великої розмірності		
4	Застосування розроблених методів в	02.12.20-07.12.20	
	інтелектуальних системах великої розмірності		
5	Подання атестаційної роботи керівникові та	08.12.20-09.12.20	
	її попередній захист		
6	Подання атестаційної роботи на рецензування	10.12.20-11.12.20	

Дата видачі завдання 02 листопада 2020

Студент _____
(підпис)

Керівник роботи _____ доц. Федорченко В.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 72 с., 23 рис., 2 табл., 1 дод., 26 джерел.

СКБД, SQL, ЕКСПЕРТНА СИСТЕМА, ПРОТОКОЛ, SEMANTIC WEB RULE LANGUAGE.

Метою атестаційної роботи є підвищення ефективності процесів логічного пошуку в базах знань на продукционной моделі знань за рахунок використання більш швидких, ніж відомі алгоритми логічного виведення в базах знань великої розмірності. Дана мета досягається вирішенням наступних основних завдань:

1. Аналіз стану робіт в області IT, формування основних напрямів дослідження в частині синтезу більш швидких, ніж існуючі, методів логічного пошуку в базах знань великої розмірності.

2. Розробка концептуальної моделі інтелектуальної системи на продукційної моделі знань, яка враховує предметну неоднорідність простору пошуку та орієнтованої на використання методів швидкого логічного висновку у великих базах знань.

3. Дослідження і розробка загальносистемних методів прискорення логічного висновку в базах знань великої розмірності за рахунок скорочення структурної надмірності дерева рішень.

4. Розробка комплексу алгоритмічних методів прискорення логічного висновку в інтелектуальних системах на продукційній моделі знань.

5. Розробка методів редукування простору пошуку в базах знань великої розмірності на основі інформаційного підходу, що враховує структурні особливості баз знань великої розмірності.

ABSTRACT

Master’s thesis: 72 pages, 23 figures, 2 tables, 1 appendices, 26 sources.

DBMS, SQL, EXPERT SYSTEM, PROTOCOL, SEMANTIC WEB RULE LANGUAGE.

The major goal of this thesis is to increase the efficiency of logical search processes in knowledge bases on the production model of knowledge through the use of faster than known algorithms for logical inference in knowledge bases of large dimensions. This goal is achieved by solving the following main tasks:

1. Analysis of the state of work in the field of IT, the formation of the main directions of research in terms of synthesis of faster than existing methods of logical search in knowledge bases of large dimension.

2. Development of a conceptual model of an intelligent system on a production model of knowledge, which takes into account the subject heterogeneity of the search space and focused on the use of methods of rapid inference in large knowledge bases.

3. Research and development of system-wide methods of accelerating logical inference in knowledge bases of large dimension by reducing the structural redundancy of the decision tree.

4. Development of a set of algorithmic methods for accelerating logical inference in intelligent systems on the production model of knowledge.

5. Development of methods for reducing the search space in large knowledge bases on the basis of an information approach that takes into account the structural features of large knowledge bases.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЕС – експертна система

ІС – інтелектуальна система

ІСРЧ – інтелектуальна система реального часу

СУБД – система управління базами даних

ШІ – штучний інтелект

CBR – Case Based Reasoning (висновок на основі прецедентів)

CWA – Closed World Assumption

HTML – Hypertext Markup Language

HTTP – HyperText Transfer Protocol

IQ – Intelligence Quotinent

OWA – Open World Assumption

OWL – Web Ontology Language

RDF – Resource Description Framework

RuleML – Rule Markup Language

SQL – Structurized Query Language

SWRL – Semantic Web Rule Language

URI – Uniform Resource Identifier

URL – Uniform Resource Location

W3C – WorldWide Web Consortium

XML – eXtended Markup Language

ВСТУП

Актуальність проблеми. Впродовж останнього десятиріччя характеризується зростанням потреби суспільства в інформатизації та інтелектуалізації у всіх сферах, та в державному управлінні, охороні здоров'я, освіті, на транспорті, в промисловості та ін. областях. Дана тенденція усвідомлюється як науковим співтовариством, так і на урядовому рівні. Процес інтелектуалізації систем управління, підтримки прийняття рішень, робототехнічних та інформаційних систем супроводжується як істотним збільшенням обсягів баз знань, так і зростанням динамічності контенту за рахунок залучення в процеси пошуку рішень не тільки статичних, але і темпоральних знань.

Найбільш популярною моделлю знань є продукційна модель, на якій будується переважна більшість баз знань, про що свідчить той факт, що за даними Scopus кількість публікацій у науковій періодиці, присвячених продукційній моделі знань перевищує кількість статей по іншим моделям знань. Мають місце глобалізація знань для конкретних додатків і інтеграція з базами даних, і як приклад тут можна навести глобальну систему бронювання авіаквитків. Це дозволяє говорити про тенденції появи баз знань великої розмірності, для яких характерне падіння ефективності пошуку в міру збільшення обсягу й ускладнення структури.

Одночасно з ростом обсягів і ускладненням структури баз знань посилюються вимоги до глибини і широти пошуку при обмеженнях на час вилучення знань. Незважаючи на істотне зростання загальної продуктивності засобів обчислювальної техніки, апаратну підтримку вузьких обчислювальних операцій, можливості агрегування ресурсів в ГРІД-структурах, зростання складності в просторі пошуку в багато разів перевищує можливість реалізації пошуку за допомогою алгоритмів з експоненційної складністю. Цією обставиною зумовлено ведення

широкомасштабних досліджень в області прискорення логічного пошуку як за кордоном (Стенфордський університет, компанії IBM, HP Lab, Google, Yahoo, і ін.). На жаль, існуючі методи логічного висновку не в повній мірі враховують багатомірність баз даних і знань великої розмірності. Крім того, відомі реалізації зводяться до алгоритмічних методів прискорення пошуку, і навіть досягнене прискорення логічного висновку на 2-3 порядки не дозволяє усунути комбінаторну складність завдання пошуку рішень.

Таким чином, спостерігається протиріччя між зростаючими обсягами і розмірністю баз знань ІС з одного боку і обчислювальними можливостями програмно-апаратних засобів і алгоритмів пошуку з іншого боку, що робить проблему реалізації швидкого логічного пошуку в базах знань великої розмірності актуальною.

Розв'язанням наукової проблемою в магістерській роботі є розробка методів підвищення ефективності пошуку в базах знань великої розмірності.

Об'єкт дослідження – процес пошуку в базах знань великої розмірності у системах штучного інтелекту, що використовують великі бази знань, які побудовані на продукційній моделі.

Предмет дослідження – методи логічного пошуку в базах знань великої розмірності.

Мета дослідження – підвищення ефективності процесів логічного пошуку в базах знань на продукційній моделі знань за рахунок використання більш швидких, ніж відомі алгоритми логічного виведення в базах знань великої розмірності. Дана мета досягається вирішенням наступних основних завдань:

1. Аналіз стану робіт в області ІТ, формування основних напрямів дослідження в частині синтезу більш швидких, ніж існуючі, методів логічного пошуку в базах знань великої розмірності.

2. Розробка концептуальної моделі інтелектуальної системи на продукційній моделі знань, яка враховує предметну неоднорідність простору пошуку та орієнтованої на використання методів швидкого логічного

висновку у великих базах знань.

3. Дослідження і розробка загальносистемних методів прискорення логічного висновку в базах знань великої розмірності за рахунок скорочення структурної надмірності дерева рішень.

4. Розробка комплексу алгоритмічних методів прискорення логічного висновку в інтелектуальних системах на продукційній моделі знань.

5. Розробка методів редукування простору пошуку в базах знань великої розмірності на основі інформаційного підходу, що враховує структурні особливості баз знань великої розмірності.

6. Експериментальне дослідження розроблених методів і алгоритмів прискорення логічного висновку для підтвердження достовірності отриманих результатів.

Наукова новизна:

1. Концептуальна модель інтелектуальної системи великої розмірності на продукційній моделі знань, що відрізняється урахуванням предметної неоднорідності простору пошуку, що дозволяє сформулювати умови вибору складу методів прискорення логічного висновку для конкретних умов або додатків.

2. Загальносистемні методи підвищення швидкості пошуку в базах знань великої розмірності, що відрізняються процедурами попередньої загальної мінімізації надмірності дерева рішень, що дозволяє досягти поліноміальної складності логічного висновку.

1 АНАЛІЗ СУЧАСНИХ ДОСЛІДЖЕНЬ В ОБЛАСТІ ПОБУДОВИ ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ БІЛЬШОЇ РОЗМІРНОСТІ НА ПРОДУКЦІЙНІЙ МОДЕЛІ ЗНАНЬ

1.1 Актуальність продукційної моделі знань в інтелектуальних системах

До інтелектуальних систем (ІС) відносяться технічні або програмні системи, в функціонал яких входить вирішення завдань для конкретної предметної області, які традиційно вважаються творчими, знання, які зберігаються в пам'яті такої системи. Творчість тут розуміється в широкому сенсі: це і функції оператора технічних пристроїв, та транспортних засобів, і прийняття рішень в керуючих системах, і інформаційний пошук. У структурі ІС виділяють інтелектуальний інтерфейс, машину логічного висновку і базу знань. База знань є основою ІС, визначає її приналежність до конкретної предметної області та можливість вирішувати поставлені перед ІС завдання.

Знання можуть представлятися різними способами, як показано на рисунку 1.1. Найбільш древнім, але і найбільш широко застосовується в даний час, є уявлення знань у вигляді природно мовних текстів. Оскільки тексти спочатку орієнтовані на людину, для його сприйняття не потрібно підготовки. Недоліками текстового представлення знань є можлива неточність, неоднозначність і суперечливість, а також складність мовних конструкцій та різноманіття мов, які роблять практично неможливим його використання в ІС в якості сховища знань. Крім того, мовними засобами складно описати геометричні фігури, не кажучи про більш складні графічні об'єкти, тому тексти супроводжують ілюстраціями, які також не можуть сприйматися комп'ютером. При цьому текстове представлення знань залишається єдиним для багатьох логічно-суворих документів, таких, як закони, договори та ін.

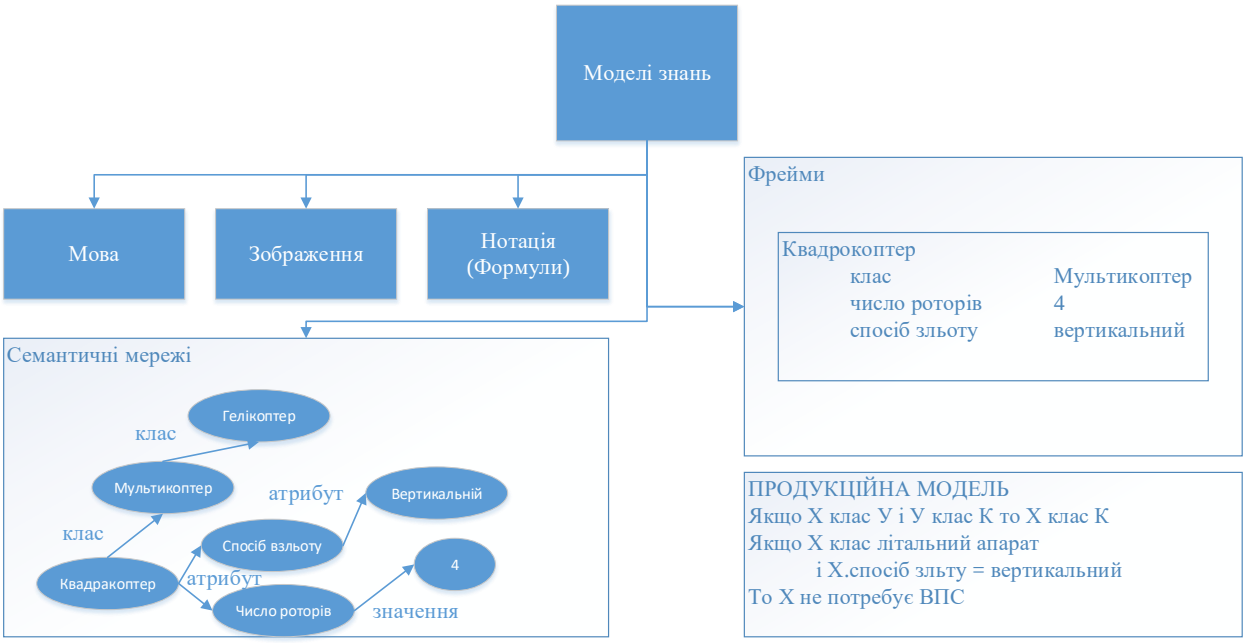

Рисунок 1.1 – Різновиди моделей знань

Зображення як модель знань володіє наочністю, не залежить від мови. Однак, такі зображення як креслення, електричні схеми, карти та ін. вимагають спеціальних знань і навичок для їх розуміння. Крім того, такі зображення не мають повноти і завжди забезпечуються легендою або тестовими поясненнями.

Повну протилежність природно-мовного подання знань становить математична або інша символьна нотація. Формули відрізняються граничною формалізацією і незалежністю від мови, проте вимагають від людини спеціальної підготовки і не є машиночитаною. Крім того, математична нотація оперує з досить обмеженим набором змінних (латинські і грецькі символи) і операторів, що робить її непридатною для великих предметних областей. Крім математичної існують і інші нотації, зокрема, нотний запис, що володіє своїми достоїнствами і недоліками. Формалізація знань у вигляді фреймів дозволяє абстрагуватися від граматики і синтаксису, але зберігає залежність від термінології. Крім того, фреймова структура диктує також обмеженість контексту заздалегідь заданими формами. Фрейми широко використовуються для формалізації анкет, платіжних документів і т.і.

Семантичні мережі мають гарну наочністю, яка зумовлює їх застосування для візуалізації знань. При цьому в них відсутня універсальність, зокрема, не відображаються заперечення і диз'юнкції, а спроби їх додати призводять до істотної втрати наочності.

Продукційна модель знань у вигляді правил на основі імплікації виду ЯКЩО А ТО В вільна більшості перерахованих недоліків.

Простота такого уявлення робить його зрозумілим без спеціальної підготовки, достатньо лише знати особливості операції імплікації (з В не слідує А і з А не слідує В, але з В слідує А).Порівняння характеристик різних моделей знань представлено в таблиці 1.1.

Переваги продукційної моделі по перерахованим вище критеріям забезпечили їх широке поширення близько 30 років тому в експертних системах таких, як MYCIN, PROSPECTOR, DENDRAL і ін. Наведене на початку параграфа визначення ІС не обумовлює її масштабу. На відміну від тесту Тьюринга, де наявність або відсутність штучного інтелекту прямо залежить від вмісту їм обсягу знань, тут мова може йти про досить прості технічні системи з вузьким набором функцій, нескладними алгоритмами і, отже, невеликою базою знань. Агентний підхід набув широкого поширення саме внаслідок вузького контексту, що дозволило обійти проблему експоненційної складності пошуку рішень.

Таблиця 1.1 – Характеристики моделей знань

Характеристики	Єстествена мова	Символьна нотація	Графічна	Фрейми	Семантична мережа	Продукційна модель
Повнота	+	+	-	-	-	+
Зрозумілість для людини	+	-	+	+	+	+
Точність	-	+	-	+	+	+
Універсальність	+	-	-	-	-	+

Продовження таблиці 1.1

Широкий контекст	+	-	-	-	+	+
Однозначність	-	+	-	+	+	+
Несуперечність	-	+	-	+	+	+
Незалежність від мови	-	+	+	+	+	+
Наочність	+	-	+	+	+	+

Постійне зростання продуктивності ЕОМ привило до того, що останнім часом стали з'являтися ІС великої розмірності, базу знань яких складають сотні тисяч і мільйони фактів. Серед них слід згадати ResearchCyc, розробку компанії Cycorp. Слід зазначити дослідний характер даної ІС і відсутність комерційних застосувань.

Крім проекту Cyc можна відзначити розробку ConceptNet, що ведеться в MIT, база знань якої налічує понад мільйон тверджень англійською мовою і ще бази знань меншого обсягу на дев'яти інших мовах. Спочатку ConceptNet позиціонувалася як інструмент для практичного моделювання міркувань над текстами на природних мовах. Як і Cyc, ConceptNet використовується на практиці для анотації зображень і в задачах розпізнавання мови, але в силу високої чутливості до контексту, її застосування для декларованих авторами більш складних завдань, зокрема, смислового розширення запитів до баз знань (query expansion) залишається неясним. ConceptNet має Web-сервіс Open Mind Common Sense, запущений в 2000 р. і використовує близько 14000 постачальників Web-контенту в стилі шаблонів пропозицій виду «The effect of eating food is <...>« (Результат прийому їжі це ...), «A knife is used for <...>« (Ніж використовується для ...) і т.п. MIT також підтримує проект ARIA (Annotation and Retrieval Integration Agent) – програмний агент для автоматичної ілюстрації повідомлень електронної пошти зображеннями.

Ще один ресурс, призначений для смислового розширення запитів, WordNet, що розробляється починаючи з 1985р. у Прінстоні і оптимізований для лексичної категоризації і визначення схожості слів, складається з англійських слів, організованих в 200 тис. смислових одиниць, і ієрархічних

відносин типу «is a». В силу нескладної структури і яка витікає з цього простоти використання, WordNet є більш успішним в порівнянні з ConceptNet. Практично всі відомі реалізації інтелектуальних систем є антропоцентричними і тільки асистують користувачеві. Наприклад, в сервісі Open Mind Common Sense для пропозиції «A rocket can <..> «(ракета може) пропонується кілька десятків варіантів, ранжируваних по семантичної наповненості, наприклад, hit, coach, land, slam (уразити ціль, наводити, приземлятися, наносити удар) і т.д. Більш ніж скромні досягнення інтелектуальних систем у порівнянні з обсягом вкладеного праці зазвичай пояснюються двома причинами: недостатнім числом знань початкового рівня в базі і комбінаторної складністю завдання логічного висновку [1]. Дійсно, у відміну від експертних систем (ЕС), в яких кількість фактів обмежена можливостями діалогу з користувачем або числом сенсорів в складі вбудованих ЕС, інтелектуальні системи мають потребу у великій кількості фактів, які повинні зберігатися разом з правилами. При цьому в будь-якій предметній області можуть зажадати знань з інших областей. М. Мінскі оцінює знання загального рівня на рівні 30-60 млн. зрозуміти про навколишній світ.

У той же час, завдяки збільшеній обчислювальної потужності, існуючі технічні та інформаційні системи стали інтелектуалізувати, тобто приймати на себе творчі функції. До таких систем можна віднести системи управління технологічними процесами, що включають в себе підсистеми SCADA, в яких за допомогою інтелектуальних алгоритмів на основі продукційних правил виконується рання діагностика, що дозволяє запобігати вихід керованих систем за межі допустимих параметрів або запобігати аваріям. Такі системи широко використовуються в енергетиці, трубопровідний транспорт, на підприємствах нафтохімії та ін. Таким чином, продукційна модель знань є найкращою для побудови інтелектуальних систем в силу повноти, точності, універсальності, зрозумілості для людини, можливості охоплення широкого контексту, наочності і незалежності від національної мови.

У зв'язку з тим, що інтелектуальними можуть бути навіть нескладні функції, які можуть вбудовуватися в пральні машини, пилососи і т.і., необхідно обмежити сферу застосування моделей і методів, які розробляються в рамках цього дослідження. Тут і далі мова буде йти про інтелектуальні системи великої розмірності, де проблема експоненційної складності пошуку рішень виступає на перший план.

Інтелектуальною системою великої розмірності будемо називати ІС, яка оперує з числом фактів, що вимірюється мільйонами. Для таких систем існуючі методи прискорення пошуку рішень є неефективними.

1.2 Місце проблеми прискорення пошуку при створенні інтелектуальних систем на продукційній моделі знань

Поняття комбинаторного вибуху, тобто експоненціального або сверхекспоненціального зростання складності рішення від розмірності вхідних даних було знайоме дослідникам задовго до появи комп'ютерів і спроб їх застосування до розв'язання задач пошуку на дереві рішень. Проте, в основу перших мов логічного програмування, зокрема механізму виведення Прологу, був покладений зворотний логічний висновок, від консеквента до антецеденту з обходом дерева рішень спочатку в глибину.

Складність зворотного логічного висновку $O_o\ (G)$ для мети G може бути визначена наступною рекуррентной формулою:

$$O_{o\,(G)}=f_{G+}\sum_{i=1}^{R_G}\prod_{I=1}^{C_i}O_o\ \ (i,j) \qquad (1.1)$$

де f_G – число фактів, релевантних цілі G, R_G – число правил, консеквент яких релевантний цілі G, C_i – число умов у i-му правилі, $O_o(i, j)$ – складність резолюції підцілі, породженої j-ою умовою i-го правила. Заздалегідь визначити кожну з підцілей неможливо, оскільки вони формуються в момент підстановки змінних в консеквента правил.

Процес прямого логічного висновку на відміну від зворотного виведення є передбачуваним, і його складність O_n можна визначити наступним чином:

$$O_n = \sum_{i=1}^{R} \prod_{j=1}^{Cr} f(i, j), \tag{1.2}$$

де R – кількість правил в базі знань, f (i, j) – число фактів в базі знань, релевантних j–му умові j-го правила, Ci – число умов у i-му правилі. На рівні середніх значень дану формулу можна спростити

$$O_n = Rf^C, \tag{1.3}$$

де f – середня кількість фактів, релевантних одній умові правила, C – середнє число умов у правилі.

Рисунок 1.2 демонструє зростання розмірності дерева рішень. З представлених даних видно, що можливості наявного висновку не виходять за рамки однозначних чисел f і c, що відповідає дуже простим завданням.

Несуттєве прискорення наївного виведення забезпечує механізм відсікань в Пролозі, який блокує відкат після невдачі і тим самим дозволяє усувати спуск по свідомо безперспективним гілкам дерева пошуку в випадках, коли рішення або єдине, або відсутній зовсім.

Однак, використання відсікань, так само як і інших аналогічних прийомів привели до того, що Пролог, будучи анонсованим в якості декларативного мови програмування, за фактом перестав бути таким, а програмування на Пролозі стало швидше мистецтвом, що підтверджується такими виданнями, як широко відома книга Л. Стерлінга і Е Шапіро [2]. Евристичні алгоритми пошуку на дереві рішення дозволяють досить ефективно долати комбінаторні складності пошуку, які виникають за рахунок швидкого зростання числа альтернатив. Однак для успішного застосування

таких алгоритмів необхідно мати оцінками перспективності кожної з вершин дерева пошуку з точки зору просування до цілі, що робить проблематичним застосування евристики, створеної для одного завдання, будь-яких інших. В якості еталонної задачі для випробування методів ШІ вже протягом півстоліття служить гра в шахи. Найпотужнішою шаховою програмою XX століття, яка перемогла Гаррі Каспарова в 1997 році, була Deep Blue,створена на платформі суперкомп'ютера в компанії IBM.

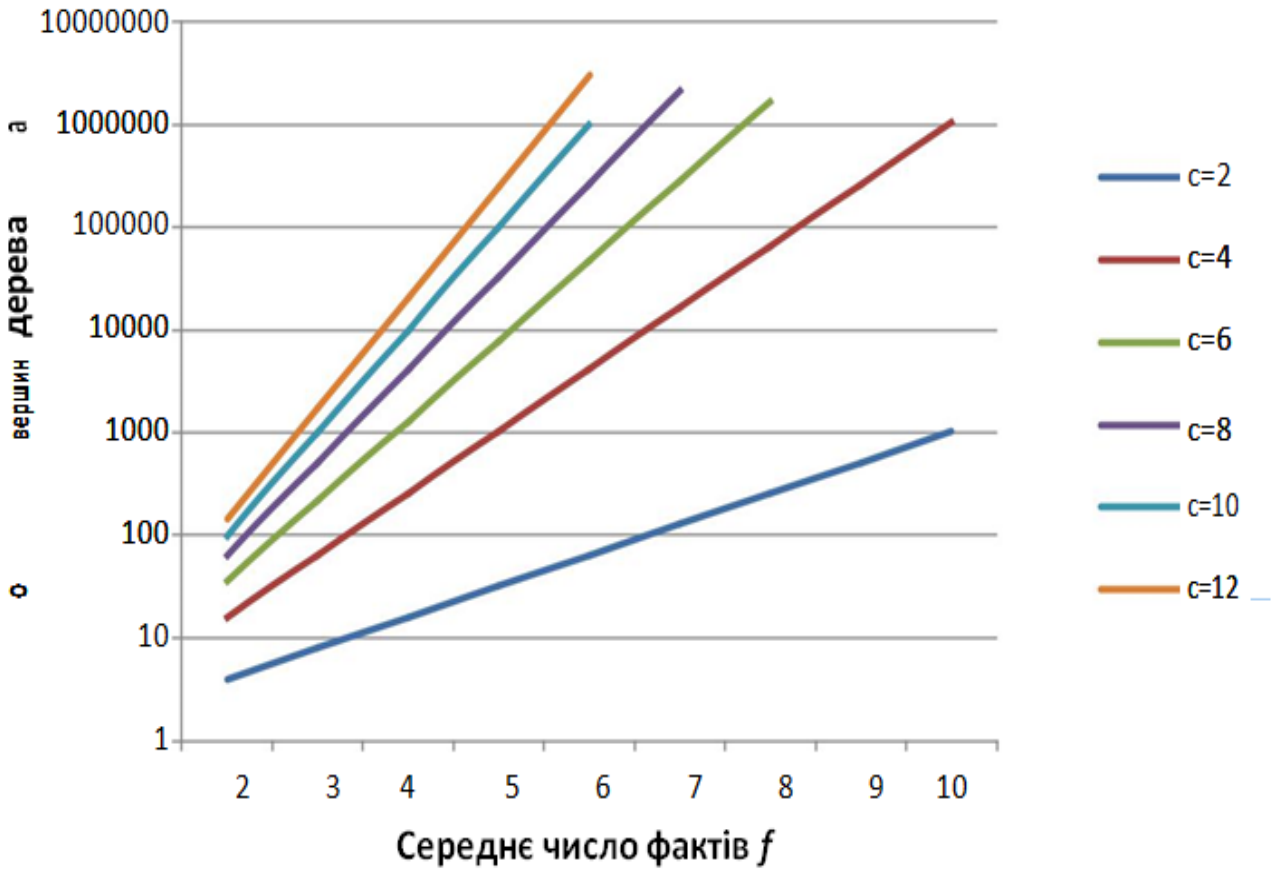

Рисунок 1.2 – Залежність числа розгорнутих вершин від числа фактів f і числа умов правила c

Таким чином, вибір евристик завжди є творчим процесом, а евристика, що дозволяє успішно вирішити завдання, для якої вона була побудована, не має властивість переносимості, а значить, є марною для інших завдань.

1.3. Аналіз існуючих методів підвищення ефективності механізмів пошуку в продукційних базах знань великий розмірності

Найбільш відомим методом прискорення резолюції правил є алгоритм RETE, створений Ч. Фордж в 1982 р [1], [2] і використовується в експертних системах CLIPS, JESS, SOAR і ін. В основу алгоритму покладено префіксне дерево, вузлами якого є умови правил. У кожному вузлі префіксного дерева створюється список фактів з бази знань, які задовольняють умовам правила.

Коли факти додаються в робочу пам'ять, створюються елементи робочої пам'яті (WME – working memory elements). Кожен елемент робочої пам'яті може містити один кортеж, або кожен факт може бути представлений набором елементів робочої пам'яті, де кожен елемент містить кортеж фіксованою довжини, зазвичай триплет.

Вузьким місцем алгоритму RETE є необхідність модифікації префіксного дерева при зміні, додаванні або видаленні фактів бази знань. При кожній модифікації бази фактів префіксне дерево має або оновлюватися, або Modify-in-place, Scaffolding, Decision Tree [3], [4], націлені саме на зміну префіксного дерева. Між тим, модифікація безлічі фактів відбувається в системах ШІ постійно, оскільки кожне правило дає в якості результату нові факти, які повинні відразу ж використовуватися в ході резолюції цілі.

1.4. Аналіз існуючих реалізацій інтелектуальних систем з позиції антропоморфного підходу

В середині минулого століття, коли тільки з'явилися ЕОМ, що дозволяють обробляти символьні дані, в ШІ переважав логічний підхід [4], [5] на основі математичної логіки і числення предикатів. Швидкість, з якої людина робить висновки, невелика, і навіть найперші комп'ютери виконували логічні операції істотно швидше. Мабуть, цією обставиною можна пояснити невиправданий оптимізм прихильників логічної школи ШІ

щодо термінів його практичної реалізації. Сучасні комп'ютери виконують набагато більше (в мільярди раз) логічних операцій в одиницю часу, ніж людина, але при цьому безнадійно програють йому в швидкості вирішення інтелектуальних завдань. Очевидно, що людина не в змозі змагатися з машиною в швидкості пошуку на дереві рішень і використовує інші техніки.

Якщо застосувати до штучного інтелекту антропоморфний підхід, що включає в себе тріаду «знання, уміння і навички», який добре зарекомендував себе в сфері освіти, то можна провести наступні аналогії:

Знання – факти і правила, що становлять традиційну базу знань в продукционной моделі.

Уміння – машина виведення (reasoner, inference engine), програмно–апаратні засоби, що забезпечують витяг відомих фактів і породження нових фактів шляхом застосування правил.

Навички для людини – дії, сформовані шляхом повторення і доведення до автоматизму. У штучному інтелекті аналогами навичок можуть вважатися прецеденти, тобто факти, виведені з правил, а також прості правила, синтезовані з ланцюжків правил (шляхом доведення теорем) і дозволяють отримувати результати з відомих фактів без складного пошуку.

Розглянемо ті ж різновиди систем ШІ в просторі компетенцій і контексту (рисунок 1.5). Напрямки розвитку ШІ, які можна вважати успішними, відзначені заливкою.

Характер розташування існуючих напрямків ШІ в цьому просторі дозволяє виявити наступні закономірності:

1. Існують реалізації систем ШІ, що діють тільки на основі знань в обмеженому контексті.

2. Успішні напрямки ШІ знаходяться або в області вузького контексту або в області переважання навичок.

3. Існує вкрай мало реалізацій систем ШІ, заснованих тільки на навичках, але в широкому контексті. В якості однієї з небагатьохуспішних реалізацій такої системи можна привести IBM Watson – концепція

семантичної павутини якраз і може заповнити цю прогалину. Широку базу фактів Semantic Web в певній мірі можна розглядати як матеріалізований досвід або навички.

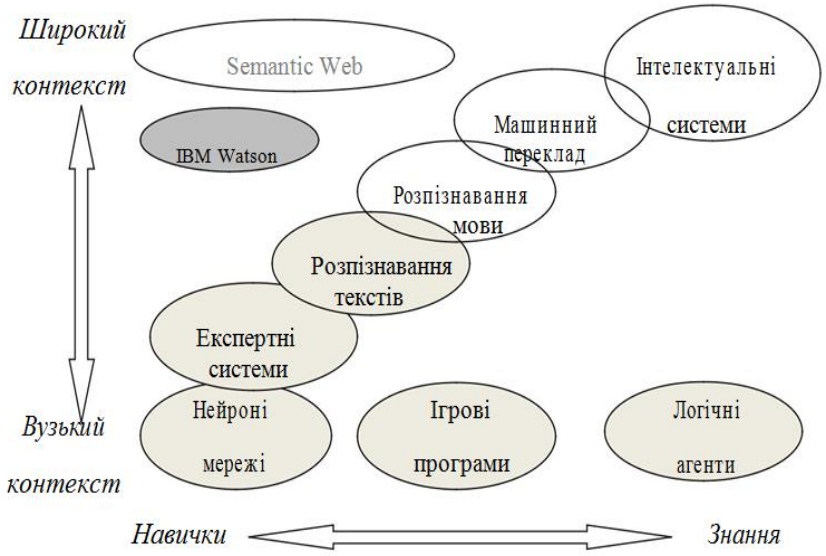

Рисунок 1.5 – Характеристики систем з в просторі вузький-широкий контекст і знання-навички

Таким чином, з вищесказаного можна зробити висновок, що підхід до створення інтелектуальних систем з широким контекстом (універсальних ІС) слід шукати не на логічному підході, а із застосуванням механізму інтелектуальних навичок. Необхідність відтворення навичок в ІС в загальному і цілому недооцінюється [5], [6]. Наприклад, в роботі Г.М. Звєрєва [7] інформаційна модель агента (об'єктивувати суб'єкта) включає в себе знання (інформацію про об'єкт) і вміння (сенсорний і рефорний процеси, а також процеси оцінки точності знань та адекватності реакцій агента), але повністю відсутній механізм навичок. Проте, в ШІ є напрямок досліджень, пов'язаних з використанням попереднього досвіду в нових ситуаціях – висновок на основі прецедентів.

2 МОДЕЛЬ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ НА ОСНОВІ БАЗИ ЗНАНЬ БІЛЬШОЇ РОЗМІРНОСТІ

2.1 Принципи організації баз знань з використанням прецедентів

Інтелектуальна система, яка реалізує в собі механізм навичок у вигляді прецедентів, повинна в першу чергу намагатися знайти готове рішення, яке отримане в процесі обробки попереднього ідентичного запиту, і тільки якщо прецедента, не знайдено, запускати машину виведення.

На рисунку 2.1 відображений алгоритм роботи інтелектуальної системи, яка відтворює в собі прецедентну модель знань і яка реалізована у вигляді технологічної платформи в програмі Semantic [7]. В першу чергу пошук виконується серед раніше знайдених рішень (прецедентів) і тільки якщо прецедента не знайдено, запускається машина виведення, а знайдене рішення поповнює базу прецедентів. Відсутність рішення найчастіше означає, що предметна область (домен) обрано невірно, а суперечливість результату є свідченням занадто широкого домену і потрібно його уточнення. У схемі даного алгоритму відсутня гілка для ситуації, коли рішення не може бути знайдено. Рішення в даній ситуації має прийматися окремо для кожного об'єкта.

Створення інтелектуальної системи з використанням прецедентів пов'язане з проблемами в частині:

- створення прецедентів (необхідність прискорення логічного висновку);
- актуальності прецедентів в умовах мінливої бази знань;
- розміщення і адміністрування прецедентів;
- суперечливості фактів;
- відкритого і замкнутого світу;
- формування запитів до бази знань.

Незважаючи на те, що в роботі інтелектуальної системи повинні переважати звернення до прецедентів, щоб прецеденти з'явилися, їх потрібно створити. Хоча прецеденти можуть створюватися як емпіричні факти, нормальним способом їх створення слід вважати запуск машини виведення.

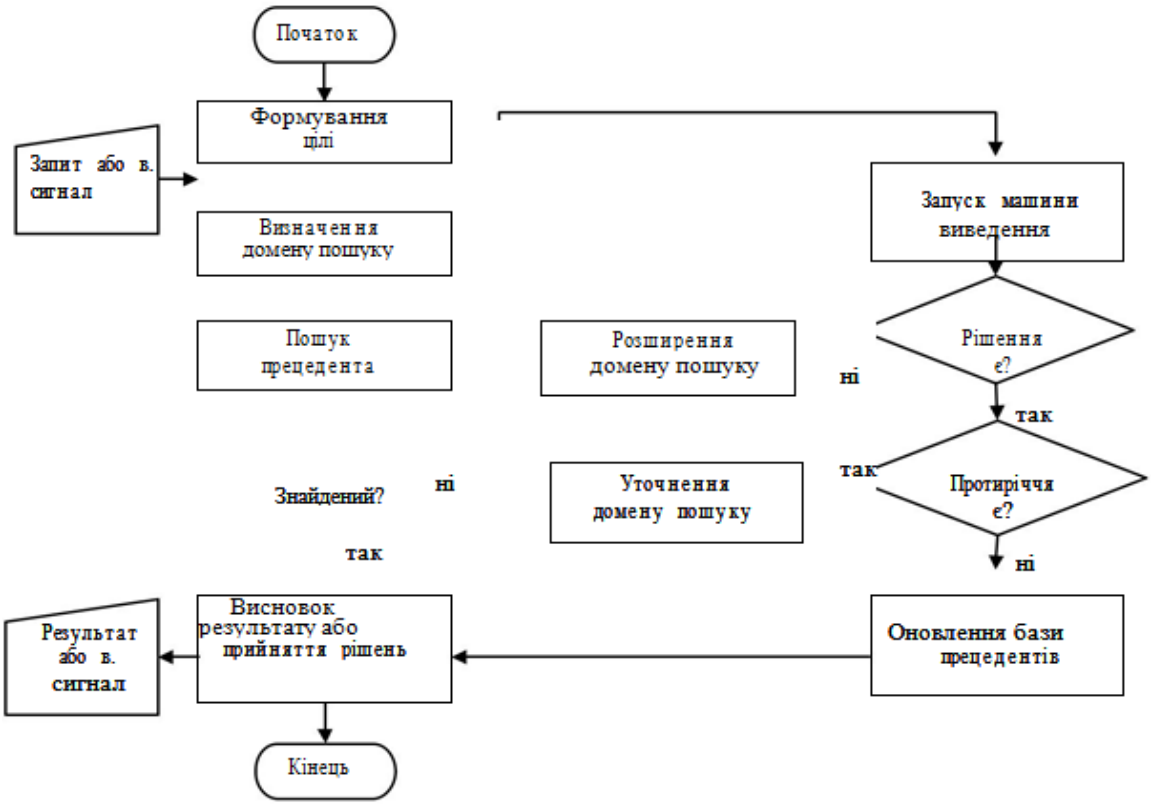

Рисунок 2.1 – Алгоритм пошуку рішень з використанням прецедентів

Наведений на даному рисунку алгоритм не відображає ту обставину, що в умовах мінливої ази знань (зміни, видалення або появи нових фактів) прецеденти можуть застарівати, тому потрібно контролювати їх актуальність. Контроль актуальності прецедентів може здійснюватися в наступні моменти часу:

- при добуванні прецеденту з бази знань;
- періодично за розкладом;
- при появі подій, які роблять прецеденти неактуальними.

2.2 Модель машини виведення для баз знань великий розмірності

Як показано в роботах [10], [11], семантична павутина (Semantic Web), спочатку була задумана як сховище фактів, доступних для пошуку за допомогою інтелектуальних агентів, але теоретично може стати базою знань глобального штучного інтелекту, доступного для вбудованих систем, які спочатку не можуть спиратися на потужну локальну базу знань, але мають доступ до Інтернет-ресурсів.

На відміну від логічних агентів, агенти для семантичної павутини діють не в реальному, а віртуальному формалізованому просторі і поряд з даними від сенсорів використовують контекст користувача. Інформаційну модель такого агента можна уточнити в такий спосіб:

$$usr \xrightarrow[G,cSW,P]{} q \longrightarrow y \longrightarrow usr, \tag{2.1}$$

де usr – користувач, що генерує мета, q (G, c) – запит до бази знань, що формується на основі мети G, доповненої контекстом з користувача, y (SW, P) – результат пошуку в базі знань SW і базі прецедентів P. Контекст користувача, необхідність якого оголошена в концепції Семантичної павутини [12], дозволяє формулювати мету G користувача в скороченій формі або повністю автоматично. База прецедентів P забезпечує швидкий доступ до раніше витягнутим фактами.

Таким чином, визначальним фактором для успішної реалізації інтелектуальних систем є матеріалізований досвід інтелектуального агента, який було збережено у вигляді прецедентів, який дозволяє або отримувати готові факти з пам'яті або обходитися дуже простими міркуваннями, не заглиблюючись в ланцюжка правил. Для цього база знань системи штучного інтелекту повинна бути доповнена базою прецедентів.

Спрощена схема взаємодії інтелектуального агента з Семантичної павутиною зображена на рисунку 2.2. Якщо виключити зображені на схемі вторинні факти, то ми отримаємо звичайну продукційну систему.

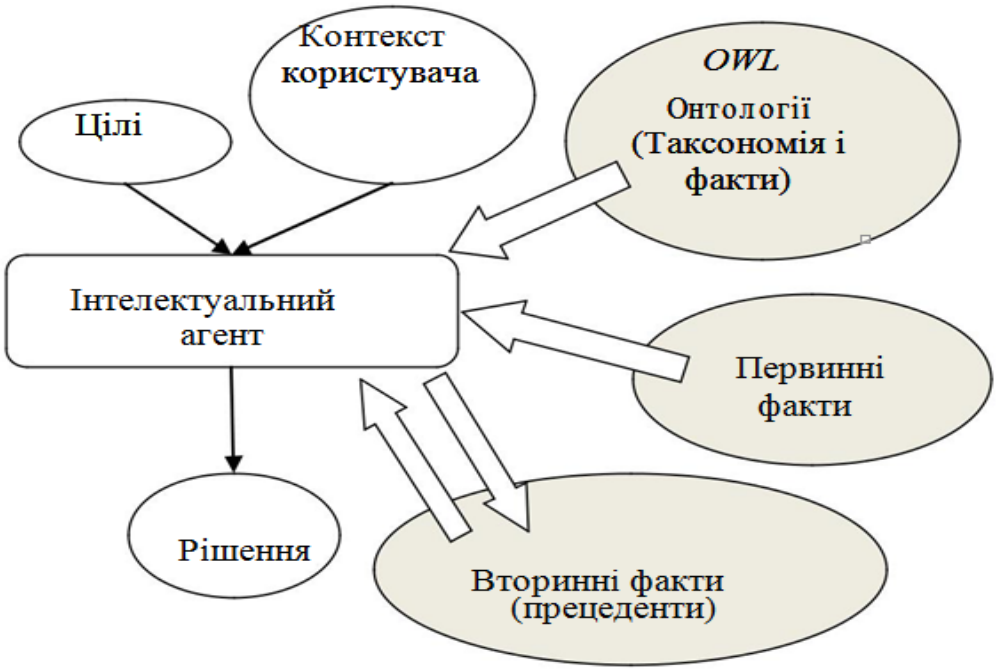

Рисунок 2.2 – Система Інтелектуальний агент – Semantic Web

На рисунку 2.3 наведено спрощений алгоритм роботи сервера прецедентів.

Під час отримання запиту виконується пошук в базі прецедентів. Якщо прецедент знайдений, виконується перевірка його актуальності шляхом пошуку наявності в базах знань фактів, які використовувалися при створенні прецеденту, і якщо актуальність підтверджується, то прецедент видається в якості результату. В іншому випадку прецедент видаляється з бази і виконується пошук в базах знань. Знайдене рішення зберігається як прецедент.

В даному алгоритмі реалізується метод управління прецедентами за запитом. Можливі також методи управління прецедентами за розкладом і по настанню подій.

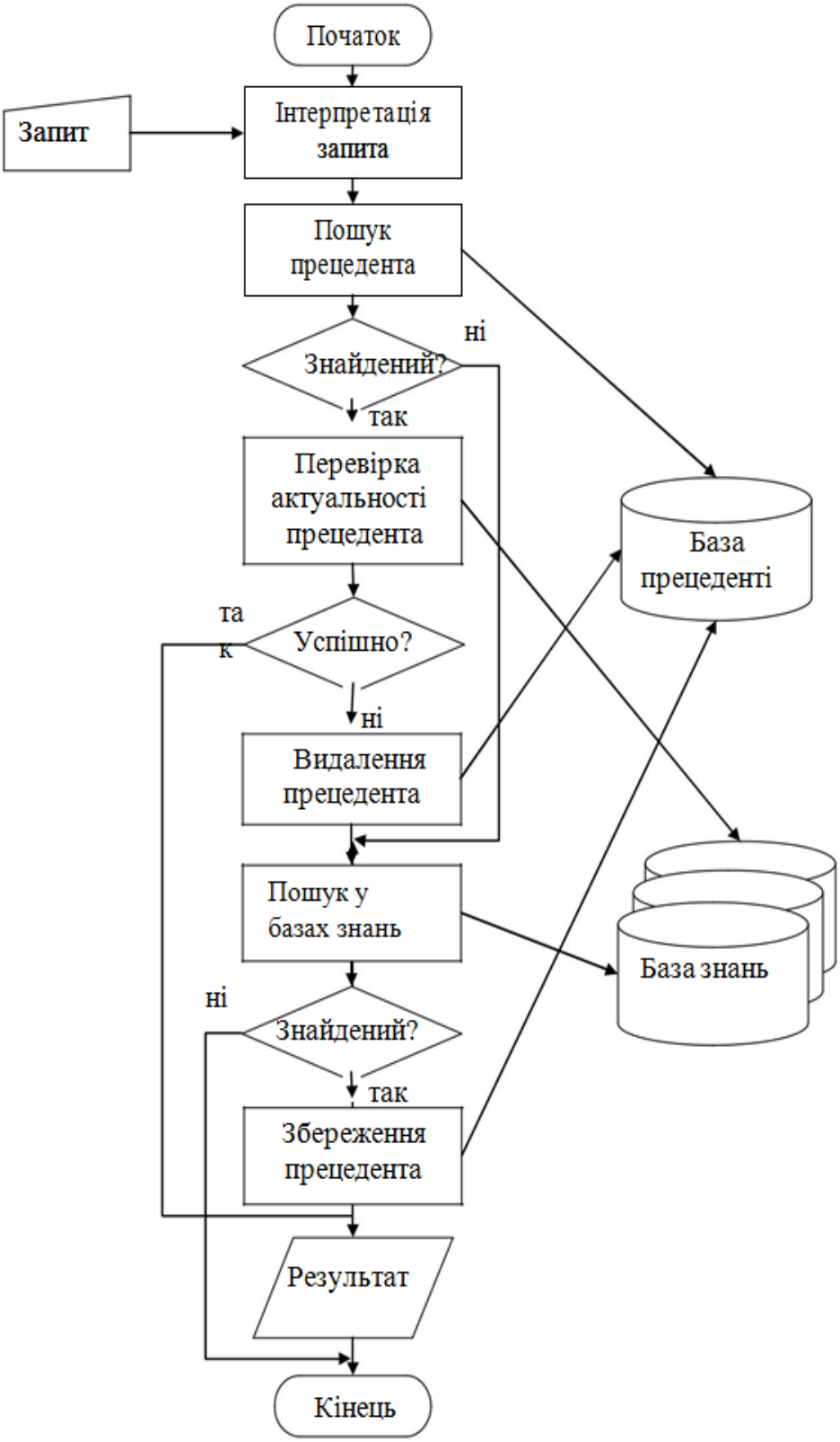

Рисунок 2.3 – Алгоритм роботи сервера прецедентів

2.3 Перспективні інтелектуальні системи великій розмірності

Прикладами сучасних інтелектуальних систем великої розмірності можуть служити системи оперативного управління складними виробничими процесами на об'єктах промисловості, транспорту, зв'язку, енергетики і т.і., а одним з об'єктів, який заслуговує уваги слід визнати аеропорт, складність якого як об'єкта управління зумовлена такими факторами:

- залежність від багатьох зовнішніх чинників, таких, як метеоумови, робота інших аеропортів і наземного транспорту;
- непередбачуваність поведінки пасажирів (запізнення на посадку і відмови від польоту після проходження реєстрації тощо);
- складні технологічні графіки підготовки літаків до вильоту;
- динамічний характер основної діяльності (оперативні заміни типів літаків, об'єднання та скасування рейсів, чартерні програми).

Спрощена схема функціонування центру оперативного управління аеропортом представлена на рисунку 2.4. Основним завданням центру оперативного управління (ЦОУ) є організація роботи оперативних служб аеропорту в умовах збоїв добового плану. Базу даних ЦОУ складають добовий план польотів, база штурманського забезпечення (картографічна інформація, телеграми про зміни на аеродромах і ін.) база метеоданих, технологічні графіки підготовки літаків до вильоту і ін.

Поряд з базою даних ЦОУ використовує в своїй роботі безліч правил і обмежень, що відносяться до технологічних процесів обслуговування прильотів і вильотів, управління повітряним рухом, обслуговування пасажирів та ін.

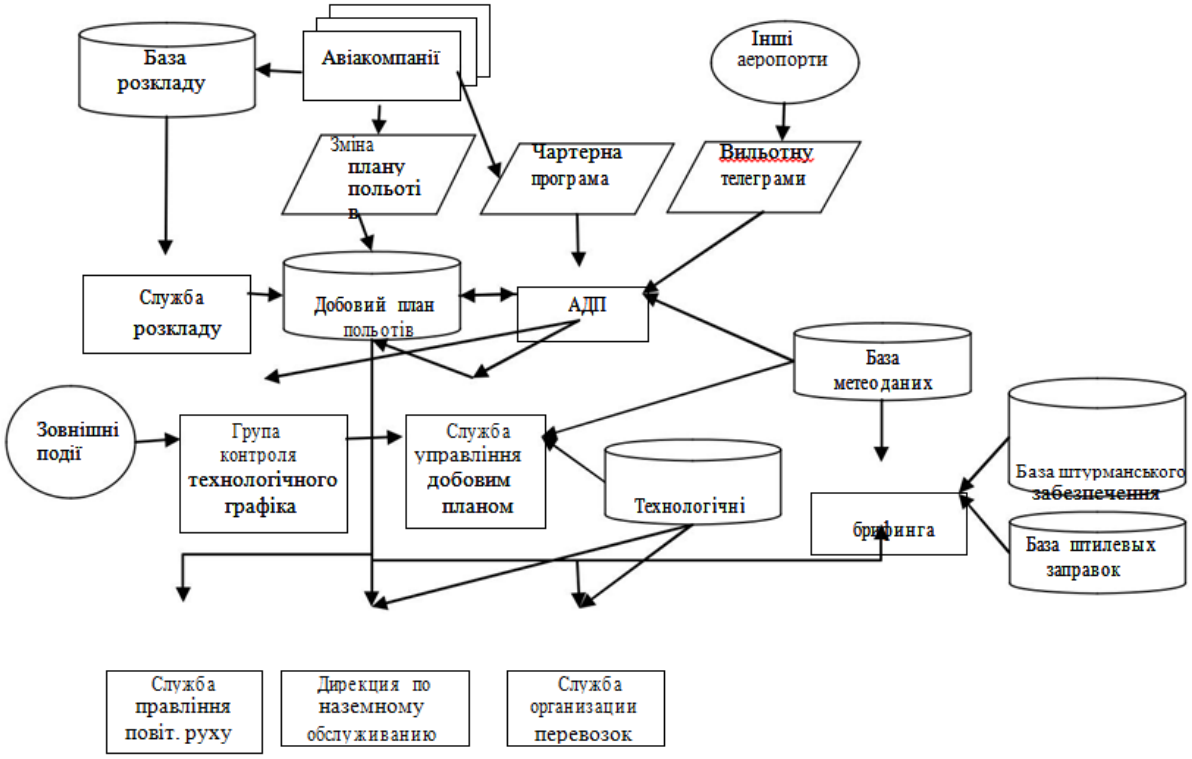

Рисунок 2.4 – Схема функціонування центру оперативного управління

У зв'язку з тим, що ресурси авіакомпаній (літаки та екіпажі) і аеропорту (паливозаправники, тягачі, автобуси, трапи, стоянки, багажні транспортери, зони огляду, паспортного контролю і т.п.) є обмеженими, затримка навіть одного вильоту або прильоту може спричинити за собою ланцюжок збоїв добового плану. У таких випадках необхідна його оперативне коректування, що представляє собою задачу багатокритеріальної оптимізації в умовах недостатньої інформації.

Використання механізму дедуктивного виведення з правил дозволяє в нескладних випадках автоматично скорегувати добовий план польотів. Однак в ситуаціях масового збою добового плану швидкість пошуку рішень не дозволяє вирішувати цю задачу в реальному масштабі часу, що обумовлює необхідність залучення до цієї роботи великої кількості фахівців в режимі цілодобового чергування. Тим часом, є можливість зберігати у вигляді прецедентів результати попередніх рішень і використовувати їх в подальшому.

Ще однією з інтелектуальних систем великої розмірності є глобальна автоматизована система бронювання (АСБ) авіаперевезень, яка об'єднує окремі національні і міжнародні АСБ, глобальні розподільні системи (ГРС), а також перевізників і агентів. Комплекс технічних засобів глобальної АСБ складають сотні серверів на базі мейнфрейм-комп'ютерів, сотні тисяч терміналів агентів і не один мільйон Інтернет-клієнтів. Кожна АСБ обслуговує деякий безліч авіакомпаній і здійснює резервування і продаж авіаквитків для цих авіакомпаній. Однак далеко не завжди запит клієнта може бути виконаний рейсами своїх авіакомпаній, і потрібно стикування з рейсами інших авіакомпаній. Крім того, існують незалежні (Нейтральні) агенти, які в умовах конкуренції повинні пропонувати клієнтові найбільш прийнятні для нього умови перевезення, для чого необхідно звертатися до баз даних інших АСБ в ролі клієнтів. Нарешті, для забезпечення взаєморозрахунків використовується транспортноклірингова палата (ТКП), оскільки продаж квитків здійснюється водному місці, а послуги надаються в іншому місці або декількох місцях.

Інформаційну базу даних такої розподіленої системи складають бази даних всіх АСБ, які містять:

- дані про авіакомпанії, аеропортах, державах, містах і географічних зонах;
- дані з розкладу кожної авіакомпанії;
- тарифи на авіаперевезення і сезони;
- використовувані в системі валюти і курси валют;
- дані про наявність місць на кожен рейс, дані про кожному бронюванні і кожному проданому квитку.

Таким чином, процес бронювання авіаперевезень представляє собою завдання пошуку на дереві рішень з коефіцієнтом розгалуження порядку десятків або навіть сотень (число повітряних трас з кожного аеропорту) на великих даних.

Вилучення знань в інтелектуальних системах на продукционной моделі знань являє собою задачу пошуку на дереві рішень, має складність O (bd), де b – коефіцієнт розгалуження, d – глибина пошуку. Дана обставина обумовлює неможливість вирішення завдань реального світу методами обходу дерева. Існуючі рішення або обмежуються вузьким контекстом, або не використовують повною мірою можливості дедуктивного виведення, обмежуючись простими ланцюжками правил. В алгоритмах пошуку рішень складних завдань пошуку зазвичай використовують евристики, що володіють істотним недоліком – унікальністю для кожного конкретного завдання, що унеможливлює їх застосування в інших предметних областях.

Нехай є інтелектуальна система <E, K>, де E – машина виведення, K = <F, R> – продукційна база знань, F = {f} – безліч аксіом (Фактів), R = {r} – безліч правил. Користувач інтелектуальної системи або об'єкт управління описується вектором <G, S, X, Z>, де G ={G} – безліч цілей, S = {s} – безліч станів, X = {x} – контекст користувача або об'єкта, тобто безліч фактів, що описують його оточення, Z = {z} – безліч рішень. Функціонування ІС полягає в знаходженні рішень d (g, s, X) як функції від стану об'єкта, поточної мети і наявних знань про об'єкт і його оточенні(Контексті).

Робота машини виведення полягає у встановленні істинності фактів,обумовлюють істинність мети, а управлінський вплив має бути спрямоване на зміну стану об'єкта s на s', відповідне поставленої цілі. У тому випадку, якщо в базі знань є готове рішення z (g, s, X), витяг його з бази знань не викликає особливих проблем, а якщо готового рішення немає, потрібне застосування продукційних правил, тобто пошук на дереві рішень, і складність O (bd) роботи машини виведення стає експоненційної, де b – коефіцієнт розгалуження (число фактів, релевантних умові правила), d – глибина пошуку, тобто число умов у правилі або глибина вкладеності правил.

Для подолання експоненційної складності пошуку слід мінімізувати кількість запусків машини виведення. Цього можна домогтися шляхом

накопичення готових рішень (прецедентів) z (g, s, X), зберігаючи для кожної мети g, контексту X і стану s. Даний підхід є аналогом інтелектуальних навичок людини. Основною проблемою тут є можлива мінливість, як об'єкта, так і контекстного оточення, що зробить після успішної реєстрації прецеденти неактуальними. У зв'язку з цим потрібно розробити методи управління прецедентами, що забезпечують їх актуальність.

Так як прецеденти можуть бути універсальними або створюватися для конкретного об'єкта і контексту, створення бази прецедентів може зажадати тривалого часу, а в умовах розподіленої бази знань виникає задача розміщення прецедентів. Прецеденти можуть бути приписані або до об'єкта (користувачеві), до бази знань, в якій вони створені, або до окремого сервера прецедентів. Оскільки створення прецедентів забезпечується роботою машини логічного висновку, необхідно також провести дослідження і розробити методи прискорення пошуку на дереві рішень. У зв'язку з тим, що вилучення знань з Всесвітньої павутини стає доступним для все більш широкого кола додатків, в т.ч. мобільних, виникає також проблема автоматичної генерації запитів до ресурсів Інтернету, а також управління контекстом пошуку. Таким чином, рішення перерахованих проблем створює теоретичні та методологічні основи для побудови інтелектуальних систем на продукционной моделі знань, що спираються на бази знань великого обсягу.

3 МЕТОДИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ПОШУКУ В БАЗАХ ЗНАНЬ ВЕЛИКОЇ РОЗМІРНОСТІ

3.1 Редукція дерева рішень шляхом усунення повторювання гілок

3.1.1. Метод усунення повторюваних гілок в межах одного дерева рішень

Нехай база знань складається з безлічі правил R, що утворюють ланцюжки за рахунок того, що консеквента одних правил присутні в антецедентах інших правил. Розглянемо для цієї бази знань класичну задачу неінформованого пошуку методом зворотного виведення (back chain reasoning), складність якої можна оцінити таким чином. Нехай r –середня кількість правил, консеквента яких релевантні запиту при кожному зверненні до бази знань, n – середнє число умов (антецедентів) в тілі кожного правила, d – середня глибина вкладеності правил. Тоді дерево пошуку буде містити N вершин:

$$N = r + rn + (rn)^2 + \cdots + (rn)^d = r + \sum_{i=1}^{d} rn^i \qquad (3.1)$$

Рисунок 3.1 відображає залежність складності дерева пошуку від числа правил і кількості умов в правилі для числа предикатів в тілі правила n = 6. Якщо швидкість пошуку вважати рівною 10000 правил в секунду, то час повного обходу дерева для r = 20, n = 6 і d = 6 складе 9,5 років. При цьому, наприклад, Цивільний Кодекс України налічує біля 1 542 статті по кілька пунктів в кожній. Таким чином, база знань на основі кодексу буде складатися з декількох тисяч правил, і будь-який запит на юридичну тему буде звертатися до всіх цих правил.

Оскільки подібні правила застосовуються для різних суб'єктів права, агент може намагатися застосувати до конкретного випадку правила для підприємців, громадян або юридичних осіб, що відносяться до внутрішньодержавного або міжнародного права. Отже, кількість правил, релевантних запиту, може бути досить великим.

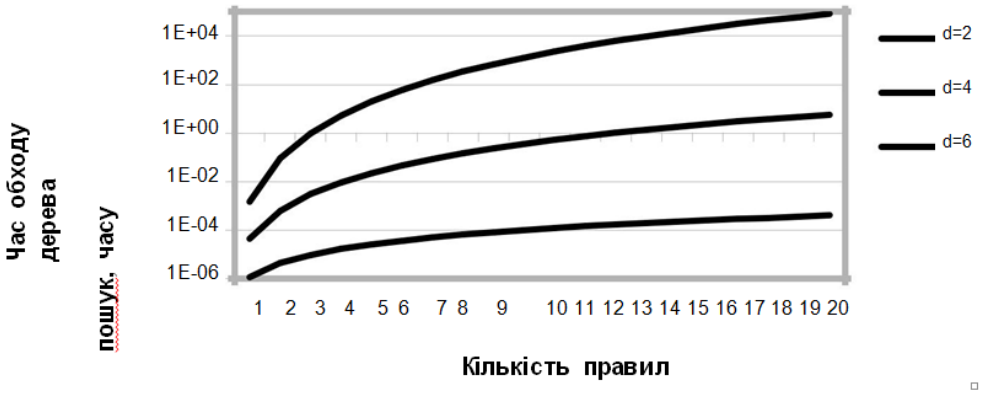

Рисунок 3.1 – Залежність часу обходу дерева пошуку від кількості правил

Розгортання вершин дерева пошуку на розподілених ресурсах вимагає, щоб ці ресурси були знайдені і завантажені. У разі застосування алгоритму обходу дерева спочатку вглиб потрібно Q операцій пошуку, яке може бути обчислено за формулою:

$$Q = 1 + rn + (rn) \tag{3.2}$$

Якщо використовувати алгоритм пошуку спочатку в ширину, то кількість операцій пошуку дорівнюватиме глибині вкладеності правил d, але потрібно запам'ятовувати $N–(rn)^{d}$ (все, крім останнього рівня) результатів пошуку. Для наведеного вище прикладу обходу дерева для r = 20, n = 6 і d = 6 буде потрібно близько 25 терабайт за умови, що одна посилання займає 1Кбайт пам'яті.Таким чином, пошук шляхом простого перебору може

виявитися дуже тривалим і навіть нескінченним. Алгоритми прискорення логічного висновку базуються на створенні префіксних дерев, що складаються з списків фактів, релевантних умовам правил. При зверненні до правил залишається тільки підставити в нього заздалегідь відібрані факти. Якщо замість цих фактів зберігати відразу результати виведення з правил і використовувати їх як готові факти, швидкість виведення буде радикально збільшено. Застосування прецедентів до задачі пошуку може виглядати наступним чином. Результати успішного пошуку можуть зберігатися у вигляді фактів і будуть надалі доступними для швидкого доступу. Основна проблема тут – неминуче накопичення суперечливих фактів при зміні контексту, яке тягне за собою необхідність їх подальшого пошуку і видалення в великому масиві накопичених фактів. Крім того, потрібно скасовувати застосування правил, на яких створені такі факти. В іншому випадку дерево рішень стане тільки більше розгалуженим, і замість скорочення пошуку матиме місце його істотне ускладнення.

Застосування прецедентів має сенс навіть всередині однієї операції пошуку (короткочасна пам'ять). Розглянемо як приклад відому логічну гру «23 сірники». Правила гри такі: два гравці по черзі беруть з купки одну, дві або три сірники. Програє той, хто бере останній сірник. Нижче наведено фрагмент програми, розробленої автором на мові Prolog і реалізує логіку пошуку рішення у вигляді рекурсивного виклику правила пошуку хорошого ходу [95]. Хороший хід – це такий хід, після якого залишається одна сірник. Якщо це неможливо, то хороший хід – такий хід, після якого у супротивника немає хорошого ходу.

```
% Допустимі ходи
move (1). move (2). move (3).
% Правило хорошого ходу
% Хід, відразу веде до виграшу
good_move (X, M): - move (M), X - M = 1.
% Якщо за один хід виграти неможливо good_move (X, M): -
move (M), X1 is X-M,
not (good_move (X1, _)).
```

Приклад 3.1 – Псевдокод хорошого ходу

Рисунок 3.2 відображає фрагмент дерева пошуку для проміжного стану 6 сірників. Число в вершині означає кількість сірників, що дуги – ходи гравців. Повний обхід даного фрагмента вимагає розгортання 28 вершин. Пунктиром на графі позначені повторювані фрагменти. Якщо запам'ятовувати результат першого обходу кожного з таких фрагментів, це позбавить від необхідності повторного поглиблення і скоротить число розгорнутих вершин до 12 (виділені заливкою), тобто більш ніж в два рази.

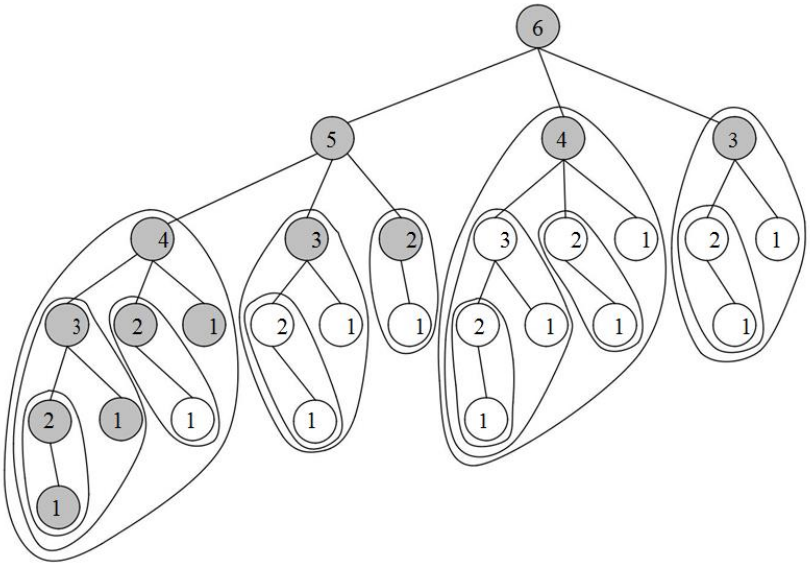

Рисунок 3.2 – Навчання на повторюваних фрагментах дерева пошуку

Нижче наведено текст модифікованої програми на мові Prolog, що включає висновок на основі прецедентів. Напівжирним шрифтом відзначені додані предикати. Оновлена логіка програми передбачає спочатку пошук рішення серед прецедентів, які запам'ятали, а якщо такого не знаходиться, то відбувається спуск по дереву, а знайдене рішення запам'ятовується в базі прецедентів для використання при пошуку на сусідніх гілках дерева.

```
% Допустимі ходи
move (1). move (2). move (3).
% Правило хорошого ходу - якщо є прецедент
good_move (X, M): - precedent (X, M),!.
% Якщо прецеденту немає
% Хід, відразу веде до виграшу
```

```
good_move (X, M): - move (M), X - M = 1.
% Якщо за один хід виграти неможливо good_move (X, M): -
move (M), X1 is X-M,
not (good_move (X1, _)),
% Додавання прецеденту в базу прецедентів assert (precedent (X, M)).
```

Приклад 3.2 – Додавання прецеденту в базу прецедентів assert

Повне дерево пошуку для початкового стану 23 сірники містить 900140 вершин, а пошук до першого рішення – 20009 вершин; в цьому випадку запам'ятовування проміжних результатів скоротить обхід до 57 вершин або в 351 раз. База прецедентів для цієї програми буде виглядати наступним чином:

```
precedent(2,1).    precedent(10,1).
precedent(3,2).    precedent(11,2).
precedent(4,3).    precedent(12,3).
precedent(6,1).    precedent(14,1).
precedent(7,2).    precedent(15,2).
precedent(8,3).    precedent(16,3).
```

Приклад 3.2 – База прецедентів

Фактично ця база прецедентів містить в собі виграшну стратегію гри, яка полягає в тому, що супротивнику слід залишати 1, 5, 9, 13, 17, 21 сірник. Після створення бази прецедентів на всю глибину дерева рішень наступний виклик програми взагалі не буде звертатися до дерева рішень. Зауважимо, що в тексті програми ця стратегія не присутній, а програма її синтезує самостійно.

Висновок на основі прецедентів може породжувати великі обсяги вторинних фактів, доступ до яких також може збільшувати загальний час доступу до бази знань. Нехай в базі знань є F фактів з часом вилучення одного факту t_f і R ланцюжків правил з часом виведення t_r. Тоді середній час доступу до бази знань

$$T = p(f) \cdot F \cdot t_f + (1 - p(f)) \cdot t_r, \qquad (3.3)$$

де p (f) – ймовірність того, що шуканий факт є в базі знань, (1 – p(f)) – ймовірність того, що машина виведення буде звертатися до правилу. Для простоти можна вважати, що система звертається тільки до одного правила.

При кожному виведенні з правила машина виведення породжує *B* нових фактів, які поповнюють базу знань. Таким чином, при кожному наступному зверненні до бази знань ймовірність виявити прецедент без поглиблення в ланцюжка правил постійно збільшується, а база вторинних фактів зростає, що теж може збільшувати час доступу до бази знань.

Рисунок 3.3 ілюструє динаміку часу доступу до бази знань при p (f) = λF / (λF + R), де λ = Fr / F – частка релевантних фактів при зверненні до бази знань, $R = 1000$, $t_f = 10^{-6}$ с, $t_r = 1000$с, $B = 500$. Графік демонструє, що спочатку час доступу до бази знань падає за рахунок прямого вилучення фактів замість пошуку в ланцюжках правил, потім зростає внаслідок різкого збільшення обсягу бази фактів.

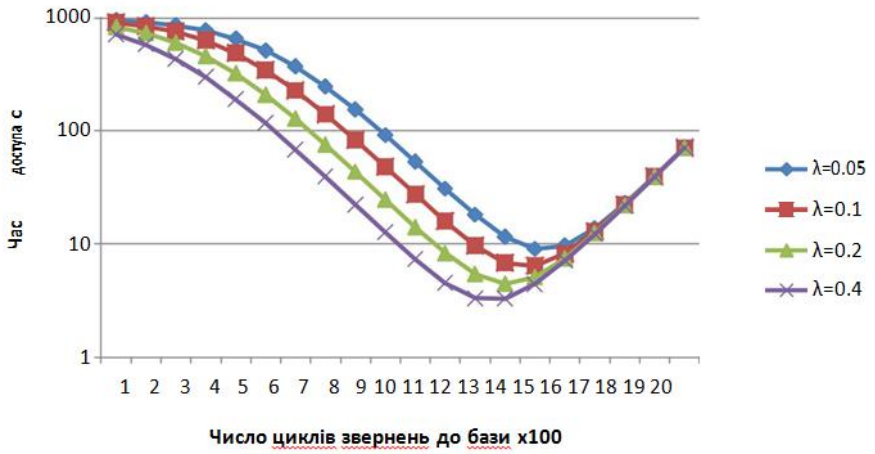

Рисунок 3.3 – Динаміка часу доступу до бази знань при зростанні числа прецедентів

У даній гіпотетичної моделі кожне звернення до правила генерує фіксоване число вторинних фактів. У реальних інтелектуальних системах можна очікувати різкого збільшення числа вторинних фактів при першому виклику кожного правила за рахунок збереження проміжних результатів при

обробці ланцюжків правил, що рівносильно поширенню комбинаторного вибуху в просторі замість часу. Отже, при автоматичному збереженні вторинних фактів під час обробки правил виникає завдання скорочення бази прецедентів, що є аналогом процесу забування в людському інтелекті.

Таким чином, при створенні прецедентів в межах одного дерева пошуку мають місце два процеси: редукція глибини дерева за рахунок заміни повторюваних гілок одиничними вузлами (прецедентами) і збільшення ширини внаслідок зростання числа прецедентів.

3.1.2. Прискорення пошуку при прямому і зворотному логічному висновку

У попередньому параграфі показано, що зберігання та застосування прецедентів в рамках одного пошуку дозволяє усунути повторення спуску по повторюваним фрагментами дерева пошуку. Таким чином, економія часу тут досягається за рахунок витрати пам'яті, і далеко не завжди прецеденти можуть бути записані в такий компактній формі, як це зроблено в наведеному вище прикладі. Слід зауважити, що в даному випадку все прецеденти використовуються як є, тобто не існує проблеми пошуку найбільш відповідного прецеденту. Однак така проблема виникає при нечіткому логічному висновку.

При прямому логічному висновку кожне правило породжує від 0 до bd нових фактів, де b – число фактів релевантних одній умові правила (Коефіцієнт розгалуження), d – число умов у правилі.

Нехай при прямому виведенні з правила отримані нові факти, які поповнили базу фактів. Отже, при наступному зверненні до бази знань для отримання даних фактів немає необхідності повторно обробляти це правило. Однак, якщо нові факти будуть доречними умов правила (того ж самого або будь-якого іншого), то буде потрібно повторний пошук за всіма правилами. При цьому будуть повторно породжені всі факти, отримані при

попередньому прогоні правил. Природно, заздалегідь невідомо, породить новий прогін правил нові факти, і сигналом до закінчення процедури прямого виведення може бути тільки відсутність нових фактів на виході.

Таким чином, в разі вкладених правил запам'ятовування проміжних фактів є необхідним, але не призводить до прискорення виведення. Більш того, для виключення багаторазового дублювання фактів при кожному додаванні необхідно перевіряти, чи немає вже такого факту в базі.

Особливість прямого логічного висновку полягає в тому, що правила застосовуються наосліп до всіх відомих фактів, оскільки невідомо, приведе дане правило до мети чи ні. Необхідність підставляти в умови правил всі можливі поєднання фактів обумовлює високу комбінаторних складність прямого логічного висновку і витрати пам'яті для зберігання всіх проміжних результатів. Позитивною стороною прямого логічного висновку є породження всіх можливих результатів, навіть якщо в даний момент потрібно з'ясувати один єдиний факт, і наступний запит буде виконаний дуже швидко, якщо звернеться до факту, який вже виведений раніше.

Розглянемо просте генеалогічне дерево (рис. 3.4), на якому двосторонні зв'язки – подружні, а односторонні – батьківські. Зазначені зв'язку є первинними, а решта – похідними від них.

Кожен член сімейства може мати максимум три входять зв'язку (один чоловік і двоє батьків, причому подружній зв'язок – одна на двох).

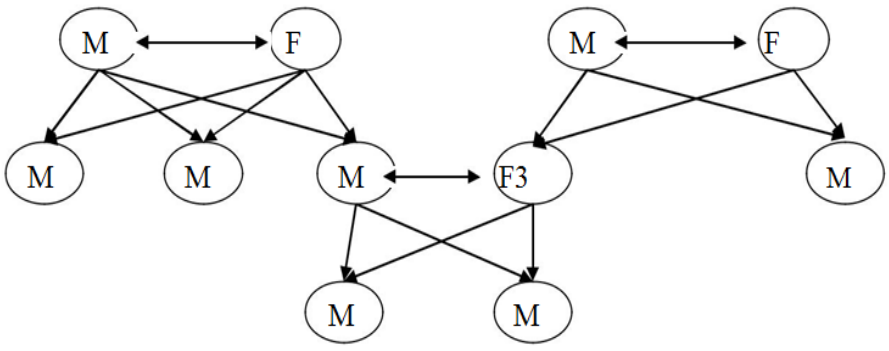

Рисунок 3.4 – Граф родинних відносин з первинними зв'язками

Таким чином, верхня межа числа первинних зв'язків Lp в генеалогічному дереві дорівнює Lp = 2,5N, де N – число членів сімейства.

Дерево на рисунку 3.4 містить 11 вузлів і 17 зв'язків з 27 можливих. Тим часом, в сім'ї кожен пов'язаний з кожним родинними відносинами, мають до того ж унікальні назви. Таким чином, загальне число родинних зв'язків L = N (N-1) або для даного прикладу L = 110 (рисунок 3.5).

Крім того, між двома вузлами генеалогічного дерева може існувати більше однієї зв'язку: X чоловік Y, X чоловік Y, X близький_родич Y, X родич Y. Зі сказаного випливає, що кількість породжених правилами фактів може бути дуже великим, причому більшість породжених фактів швидше за все ніколи не будуть затребувані.

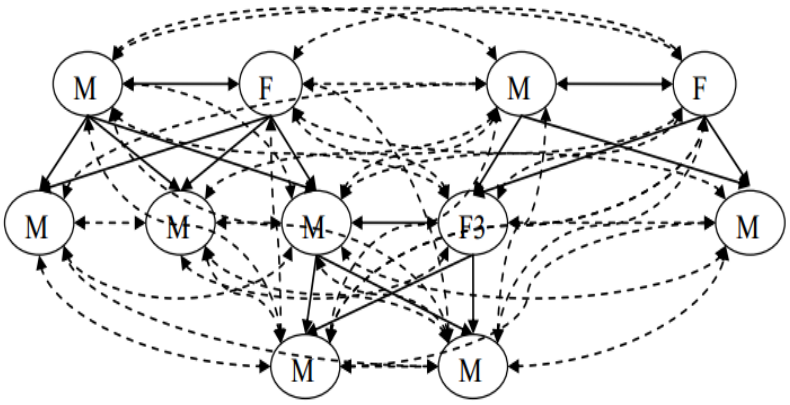

Рисунок 3.5 – Граф родинних відносин з первинними і вторинними фактами

При зворотному логічному висновку цільове твердження зіставляється з фактами і правилами бази знань. Якщо знайдений факт, то мета досягнута, інакше машина виведення починає заглиблюватися в правила. Породження нових фактів в результаті успішного застосування правил на всіх рівнях вкладеності останніх дозволяє значно скорочувати час пошуку на дереві рішень, якщо дерево містить повторювані фрагменти.

Як і при прямому виведенні, результати пошуку можуть використовуватися в наступних запитах. На відміну від прямого виведення

генеруються тільки факти, які були затребувані хоча б один раз, отже, висновок на основі прецедентів тут не буде таким марнотратним з точки зору пам'яті.

Таким чином, використання прецедентів є кращим при зворотному логічному пошуку, оскільки не вимагає додаткових заходів, щодо обмеження зростання числа прецедентів.

3.2 Методи організації пошуку в інтелектуальних системах

3.2.1. Методи забезпечення актуальності прецедентів в мінливих базах знань

Результати логічного висновку можуть стати недійсними (неповними, неактуальними або помилковими) при зміні бази знань, яке полягає у видаленні, появі нових або зміну існуючих фактів. Слід зазначити, що перераховані випадки відповідають, головним чином, моделі знань експертних систем, в яких факти є темпоральними або динамічними.

У реальному світі слід розрізняти первинні факти, які не можуть ні зникати, ні змінюватися, і вторинні факти, які є результатами висновків. Первинний факт може бути видалений лише в разі, якщо виявлено його помилковість. Вторинний факт може стати хибним в разі появи фактів, заперечують його. Первинний факт відповідає деякому події, а вторинний станом. Стани, які є незмінними, будемо називати аксіомами.

Таким чином, безліч фактів F про навколишній світ можна розділити на два підмножини:

$$F = \{S, E\}, \tag{3.4}$$

де S = {s}, E = {e} – множина станів і подій відповідно. Кожне стан st в момент часу t може бути або аксіомою (початковим станом), якщо воно не

може змінюватися, або результатом подій, визначаються за допомогою безлічі правил R = {r}.

$$R: (\{s_{t-1}\}, \{e_t\}) \rightarrow s_t \qquad (3.5)$$

виходячи з станів $\{s_{t-1}\}$ в попередній момент часу і подій $\{e_t\}$,

що відбулися в момент t.

Еволюцію фактів в просторі «стану – події» ілюструє рис.3.6. Прототипом такої моделі бази знань є бухгалтерський облік, в якому стану {s} відповідають залишки коштів на рахунках, матеріалів на складі і ін., Події {e} – первинним документам, що викликає рух за рахунками. Моментальний знімок підмножини станів, упорядкованих певним чином, становить регістр бухгалтерського обліку.

Для того щоб отримати стан якого-небудь рахунки на поточний момент, потрібно або провести всі обчислення за документами від початкового стану s_0 до поточного дня t

$$e_t = f(s_0, e_1, e_2, \ldots, e_{t-1}), \qquad (3.6)$$

або просто взяти дані з регістра.

Але, щоб бути впевненим, що дані в регістрі вірні, необхідно знати, що всі документи проведені.

Таким чином:

1) все різноманіття фактів в базах знань складається з безлічі станів S = {s} (моментальних знімків) і безлічі подій E = {e}, які обумовлюють переходи з одного стану в інший за допомогою правил R = {r};

2) визначення стану об'єкта в довільний момент часу вимагає обчислення або всіх переходів від початкового стану до поточного шляхом обробки безлічі подій про об'єкт, що передують цьому моменту, або всіх

переходів від збереженого стану до поточного моменту;

3) збережений поточний стан об'єкта s_t є поточним до тих пір, поки не відбудеться подія e_{t+1}, що змінює стан даного об'єкта

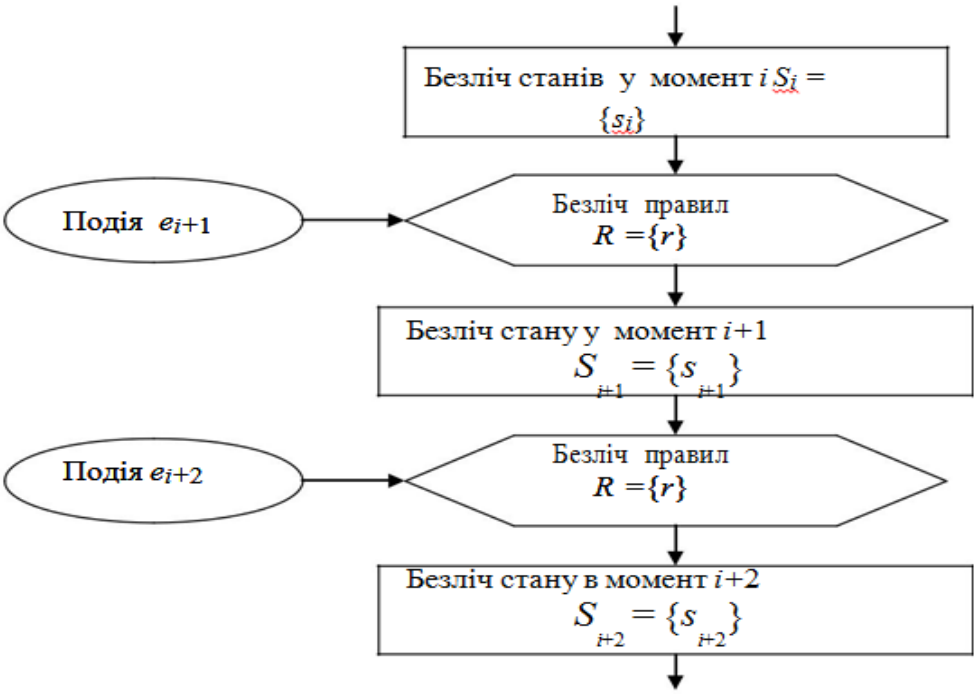

Рисунок 3.6 – Еволюція фактів в просторі «стану – події»

Необхідною умовою застосування такої моделі бази знань є вичіслімость кожного стану s_i як функції від події e_i і стану в попередній момент часу s_i

$$s_i = \theta(e_i, s_{i-1}). \quad (3.7)$$

Нехай стан s_i є результатом не тільки події e_i і стану s_{i-1} але і стану s_{i-2}:

$$si = \theta(ei, si\text{-}1, si\text{-}2). \quad (2.8)$$

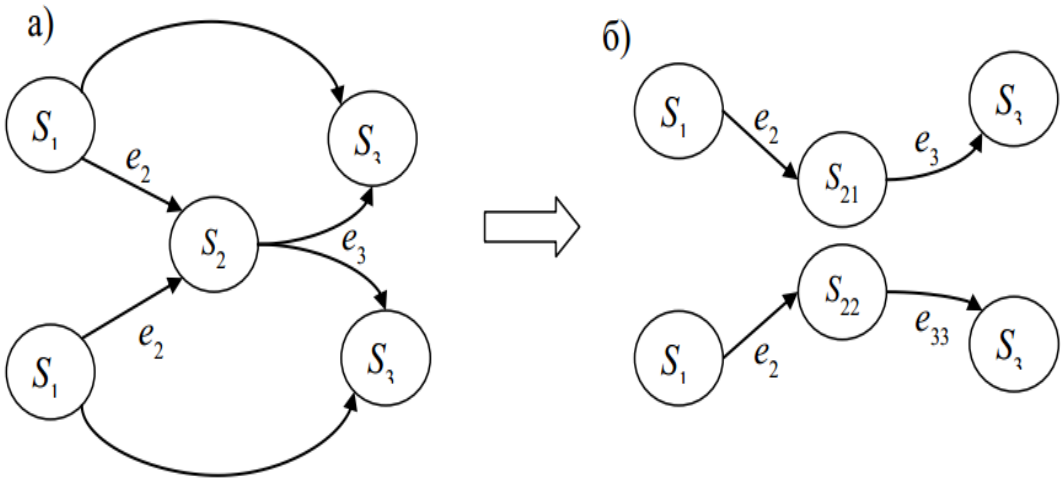

Рисунок 3.7 – Декомпозиція станів

Рисунок 3.7а ілюструє цю ситуацію. Тут подія e_2 призводить до стану s_2 з двох можливих станів s_{11} і s_{12}, а подія e_3 залежно від попереднього стану s_{11} або s_{12} викликає перехід в стан s_{31} і s_{32}. Цілком очевидно, що така неоднозначність зумовлена невдалим кодуванням станів. Досить піддати стан s_2 декомпозиції на два стану s_{21} і s_{22}, як показано на рисунку 3.7б, щоб цю неоднозначність усунути.

Нижче наведена спрощена схема бази знань в просторі «Стану – події», яка припускає створення універсальних правил для поновлення станів при появі нових подій. Визначимо стану і події Пролог-предикатами status і event відповідно: status(<subject>, <predicate>, <object>, <date>), event(<subject>, <event>, <date>).

Аргумент date в предикате status відповідає даті, починаючи з якої даний стан є істинним. Для незмінних станів ця дата може бути будь-який; головне, щоб вона була раніше будь-якої події в базі знань.Визначимо також допустимі переходи з одного стану в інший: jmp(<class>, <event>, <oldstate>, <newstate>).

Предикат jmp визначає клас суб'єктів і події, до яких застосовні переходи зі стану oldstate в стан newstate.

Наприклад, для класу aircraft і стану повітряного руху

(FlightState) допустимими є наступні переходи: jmp(aircraft, engineStart, flightState, parking, taxying).

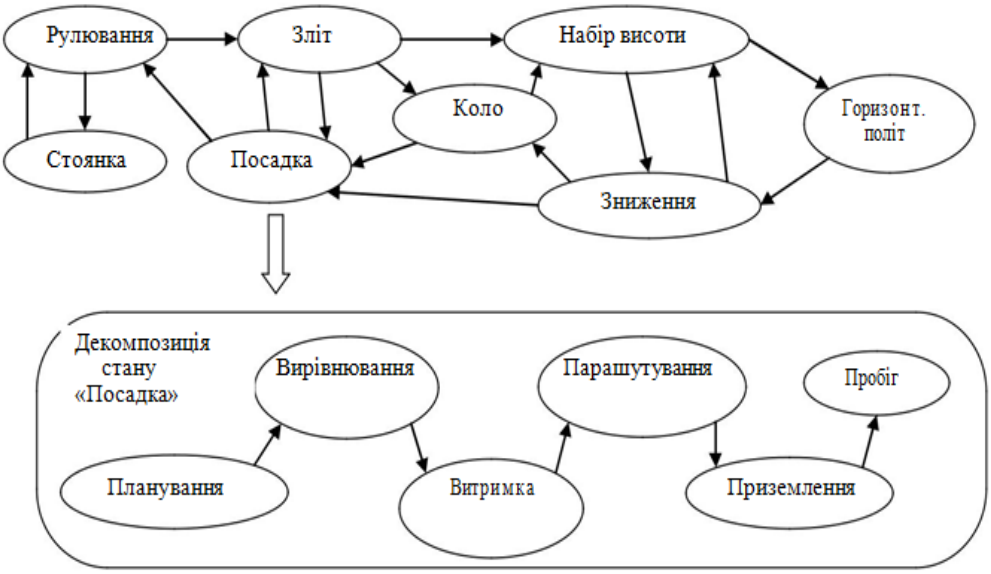

Рисунок 3.8 – Граф станів повітряного судна

Завдання допустимих переходів допомагає систематизувати знання предметної області, як показано на рисунку 3.8, де наведено граф станів повітряного судна, на якому визначені всі можливі стану і всі дозволені переходи з одного стану в інший.

Так, для стану «Горизонтальний політ» єдино допустимим є перехід в стан «Зниження», а для стану «Круг» стану «Посадка» і «Набір висоти». Всі інші переходи є забороненими.

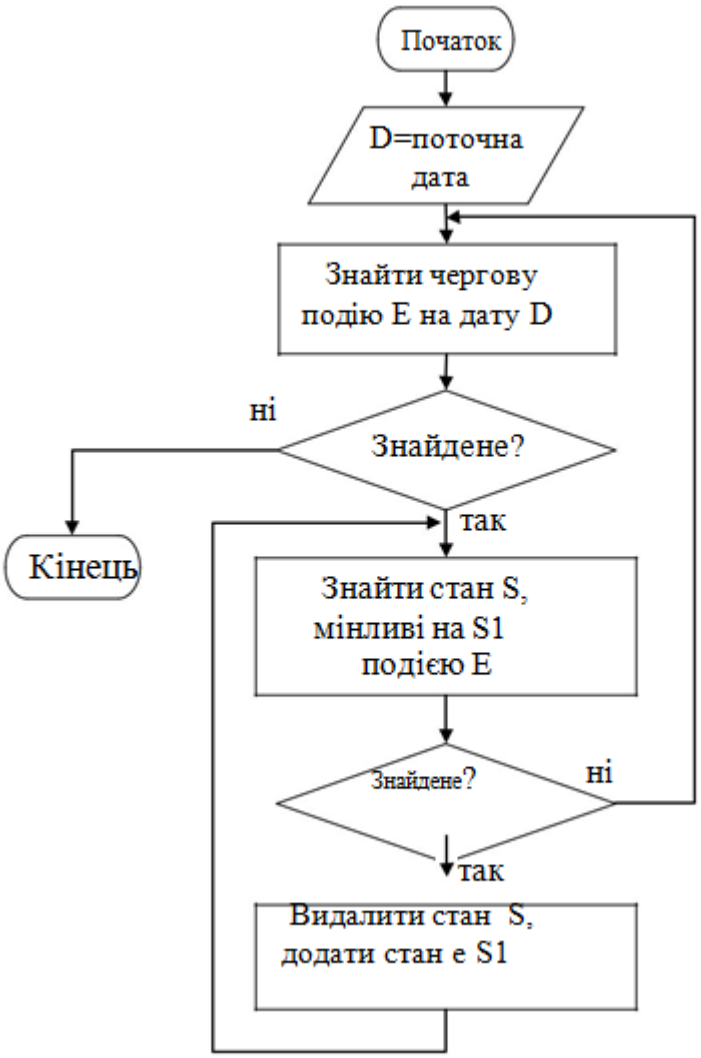

Рисунок 3.9 – Блок-схема алгоритму поновлення станів подіями поточного дня

На рис. 3.9 показана блок-схема алгоритму, що виконує пошук всіх подій на поточну дату і реалізує оновлення всіх станів, керованих цими подіями, тобто для яких є дозволений перехід з початкового стану в наступне.

Правила для поновлення станів виглядають наступним чином:

```
      update_facts(Instance, Event, Date) :-status(Instance, is_a, Class,_),
status(Instance, Predicate, CurrentStatus,_),
      jmp(Class,Event,Predicate,CurrentStatus,                    NewStatus),
retractall(status(Instance,           Predicate,          _,          _)),
assert(status(Instance,Predicate, NewStatus, Date)), fail.
      update_facts(_,_,_). update_all :-
```

```
      get_time(T),stamp_date_time(T, date(Y,M,D,_,_,_,_,_,_),0),
      event(Subject,      Event,      date(Y,M,D)),      update_facts(Subject,
Event,date(Y,M,D)), fail.
      update_all.
```

Приклад 3.3 – Правила для поновлення станів

Завдання мети update_all викликає обробку всіх подій,що відбулися сьогодні і оновлення всіх станів, до яких ці події відносяться.

Таким чином, уявлення фактів в просторі «стану – події» дозволяє, по-перше, легко додавати в базу нові первинні факти (події) в міру їх появи, по-друге, досить просто модифікувати базу вторинних фактів (станів), подальше вилучення яких є такий же простий процедурою, як і звернення до первинних фактами.

Формування бази знань в просторі «стану-події» з використанням прецедентів тягне за собою проблему контролю актуальності прецедентів в умовах мінливості бази фактів. В даній роботі пропонуються три варіанти організації контролю актуальності прецедентів, як показано на рис. 3.10:

1. В момент звернення до прецеденту виконується пошук можливих подій, що змінюють або скасовують даний прецедент (прецеденти, керовані запитами, рис.3.10а).

2. Оновлення всіх прецедентів з певною періодичністю (прецеденти, керовані розкладом, рис.3.10б).

3. У момент появи події виконується пошук і оновлення всіх прецедентів, які належать до учасників події (Прецеденти, керовані подіями рис.3.10в).

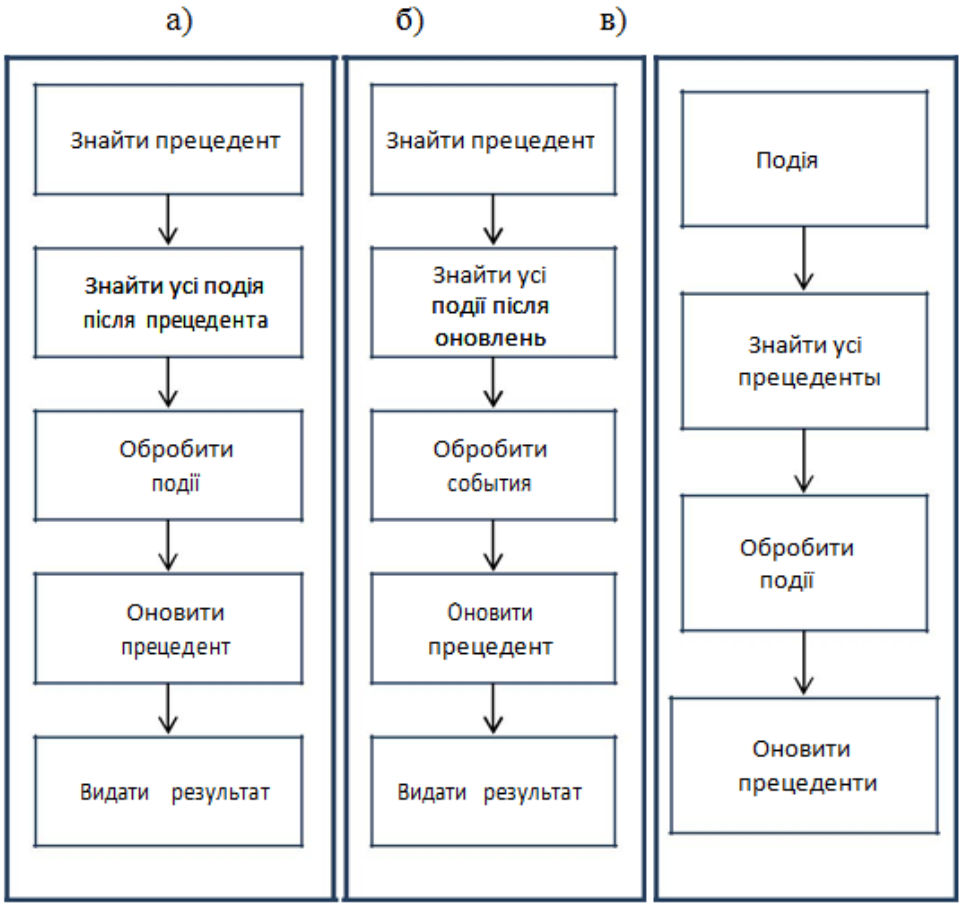

Рисунок 3.10 – Алгоритми управління прецедентами за запитом (а), за розкладом (б) і по настанню події (в)

Перший спосіб гарантує актуальність прецедентів, але вимагає додаткових витрат часу при кожному зверненні до будь-якого прецеденту.

Для цього виконується обробка всіх подій, що сталися з моменту створення прецеденту, тобто замість всіх подій від стану s_0 обробляються тільки частина, від стану s_{t-1}, де t-1 – момент виконання попереднього запиту. Щоб це стало можливим, необхідно разом з прецедентом зберігати час його створення. Другий варіант, управління прецедентами за розкладом має право на існування в тих випадках, коли появою нових фактів, зміною або видаленням існуючих в проміжку між двома оновлень бази прецедентів можна знехтувати.

Перший варіант уповільнює процес вилучення фактів з бази знань,

другий призводить до збільшення часу видалення фактів з бази. Якщо мінливість бази знань невелика, то використання другого варіанта є виправданим, в іншому випадку доцільно застосовувати перший варіант.

У зв'язку з тим, що управління за розкладом не гарантує актуальності прецедентів, в подальшому обмежимося першим і третім способом:

1. Актуальність прецеденту перевіряється при кожному зверненні до нього. В цьому випадку з прецедентом повинні бути зіставлені всі факти, на основі яких він був отриманий. триплет ff прецеденту може бути представлений таким чином: ff (s, p, o, n, L), де n – ідентифікатор прецеденту, L – список ідентифікаторів первинних фактів, пов'язаних з даними прецедентом.

2. При видаленні кожного первинного факту видаляються всі пов'язані з ним прецеденти. Для цього необхідно сканувати списки L у всіх прецедентах. Очевидно, що такий підхід може призвести до великих тимчасових витратах. Рішення може полягати в створенні для кожного i-го первинного факту індексу у вигляді x (i, N_i), де i – ідентифікатор факту, N_i – безліч пов'язаних з i-м фактом прецедентів. У такому випадку при видаленні i-го первинного факту необхідно видалити безліч N_i вторинних фактів.

При першому підході середній час t_1 вилучення прецеденту з тестуванням його актуальності визначається формулою

$$t_1 = ½\ \eta\ |P| + ½\ \eta m|F| = ½\ \eta(|P| + m|F|), \qquad (3.9)$$

де η – час вилучення одного факту, P – безліч прецедентів, m –середнє число первинних фактів, використаних при виведенні прецеденту.

При другому підході середній час t_2 вилучення прецеденту

$$t_2 = ½\ \eta|P|, \qquad (3.10)$$

а середній час t_d видалення прецедентів для кожного видаляється

первинного факту зажадає повного сканування всієї бази прецедентів і складе

$$t_d = \eta|P|. \tag{3.11}$$

Як видно з наведених формул, час звернення до бази прецедентів може істотно відрізнятися від часу вилучення первинного факту, але при збільшенні розміру бази знань час зростає лінійно, а не експоненціально.

Якщо в умові правила присутні виклики інших правил застосування прецедентів дозволяє усунути вкладеність правил і істотно скоротити час обробки правила. Середній час tc створення одного прецеденту

$$t_c = \frac{1}{2}\tau \left|F\right|^k \left|P\right|^n \tag{3.12}$$

де k – середнє число умов правила, уніфікуючих фактами,

n – середнє число умов правила, уніфікуючих консеквента інших правил. Тут має місце ступенева залежність часу виведення вторинного факту від числа умов в правилі.

Таким чином, в мінливих базах знань актуальність прецедентів може контролюватися або при зверненні до кожного прецеденту, або при кожному оновленні бази фактів. В обох випадках має місце поліноміальна складність вилучення прецедентів.

3.2.2. Метод управління контекстом при пошуку в базах знань великої розмірності

Вилучення знань з конкретного документа, створеного на основі відомої онтології, практично нічим не відрізняється від обробки SQL-запитів до БД. Якщо ж конкретний документ не визначений, виникає проблема його пошуку. Пошук за допомогою серверів, орієнтованих на людину (Google,

Yahoo, Yandex та ін.), Марний, оскільки теги HTML і будь–який інший розмітки не індексуються. Останнім часом з'явився ряд сервісів, таких, як SWOOGLE, SWSE і ін., Що дозволяють здійснювати пошук формалізованих даних. Необхідність вилучення документів інтелектуальним агентом, в т.ч. з використанням таких пошукових сервісів робить актуальним завдання автоматичної оцінки релевантності знайдених документів запиту.

Кожен документ визначений на деякому контексті, який повинен бути зрозумілий читачеві документа. Якщо назва документа в повному обсязі визначає контекст, то контекст уточнюється у вступній частині. У лінгвістиці контекст є фрагмент тексту мінус визначається одиниця. Таким чином, контекст ξ (m) для повідомлення m встановлюється як ξ (m) = <C, L, P, I, V> - m, де C – безліч класів об'єктів, L – безліч відносин або предикатів (зв'язків) між об'єктами, P – безліч властивостей об'єктів, I –безліч екземплярів об'єктів, V – безліч значень. Отже, для кожного наступного mi + 1 повідомлення попереднє повідомлення mi включається до складу контексту: ξ (mi + 1) = ξ (mi-1) + mi.

Дане визначення є антропоморфних, орієнтоване на мовну взаємодію та передбачає, що суб'єкт і об'єкт такої взаємодії мають інтелект і можуть ідентифікувати(Встановити) контекст для кожного комунікаційного акту. У разі інформаційного пошуку встановлення контексту може бути простим тільки для локальних БЗ, як, наприклад, це робиться за допомогою мікротеорій в продукті ResearchCyc компанії Cycorp. Автор запиту до БЗ повинен знати, яким чином факти в базі знань групуються в мікротеорії, і експліцитно вказувати мікротеорію в запиті.

Необхідність володіння мікротеоріямі, ідентифікаторами елементів БЗ, а також спеціальним цих слів робить взаємодію з такою БЗ подібним роботі з базою даних, яка доступна тільки для програміста, а не кінцевого користувача.

Таким чином, в даній роботі під контекстом, де U –універсум, ми будемо розуміти безліч понять предметної області D, яке встановлюється між

учасниками інформаційного обміну і дозволяє обмінюватися короткими повідомленнями. Інше призначення контексту – обмеження предметної області, що дозволяє скоротити розмірність задачі пошуку і уникнути суперечливості фактів. Зазвичай контекст встановлюється на початку комунікаційного акту шляхом його короткого опису, яке при необхідності може уточнюватися. Таким чином, для правильної інтерпретації повідомлень всі учасники обміну інформацією повинні володіти контекстом. Візуалізація знань також вимагає використання контексту, як це зроблено, зокрема, в розробленої автором програми Semantic, призначеної для створення БЗ, вилучення та візуалізації знань.

У концепції Семантичної павутини (Semantic Web) витяг знань покладається на інтелектуальні агенти. Агент самостійно відшукує необхідну інформацію, формулюючи при необхідності запити до інших агентам. На відміну від SQL–запитів до БД (SELECT ... FROM ... WHERE ...) запити до БЗ, зокрема, на мові SPARQL не містять конструкції FROM, оскільки запит завжди виконується в межах цілого документа. Пошук же документа повинен виконуватися по контексту запиту. Це означає, що витяг фактів з Семантичної павутини є двофазний процес, показаний на рис 3.11. Оскільки точний збіг контексту документа і контексту запиту є ідеальним випадком, даний процес не завжди завершується успішно і може итеративно повторюватися.

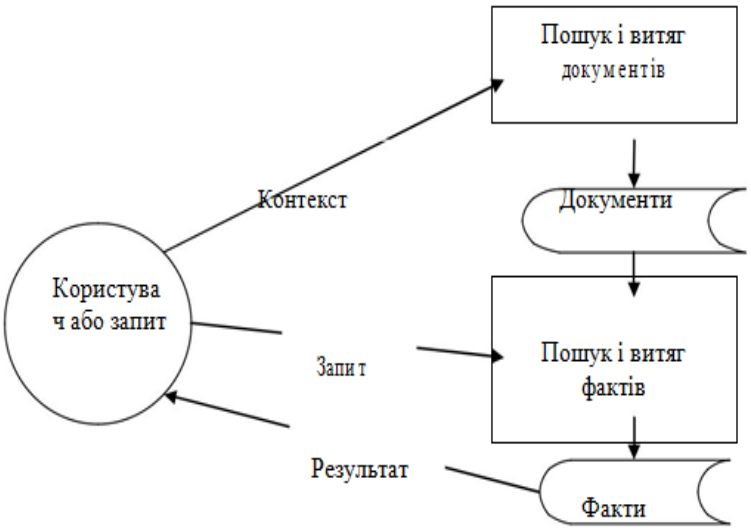

Рисунок 3.11 – Процес вилучення фактів з Семантичної павутини

На рис. 3.12 показані всі можливі поєднання контексту запиту Q і контексту предметної області (домену) D:

A. запит породив набір фактів, жоден з яких не є релевантним (відсутність рішень);

B. знайдена частина релевантних фактів і деяку кількість нерелевантних (рішення є, але воно не повно і суперечливо);

C. знайдені всі релевантні факти і частина нерелевантних (рішення повне, але суперечливе);

D. знайдена частина релевантних фактів і при цьому немає нерелевантних (Рішення несуперечливе, але неповне);

E. знайдені всі релевантні факти і жодного нерелевантного (цільовий стан: рішення повне і несуперечливе).

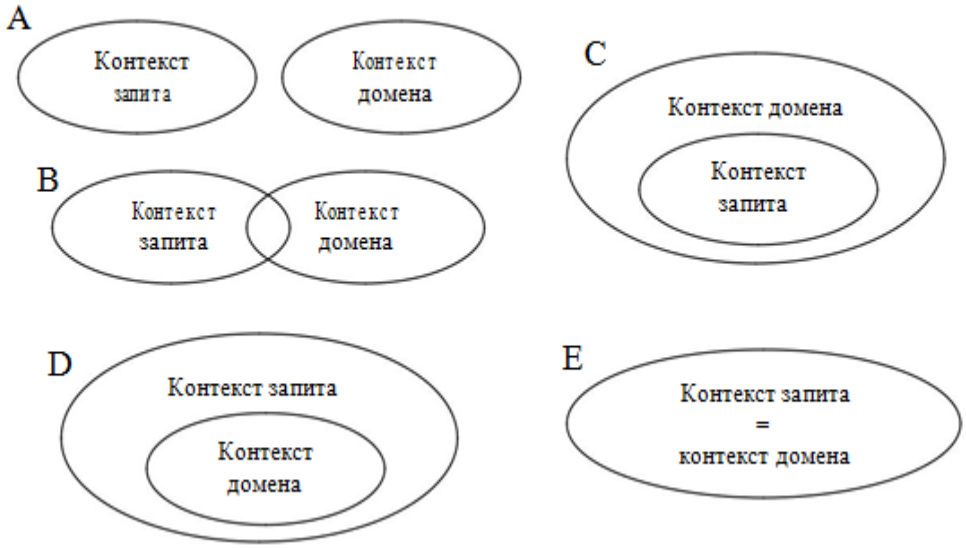

Рисунок 3.12 – Безліч поєднань контексту запиту і контексту домену

У таблиці 3.1 наведені умови появи даних поєднань і ознаки, за допомогою яких можна ідентифікувати кожне з станів. Якщо запит не привів до стану E, його слід змінити.

Обмежимо модифікації запитів двома операціями: x – узагальнення запиту(Розширення контексту) і y – уточнення запиту (звуження контексту).

Таблиця 3.1 – Умови появи поєднань ознак

Ид.	Рішення			Умови
		Повнота	Суперечливість	
A	0	0	0	$Q \cap D = \varnothing$
B	1	0	1	$Q \quad D \neq \varnothing \;\&\; Q \cap D \neq Q \quad D$
C	1	1	1	$Q \subset D$
D	1	0	0	$Q \supset D$
E	1	1	0	$Q = D$

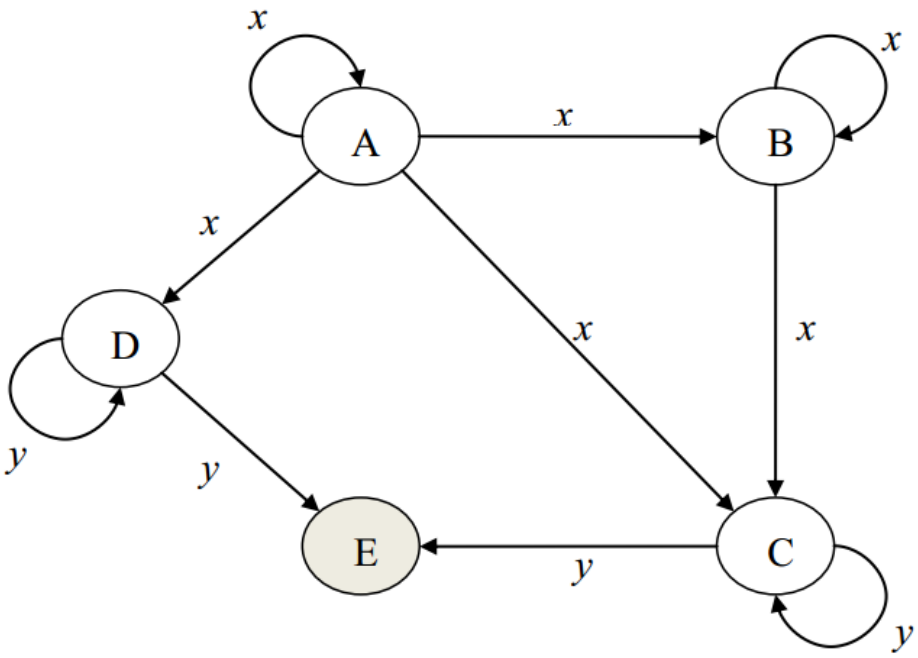

Рисунок 3.13 – Кінцевий автомат пошуку рішення

Граф на рисунку 3.13 відображає кінцевий автомат, в якому дозволеними є тільки переходи, які наближають до цільового стану E (віддаляються від E).

4 ЗАСТОСУВАННЯ РОЗРОБЛЕНИХ МЕТОДІВ В ІНТЕЛЕКТУАЛЬНИХ СИСТЕМАХ ВЕЛИКОЇ РОЗМІРНОСТІ

4.1 Реалізація методів прискорення пошуку рішень в системі технологічного обслуговування касира

Однією з реалізацій інтелектуальних систем великої розмірності є автоматизована система бронювання. Процес бронювання перевезень з робочого місця касира або інтернет-клієнта є багатофазним [94], як показано на рис. 4.1.

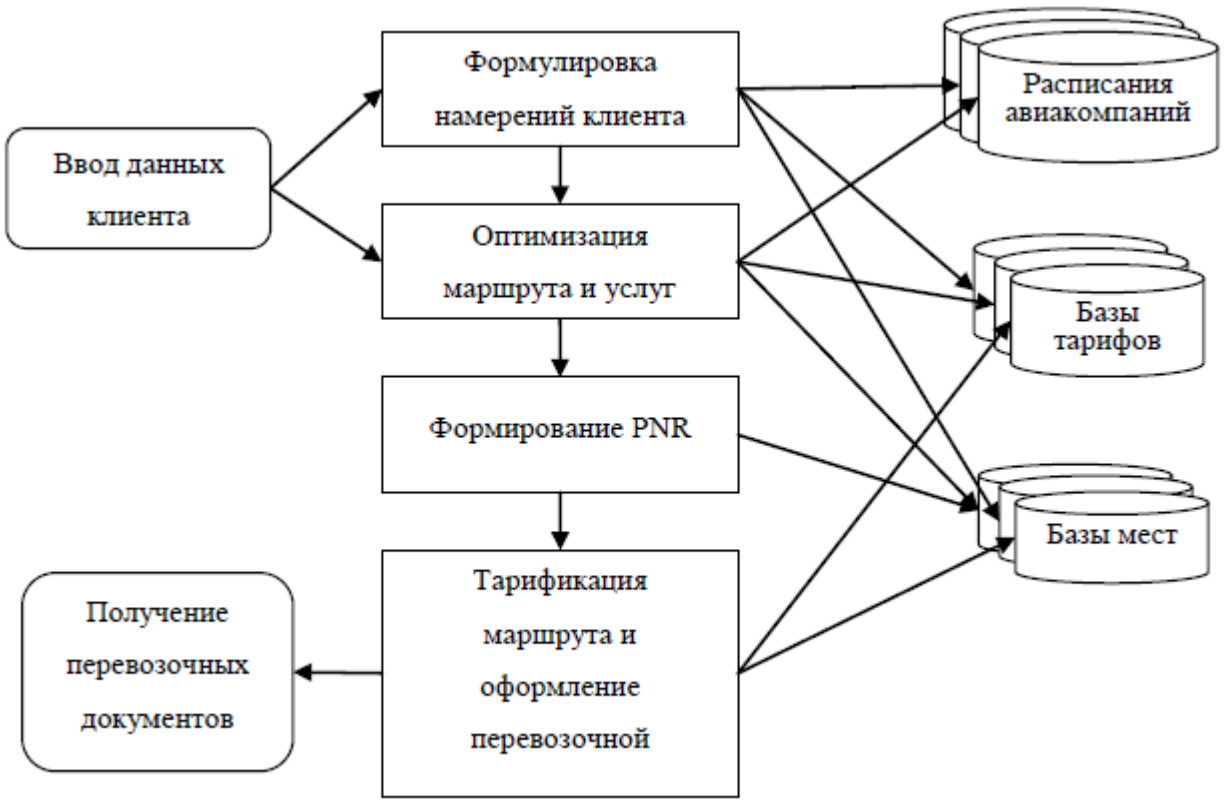

Рисунок 4.1 – Багатофазний процес бронювання перевезень

На кожній фазі потрібно звернення до розподілених баз даних, розміщених на ресурсах компаній Сирена-Тревел, Амадеус, Сэйбр, Габриэл,

Галилео, а пошук в базах даних, записів, що складаються з мільйонів, визначається деревом рішень з глибиною від 1 до 4 (кількість стикувальних маршрутів) і коефіцієнтом галуження близько сотень. Фрагмент дерева рішень для пошуку маршруту показаний на рис 4.2.

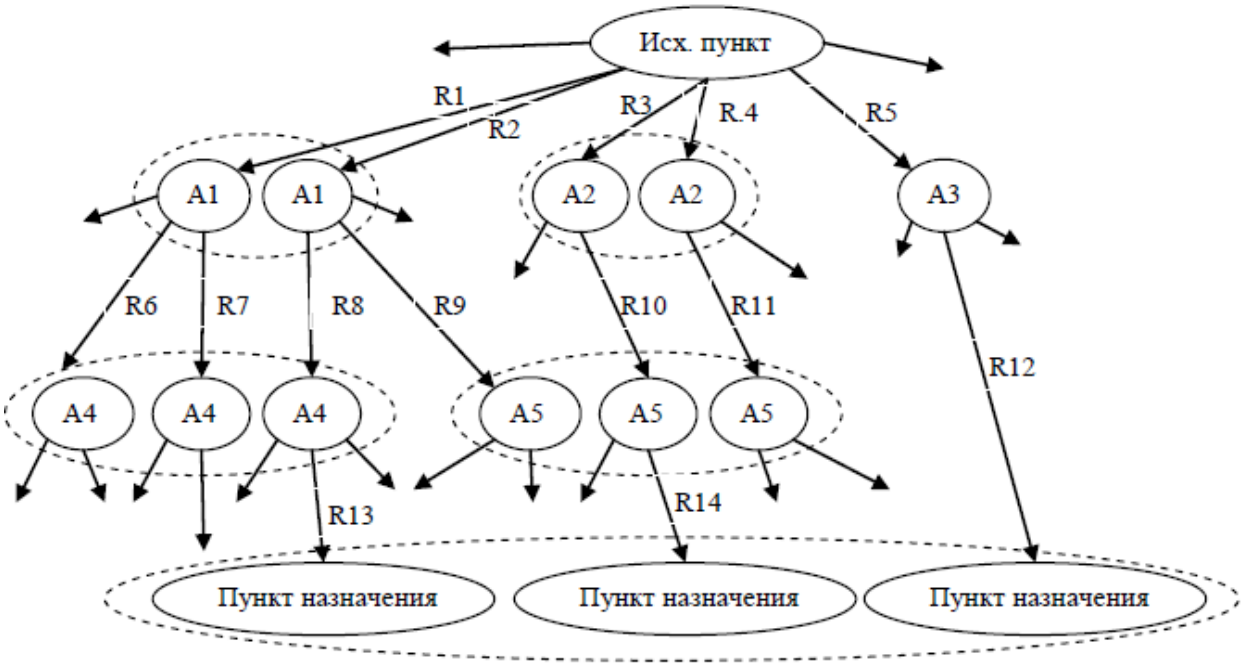

Рисунок 4.2 – Фрагмент дерева маршрутів

На цьому графові вершини A1, A2, . відбивають аеропорти, а дуги R1, R2, . – рейси, що зв'язують їх. Якщо два аеропорти зв'язують декілька рейсів, це призводить до того, що вершини аеропорту прильоту на графі розмножуються (пунктирні еліпси). Пошук маршруту полягає в знаходженні зв'язку між початковим пунктом і пунктом призначення при виконанні безлічі обмежень, в т.ч. мінімально і максимально допустимий час очікування стикувальних рейсів, наявність місць на кожен рейс та ін. Після цього вибрані маршрути сортуються в порядку зростання основної переваги клієнта : часу в дорозі, числа пересадок або ціни квитка.

Метод індексації і попередньої обробки фактів, релевантних умовам запиту, запропонований в роботах [20], [21], у застосуванні до пошуку

маршруту в спрощеному виді можна представити таким чином. Для усіх рейсів виду R=(D, dd, td, A, da, ta), де D – аеропорт вильоту, dd і td – подінь тижні і час вильоту, A – аеропорт прильоту, da і ta – подінь тижні і час прильоту, складаються індекси наступного виду : x(D, {R}); x(dd, {R}); x(td, {R}); x(A, {R}); x(da, {R}); x(ta, {R}).

Кожен з індексів містить списки рейсів для кожного аеропорту вильоту і прильоту, а також рейсів, що виконуються в цей день тижня і зараз доби.

Після з'ясування вимог клієнта система формує декілька запитів Q, проміжних посадок (S_i), що відрізняються числом, Q1 = (D, dd, _, A, _, _); Q2 = (D, dd, _, S1, d1, t1), (S1, d1, t1+, A, _, _); Q3 = (D, dd, _, S1, d1, t1), (S1, d1, t1+, S2, d2, t2), (S2, d2, t2+, A, _, _);

Тут знак підкреслення означає, що значення змінної для запиту несуттєво (анонімна змінна), а t1+ – час, що відрізняється у велику сторону від t1 на величину, що перевищує час пересадки з рейсу на рейс. В цілях простоти викладу матеріалу тут не враховується можливий перехід через добу для стикувальних рейсів.

Попередній відбір фактів для кожного запиту полягає в складанні списків рейсів з кожного індексу і знаходженні перетинів списків. Результатом пошуку в індексах є список рейсів, що задовольняють умовам пошуку, скорочений на 1–3 порядки в порівнянні з початковим. Такий розкид діапазону редукції простору пошуку пояснюється відмінністю ступенів свободи залежно від конкретного запиту. Якщо початковий пункт або пункт призначення – невеликі міста, то коефіцієнт галуження дерева пошуку істотно менший, ніж для авіаційних хабов.

Оскільки залежність переваг клієнтів від характеристик маршрутів не завжди є монотонною, то або пропоновані варіанти можуть не влаштовувати пасажира, або число варіантів може бути занадто великим. В зв'язку з цим доцільно зберігати раніше складені і сплачені маршрути як прецеденти і пропонувати їх наступним клієнтам як готові варіанти.

Такий підхід не лише дозволяє скоротити час пошуку рішень, але

також враховувати додаткові чинники, які не зберігаються в базах знань. Зокрема, недалеко від аеропорту м. Рейк'явіка знаходиться спа-курорт «Блакитна лагуна» на геотермальному озері, і багато пасажирів трансатлантичних рейсів вважають за краще робити там зупинку на декілька годин для відвідування курорту. Збереження історії бронювань у вигляді прецедентів дозволяє в пропонувати такий варіант як пріоритетного не дивлячись на те, що дані про цей курорт в системі відсутні, і надання такого роду сервісів у функціонал системи бронювання перевезень не входить.

Таким чином, застосування розроблених методів і алгоритмів в автоматизованій системі бронювання авіаперевезень забезпечило редукцію простору пошуку і скорочення часу обробки транзакцій.

ВИСНОВКИ

В ході досліджень в рамках магістерської роботи виявлено основне протиріччя в області побудови інтелектуальних систем на продукционной моделі знань, що складається в тому, що вимоги до обсягу і широтою охоплення базами знань понять і закономірностей реального світу неухильно зростають, а існуючі методи логічного пошуку не можуть забезпечити необхідну якість пошуку, обмежуючись його прискоренням на 2-3 порядку. Причина даного протиріччя полягає в тому, що існуючі методи пошуку не враховують багатомірність баз знань або враховують її лише частково.

На основі аналізу досліджень в суміжних областях запропоновано застосувати прецедентний підхід до організації логічного висновку з використанням механізму навичок, що має забезпечити потенційно більше прискорення логічного пошуку в базах знань великої розмірності при більш низьких темпах зростання складності, ніж у відомих алгоритмів. Реалізація даного твердження зажадала розробки концептуальної моделі інтелектуальної системи великої розмірності на продукційної моделі знань, що враховує багатовимірне опис простору пошуку, орієнтоване на використання методів швидкого логічного висновку.

В процесі деталізації концептуальної моделі інтелектуальної системи виявлені системні закономірності в просторі пошуку, що дозволили на основі прецедентного підходу з відтворенням механізму інтелектуальних навичок у вигляді прецедентів редукувати простір пошуку і за рахунок цього скоротити структурну надмірність дерева рішень, в результаті чого розроблений ряд загальносистемних методів, в т.ч .:

а) розроблений метод редукції дерева рішень, що враховує наявність в ньому повторюються гілок, що дозволяє за рахунок заміни ланцюжків міркувань одиничними вузлами скоротити глибину пошуку;

б) сформульовані умови застосовності прецедентів в базах знань

великої розмірності, що враховують особливості алгоритмів прямого і зворотного логічного висновку;

в) розроблені методи організації темпоральних баз знань, які використовують організацію прецедентів в просторі події-стану, а також способи контролю актуальності фактів, що дозволяють забезпечувати їх несуперечливість в базах знань;

г) розроблений метод управління контекстом пошуку в розподілених базах знань великої розмірності, що дозволяє за рахунок використання обмеженого набору операцій знаходити рішення, що володіють повнотою і непротиворечивостью;

У процесі дослідження виявлено структурні особливості баз знань великої розмірності, що дозволили на основі інформаційних ваг головних напрямків пошуку забезпечити скорочення розмірності дерева рішень в стандартному і розширеному доменах на всіх комбінаціях підзапитів.

В ході дослідження підтверджена практична реалізація і ефективність розроблених методів при впровадженні в системи технологічного обслуговування касирів автоматизованої системи бронювання авіаперевезень і систему взаєморозрахунків на повітряному транспорті, що дозволило скоротити час електронного оформлення квитків і час обробки транзакцій по інтерактивній звітності на 12%.

Таким чином, представлені вище результати дослідження дозволяють стверджувати, що сформульована наукова проблема – розробка систем штучного інтелекту на продукційній моделі знань, що використовують більш швидкі порівняно з існуючими методи логічного пошуку – вирішена з урахуванням прийнятих обмежень і припущень, що підтверджується впровадженням результатів і проведеними обчислювальними експериментами.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Сорокина, В. В., and В. А. Сергачева. «Искусственный интеллект: современный этап развития и применение в различных областях.» Аллея науки 2.1 (2019): 952-956.

2. Мински, М. и Паперт, С. Искусственный интеллект / М. Мински, С. Паперт // Univ. of Oregon Press. – 1972.

3. Одинцов, Борис Ефимович. «Порождение нового знания для создания баз знаний интеллектуальных систем.» Вестник ВГУ. Серия: Системный анализ и информационные технологии 1 (2019): 95-101.

4. Колесников, Александр Васильевич, et al. Анализ методов гетерогенного мышления и перспектив их реализации гибридными интеллектуальными многоагентными системами. Вестник Балтийского федерального университета им. И. Канта. Серия: Физико-математические и технические науки 2 (2018).

5. Еремеев, А. П. и Королев, Ю. И. Средства моделирования на основе тем-поральных сетей Петри для интеллектуальных систем поддержки принятия решений / А. П. Еремеев, Ю. И. Королев // Тез. докл. Тринадцатой национальной конференции по искусственному интеллекту КИИ-2012. – Белгород. – 2012. –URL: http://www.raai.org/resurs/papers/kii-2012/doclad/eremeev.doc.

6. Люгер, Дж. Ф. Искусственный интеллект: стратегии и методы решения сложных проблем. Изд. 4-е. / Дж. Ф. Люгер // Издательский дом «Вильямс». – 2003. – С. 864. – ISBN 5-8459-0437-4.

7. Hodges, Brian D. Ones and zeros: Medical education and theory in the age of intelligent machines. Medical Education (2020).

8. Добриборщ, Д. (2019). Адаптивное и робастное управление по выходу в условиях дискретных измерений и внешних возмущений (Doctoral dissertation).

9. Советов, Б. Я. и Яшин, А. И. Однородный математический метод процессов планирования и управления / Б. Я. Советов, А. И. Яшин // Известия СПбГЭТУ ЛЭТИ. – 2012 . №5. – С.31-35.

10. Урумбаева, О. Б., Т. А. Шалаев, and О. М. Шикульская. «КОНЦЕПЦИЯ ИНТЕЛЛЕКТУАЛЬНОГО УПРАВЛЕНИЯ ЭНЕРГОСЕТЬЮ. Инженерно-строительный вестник Прикаспия 3 (33) (2020).

11. Bessmertny, Igor. An Intellectual Agent in Training Systems / I. Bessmertny // Proceedings of The 5th International Symposium on Education and Infor-mation Systems, Technologies and Applications. – Orlando, FL, USA. – 2007. Vol. 1, – C.86-89.

12. Кудрин, Виктор Борисович, and Константин Станиславович Хруцкий. Трехзначная логика и троичная информатика НП Брусенцова: их Аристотелевские основания. Biocosmology–neo-Aristotelism 7.3, 4 (2017).

13. Бессмертный, И.А. Визуализация знаний на основе семантической сети / И.А. Бессмертный // Программирование. –М.: Pleiades Publishing, Ltd. – 2010. –Т. 36. –№ 4. – С. 16-24. – ISSN 0132-3474.

14. Бессмертный, И.А. Метод ускорения логического вывода в продукционной модели знаний / И.А.Бессмертный, Р.С. Катериненко // Программирование. – М.: Pleiades Publishing, Ltd. – 2011. – Т. 42. –№ 4. – С. 76-80. – ISSN 0132-3474.

15. Вишняков, В. А. Развитие интеллектуального управления с использованием облачных технологий. Информатика 2 (2016): 113-120.

16. Бессмертный, И.А. Методы поиска информации в продукционных системах / И.А. Бессмертный // Изв. вузов Приборостроение. – 2011. –№. 6. – С. 56-59.

17. Намиот, Д. Е., et al. Информационные роботы в системах управления предприятием. International Journal of Open Information Technologies 5.4 (2017).

18. Фомичева, Светлана Григорьевна. «Теоретические аспекты квантования баз знаний в мультиагентных системах.» Информационно-

управляющие системы 3 (88) (2017).

19. Бессмертный, И.А. Искусственный интеллект / И.А. Бессмертный // – СПб: СПбГУ ИТМО. – 2010. – 132 с. –URL: http://window.edu.ru/ resource/274/ 69274/files/itmo443.pdf.

20. Ясницкий, Л. Н. «Интеллектуальные системы: учебник.» М.: Лаборатория знаний (2016).

21. Акперов, Г. И., И. Д. Алекперов, and В. В. Храмов. «Интеллектуальные информационные системы в эпоху цифровой экономики: учеб. пособие.»

22. Ming-Hung, Hsu, Ming-Feng, Tsai и Hsin-Hsi, Chen. Query Expansion with ConceptNet and WordNet: An Intrinsic Comparison / Hsu Ming-Hung, Tsai Ming-Feng, Chen.Hsin-His // Proceedings of the Third Asia Information Re-trieval Symposium. – Singapore. – October 16-18, 2016. – С. 1-13.

23. Minsky, Marvin. HAL's Legacy. – 2001's Computer as dream and reality / M. Minsky, [ред.] David G. Stork // MIT. – 2000. – ISBN 0-262-19378-7.

24. Forgy, C. RETE: A fast algorithm for the many pattern/many object pattern match problem / C. Forgy // Artificial Intelligence. – 1982. – Vol. 19. – С. 17-37.

25. Madden, Neil. Optimizing RETE for low-memory, multiagent systems / N. Madden // Proceedings of Game-On– 2013: 4th International Conference on Intelligent Games and Simulation. London. – November. – 2013. – С. 77-81.