МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Державний заклад
«Південноукраїнський національний педагогічний університет
імені К.Д. Ушинського»
Фізико-математичний факультет

Т.Л. Мазурок, В.В. Черних

ЕКСПЕРТНІ СИСТЕМИ

Навчальний посібник
для здобувачів першого(бакалаврського) рівня вищої освіти

Затверджено
Вченою радою ПНПУ
ім. К.Д. Ушинського
Протокол № ___ від _____

Одеса
2021

УДК 378.973+378.14
ББК 32.97я73

Мазурок Т.Л., Черних В.В. Експертні системи: навчальний посібник для здобувачів першого(магістерського) рівня вищої освіти ОПП «Середня освіта (Інформатика Мова та література (англійська))», ОПП «Середня освіта (Інформатика)» спеціальності 014 «Середня освіта (Інформатика)». Одеса: ПНПУ ім. К.Д. Ушинського, 2021. 214 с.

Зміст видання відповідає освітньо-професійній програмі підготовки здобувачів першого (бакалаврського) рівня вищої освіти ОПП «Середня освіта (Інформатика Мова та література (англійська))», «Середня освіта (Інформатика)» спеціальності 014 «Середня освіта (Інформатика)». Навчальний посібник містить теоретичні основи побудови всіх основних компонентів знання-орієнтованих систем, їх використання, практичні завдання з прикладами їх розв’язання. Посібник розрахований на здобувачів вищої освіти, які вивчають навчальні дисципліни, що пов’язані з використанням засобів штучного інтелекту, експертних та знання-орієнтованих систем різного призначення.

Рецензенти:

Д. П. Проскурін, доктор фізико-математичних наук, професор кафедри дослідження операцій Київського національного університету імені Тараса Шевченка;

В. М. Плотніков, доктор технічних наук, професор, завідувач кафедри інформаційних технологій та кібербезпеки Одеської національної академії харчових технологій.

ЗМІСТ

ВСТУП

Еволюція інформаційних технологій та систем значною мірою визначається їх інтелектуалізацією. Інтелектуальні інформаційні технології є однією з найбільш перспективних прикладних галузей інформатики, що швидко розвивається. Ця галузь здійснює суттєвий вплив на всі наукові та технологічні напрями, що пов’язані із використанням комп’ютерної техніки.

Визначення інтелекту, зокрема штучного інтелекту, дозволяє характеризувати його як об’єкт, спосіб та процес. Найбільш поширеним є визначення інтелекту (від лат. intellectus - розуміння, пізнання) як загальна здатність до пізнання та вирішення проблем. Інтелект є вищим засобом вирішення практичних і пізнавальних проблем, чим відрізняється від інстинкту і навичок.

Штучний інтелект (англ. artificial intelligence) – це розділ інформатики, що пов'язаний із вирішенням задач апаратного або програмного моделювання тих видів людської діяльності, які традиційно вважаються інтелектуальними. Система вважається інтелектуальною, якщо в неї реалізовані три базові функції: представлення та обробки знань, функція розсуду, функція спілкування. Найбільший інтерес викликає поняття інтелектуальної системи, як системи, що проявляє здатність до цілеспрямованої поведінки.

Одним з напрямків ефективного використання засобів штучного інтелекту є поява нового наукового напрямку, що сформувався відносно недавно – інтелектуальні системи управління. Інтелектуальні системи управління (ІСУ) – це системи управління, які спроможні до «розуміння» та навчання відносно об’єкту управління, оббурювань, зовнішнього середовища та умов роботи. Основна відмінність інтелектуальних систем – це наявність механізму системної обробки знань. Головна архітектурна особливість, яка відрізняє інтелектуальні системи управління від традиційних – це механізм отримання, зберігання та обробки знань для реалізації своїх функцій.

Існує декілька сучасних інформаційних технологій, що дозволяють створювати такі системи управління: експертні системи, штучні нейронні мережі, нечітка логіка, генетичні алгоритми та ряд інших. Для розробки інтелектуальних систем управління данні методи мають бути об’єднаними з досягненнями сучасної теорії управління. Інтелектуальні технології між собою розрізняють перед усім , що саме лежить в основі концепції інтелектуальності – вміння працювати з формалізованими знаннями людини (експертні системи, нечітка логіка), або властиві людині прийоми навчання та мислення (штучні нейронні мережі і генетичні алгоритми).

Основними цілями інтелектуальних інформаційних технологій є розширення кола задач, що можуть бути вирішеними за допомогою ПК в слабко структурованих предметних галузях та підвищення рівня інтелектуальної інформаційної підтримки сучасного фахівця.

Зазвичай система э інтелектуальною, якщо вона може виконувати наступні базові функції:

1. функція представлення та обробки знань;
2. функція розсуду;
3. функція спілкування.

Застосування комп’ютерних засобів у навчанні має значну історію, що налічує понад шістдесят років. Сучасні технічні досягнення дозволяють створювати адекватне технічне середовище як для індивідуального, так і групового навчання. Втім, більшість сучасних систем навчання не мають розвинутого методичного супроводу, що не дозволяє повною мірою розкрити потенційні технічні можливості ПК, як засобу навчання. Особливої актуальності набувають питання щодо вдосконалення ролі ПК в якості засобу навчання в умовах різних форм змішаного, дистанційного навчання. Крім того, постійно актуальним є стратегічний напрям надання адаптивних властивостей системам комп’ютерного навчання.

Основними недоліками традиційних навчальних систем та середовищ є орієнтація на «середній» рівень навчання, що реалізується в жорстко визначеній системі навчання за заданою стратегією, невдале представлення навчального матеріалу на основі фрагментів інформації, що не підлягають подальшому розбиттю та взаємодія з особою, що навчається на формальному, а не на семантичному рівні.

В якості підходу до подолання вказаних недоліків виникли інтелектуальні навчаючі системи. Розвиток інженерії знань та методів створення експертних систем дозволяє розглядати архітектуру навчаючих систем у вигляді сукупності взаємодіючих експертних систем, кожна з яких оперує зі своїм типом знань. Такий підхід заснований на концепції експертно-навчаючою системи.

З оглядом на затребуваність інтелектуальних технологій для розробки навчаючих систем нового покоління з можливістю автоматичної адаптації навчального матеріалу до потреб та навчальних цілей і з врахуванням особистісних характеристик особи, що навчається, майбутнім вчителям інформатики необхідно мати уявлення про теоретичні та реалізаційні основи створення таких систем. Крім того, навчання роботи зі знання-орієнтованими технологіями входить до програмного матеріалу сучасного шкільного курсу інформатики. Отже, дисципліна «Експертні системи» входить до блоку обов'язкових в освітньо-професійних програмах підготовки здобувачів першого (бакалаврського) рівня «Середня освіта (Інформатика Мова та література (англійська))», ОПП «Середня освіта (Інформатика)» спеціальності 014 «Середня освіта (Інформатика)»».

На основі визначення ключових понять курсу, теоретичний матеріал та практичні завдання спрямовані на формування знань, вмінь та навичок застосування основних компонентів експертних систем для здійснення консультування на основі логічного виведення.

Метою навчальної дисципліни «Експертні системи» є формування теоретичної бази знань студентів з основ структури систем, що засновані на знаннях, теорії проектування експертних систем та практичних вмінь і навичок розробки та використання основних елементів зазначених систем в задачах управління навчанням.

Передумови для вивчення дисципліни: для вивчення навчальної дисципліни «Експертні системи» студенти мають опанувати знання з таких навчальних дисциплін, як «Інформатика», «Програмування», «Операційні системи».

Очікувані програмні результати навчання:

- уміння виділяти, чітко формулювати та знаходити шляхи до розв’язання фахових задач різного ступеня складності, у тому числі, з використанням різних інформаційних ресурсів;
- навички самостійної роботи з різними джерелами інформації. Навички самоосвіти. Здатність проектувати конкретні напрями власного професійного розвитку;
- знання про існування різних носіїв джерел інформації; навички використання раціональних способів пошуку інформації, включаючи засоби електронних інформаційних мереж;
- уміння інтегрувати та систематизувати отримані знання. Володіння навичками грамотного відбору вихідних даних дослідження, складання списку використаних джерел інформації, опису наукових результатів;
- знання про існування різних носіїв джерел інформації; навички використання раціональних способів пошуку інформації, включаючи засоби електронних інформаційних мереж;
- уміння усно та письмового презентувати складну комплексну інформацію в зрозумілій комунікативній формі, використовувати інформаційно-комунікаційні технології.

Очікувані результати навчання дисципліни:

Унаслідок вивчення дисципліни здобувач вищої освіти:

− визначає поняття штучного інтелекту, основні напрямки розвитку систем штучного інтелекту, будову типової інтелектуальної системи;

− визначає поняття знань, пояснює різницю знань від даних, знає типові моделі надання знань;

− визначає основні засоби управління логічним виведенням, стратегії виведення; нечітке виведення;

− знає поняття та сутність архітектури та особливості експертних систем, визначає структуру типової експертної системи.

Здобувач вищої освіти вміє:

– визначати ступінь інтелектуальності задач та обирати найбільш доцільний засіб її вирішення;

– складати семантичні мережі, фрейми, правила продукцій та логічні моделі;

– розробляти засоби управління логічним виведенням та здійснювати його трасування;

– перетворювати нечіткі множини, застосовувати нечіткі знання та нечітке логічне виведення;

– формувати бази знань оболонки експертної системи;

– отримувати висновки на основі застосування демонстраційної експертної системи;

– обирати доцільні методи вилучення знань та застосовувати його в конкретній ситуації;

– використовувати оболонку експертної системи для навчальних цілей.

Унаслідок досягнення результатів навчання здобувачі вищої освіти в контексті змісту навчальної дисципліни мають опанувати такі компетентності:

інтегральна компетентність:

Здатність розв’язувати складні спеціалізовані задачі та практичні проблеми в галузі середньої освіти, що передбачає застосування теорій та методів педагогіки та інформатики і характеризується комплексністю та невизначеністю педагогічних умов організації навчально-виховного процесу загальноосвітньої школи;

загальні компетентності:

здатність доцільно використовувати отримані знання у фаховій діяльності;

здатність генерувати нові ідеї, вирішувати проблеми професійної діяльності на основі абстрактного мислення, аналізу, синтезу та прогнозу;

спеціальні (фахові, предметні) компетентності:

здатність використовувати систематизовані теоретичні та практичні знання наукових фактів, концепцій, теорій, принципів і методів інформатики при вирішенні професійних завдань.

здатність володіти термінологією за фахом та комунікативно-мовленнєвими засобами.

Міждисциплінарні зв’язки: інформатика, програмування, бази даних.

Розділ 1

ЗАГАЛЬНІ УЯВЛЕННЯ ПРО ЕКСПЕРТНІ СИСТЕМИ

1.1 Поняття штучного інтелекту

Найбільш поширеним класом прикладних інтелектуальних систем є експертні системи (ЕС). ЕС – це складні програмні комплекси, що акумулюють знання спеціалістів в конкретних предметних галузях та поширюють цій емпіричний досвід для консультування менш кваліфікаційних користувачів. До характеристик, притаманних ЕС як системам штучного інтелекту можна віднести:

а) компетентність, тобто здатність приймати рішення, адекватні рішенням професіонала-експерта високого рівня;

б) здатність будувати «міркування» на основі символьних перетворень;

в) здатність використовувати як загальні схеми породження рішень, так і окремі;

г) здатність розв’язувати задачі в реальних предметних областях;

д) здатність до інтерпретації формулювання запитів і задач;

е) здатність до аналізу своєї роботи.

Розвиток інженерії знань та методів створення експертних систем визначив архітектуру інтелектуальних навчаючих систем у вигляді сукупності взаємодіючих експертних систем, кожна з яких оперує зі своїм типом знань. Розвиток такого підходу обумовив появу спеціалізованого класу ЕС – експертних навчаючих систем (ЕНС). Під ЕНС розуміють програмну систему, яка реалізує певну педагогічну ціль на основі знань експертів у відповідній предметній галузі, в галузі діагностування знань осіб, що навчаються, та управління навчанням, яка дозволяє демонструвати поведінку на рівні експертів. Взагалі ЕНС підтримує такий чотириланковий цикл навчання:

а) модель керування навчанням пропонує учню чергову навчальну дію, оптимальну з точки зору навчання й моделі учня;

б) якщо це запитання, задача та ін., учень готує відповідь і передає її модулю-експерту в даній предметній області; якщо це інформація, наприклад, розв'язання задачі – вони демонструються за допомогою модуля-експерта;

в) модуль-експерт перевіряє відповідь, порівнюючи її, наприклад, з відповіддю експерта і у випадку неправильної відповіді визначає припущені помилки;

г) за цими помилками модуль аналізу помилок намагається зробити висновок про хибні або неповні знання студента, які породили ці помилки; При цьому використовується і змінюється модель студента, через що результати аналізу враховуються на новому циклі навчання.

Отже, аналіз практичного досвіду використання ЕНС свідчить про те, що поряд із значними перевагами навчання із використанням досвіду експертів-викладачів, отримання логічних висновків для прийняття рішень щодо формування послідовності навчання, розробка та експлуатація ЕНС пов'язано із значними труднощами, що не дозволило, на нашу думку, використати потенційні можливості технологій ЕС у навчанні повною мірою. Тому розглянемо докладніше доцільність використання ЕС для здійснення управління складними системами.

Одним з нових напрямків, що розвиваються за останній час в теорії управління, є теорія інтелектуальних систем (ІС). У відповідності до ІС визначені, як деякі системи, що об'єднані єдиним інформаційним процесом, та виробляють на основі відомостей і знань за наявністю мотивації (цілі) рішення щодо дії і реалізують її раціональним способом. Згідно з структура ІС повинна містити у собі такі елементи (блоки):

а) динамічна ЕС (ДЕС), що складається з бази знань (БЗ), блока експертної оцінки, блока оцінки станів;

б) блок прийняття рішень;

в) блок вироблення управління, що містить блок виконання управління;

г) блок формування цілі;

д) блок, що характеризує вплив зовнішнього середовища на ІС.

Центральне місце у ІС посідає динамічна ЕС. ДЕС – це деяке комплексне утворення, яке спроможне оцінювати стан системи і середовища, зіставляти параметри бажаного і реального результатів дії, приймати рішення і виробляти управління, що сприяє досягненню цілі. Для цього ДЕС повинна мати запас знань і методи розв'язання завдань. Тобто, підхід до управління, що засновано на знаннях, є актуальним, блок ЕС є центральним у структурі більшості ІС, але деякі недоліки стримують широке їх використання на практиці. Серед них, найбільш суттєвими є наступні:

а) такі системи в разі зіткнення із не передбаченою ситуацією або формують повідомлення про помилку, або видають невірні результати;

б) вони не спроможні самонавчатись, так як це робить людина в процесі розв'язання проблем, що постають.

На початку 80-х років в дослідженнях зі штучного інтелекту сформований самостійний напрям, що отримав назву «експертні системи» (ЕС). Основним призначенням ЕС є розробка програмних засобів, що дозволяють при розв'язанні слабко формалізованих задач, отримувати рішення, що є не гіршими ніж рішення, що отримані людиною-експертом за якістю та ефективністю розв'язку. Ес використовуються для розв'язання так званих неформалізованих задач, для яких загальними є наступні властивості:

- задачі не можуть бути заданими в чисельній формі;
- цілі не можна виразити в термінах точно визначеної цільової функції;
- не існує алгоритмічне рішення задачі;
- якщо алгоритмічне рішення існує, то його неможливо використовувати завдяки обмеженості ресурсів (час, пам'ять).

Крім того, неформалізовані задачі характеризуються помилковістю, неповнотою, неоднозначністю та протиріччям як вихідних даних, так і знань щодо розв'язання задачі.

Експертна система – це програмний засіб, що використовує експертні знання для забезпечення високоефективного рішення неформалізованих задач у вузькій предметній галузі. Основу ЕС утворює база знань (БЗ) щодо предметної галузі, яка накопичується в процесі створення та експлуатації ЕС. Накопичення та організація знань – важливіша властивість всіх ЕС.

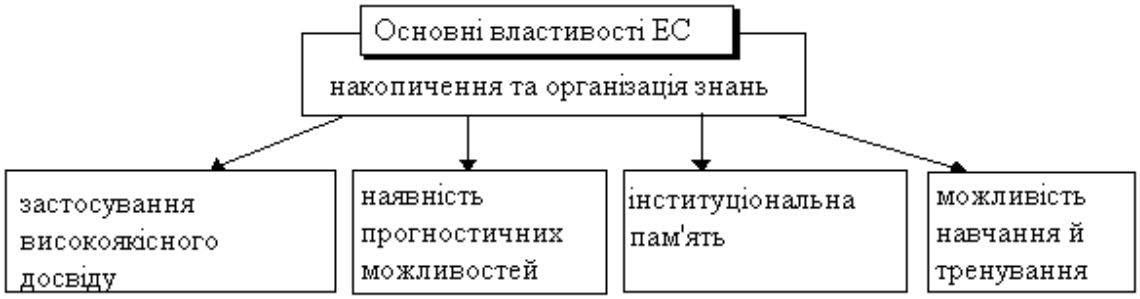

Рисунок 1.1 – Властивості експертних систем

Знання є явними і доступними, що відрізняє ЕС від традиційних програм, і визначає їх основні властивості, такі, як:

1) Застосування для вирішення проблем високоякісного досвіду, який представляє рівень мислення найбільш кваліфікованих експертів в даній області, що веде до рішень творчим, точним і ефективним.

2) Наявність прогностичних можливостей, при яких ЕС видає відповіді не тільки для конкретної ситуації, а й показує, як змінюються ці відповіді в нових ситуаціях, з можливістю докладного пояснення яким чином нова ситуація привела до змін.

3) Забезпечення такої нової якості, як інституціональна пам'ять, за рахунок входить до складу ЕС бази знань, яка розроблена в ході взаємодій з фахівцями організації, і являє собою поточну політику цієї групи людей. Цей набір знань стає зведенням кваліфікованих думок і постійно оновлюється довідником найкращих стратегій і методів, використовуваних персоналом. Провідні фахівці йдуть, але їх досвід залишається.

4) Можливість використання ЕС для навчання і тренування керівних працівників, забезпечуючи нових службовців великим багажем досвіду і

стратегій, за якими можна вивчати рекомендовану політику і методи.

Склад та взаємодія учасників створення та експлуатації експертних систем

Познайомившись з тим, що таке експертні системи і які їхні основні характеристики, спробуємо тепер відповісти на запитання: "Хто бере участь в побудові та експлуатації ЕС?".

До складу основних учасників слід віднести саму експертну систему, експертів, інженерів знань, засоби побудови ЕС і користувачів.

Експертна система - це програмний засіб, що використовує знання експертів, для високоефективного рішення задач у цікавій для користувача предметної області. Вона називається системою, а не просто програмою, так як містить базу знань, вирішувач проблеми і компоненту підтримки. Остання з них допомагає користувачеві взаємодіяти з основною програмою.

Експерт - це людина, здатна ясно висловлювати свої думки і користується репутацією фахівця, який вміє знаходити правильні рішення проблем в конкретній предметній області. Експерт використовує свої прийоми і хитрощі, щоб зробити пошук рішення більш ефективним, і ЕС моделює всі його стратегії.

Інженер знань - людина, як правило, має пізнання в інформатиці і штучному інтелекті і знає, як треба будувати ЕС. Інженер знань опитує експертів, організовує знання, вирішує, яким чином вони повинні бути представлені в ЕС, і може допомогти програмісту в написанні програм.

Засіб побудови ЕС - це програмний засіб, що використовується інженером знань або програмістом для побудови ЕС. Цей інструмент відрізняється від звичайних мов програмування тим, що забезпечує зручні способи представлення складних високорівневих понять.

Користувач - це людина, яка використовує вже побудовану ЕС. Так, користувачем може бути юрист, який використовує її для кваліфікації

конкретного випадку; студент, якому ЕС допомагає вивчати інформатику і т. д. Термін користувач кілька неоднозначний. Зазвичай він позначає кінцевого користувача. Однак з рис.2 випливає, що користувачем може бути:

• творець інструменту, налаштовує засіб побудови ЕС;

• інженер знань, уточнююче існуючі в ЕС знання;

• експерт, який додає в систему нові знання;

• клерк, заносить в систему поточну інформацію.

Важливо розрізняти інструмент, який використовується для побудови ЕС, і саму ЕС. Інструмент побудови ЕС включає як мова, яка використовується для доступу до знань, що містяться в системі, і їх уявлення, так і підтримуючі засоби - програми, які допомагають користувачам взаємодіяти з компонентом експертної системи, що вирішує проблему.

Переваги використання експертних систем

Виникає питання: "Навіщо розробляти експертні системи? І чи не краще звернутися до людського досвіду, як це було в минулому?". Відзначимо лише основні переваги, які дає використання ЕС. Перевагами і позитивними якостями штучної компетенції є:

1) *Її сталість.* Людська компетенція слабшає з часом. Перерва в діяльності людини-експерта може серйозно відбитися на його професійні якості.

2) *Легкість передачі або відтворення*. Передача знань від однієї людини іншій - довгий і дорогий процес. Передача штучної інформації - це простий процес копіювання програми або файлу даних.

3) *Стійкість і відтворюваність результатів*. Експерт-яка людина може приймати в тотожних ситуаціях різні рішення через емоційних чинників. Результати ЕС - стабільні.

4) *Вартість.* Експерти, особливо висококваліфіковані обходяться дуже дорого. ЕС, навпаки, порівняно недорогі. Їх розробка дорога, але вони дешеві в експлуатації.Разом з тим розробка ЕС не дозволяє повністю відмовитися від експерта-людини. Хоча ЕС добре справляється зі своєю роботою, проте в

певних областях людська компетенція явно перевершує штучну. Однак і в цих випадках ЕС може дозволити відмовитися від послуг висококваліфікованого експерта, залишивши експерта середньої кваліфікації, використовуючи при цьому ЕС для посилення і розширення його професійних можливостей.

Особливості побудови та організації експертних систем

Основою будь-якої ЕС є сукупність знань, структурована з метою спрощення процесу прийняття рішення. Для фахівців в області штучного інтелекту термін знання означає інформацію, яка необхідна програмі, щоб вона вела себе "інтелектуально". Ця інформація приймає форму фактів і правил. Факти і правила в ЕС не завжди або істинні, або помилкові. Іноді існує деяка ступінь непевності в достовірності факту або точності правила. Якщо це сумнів виражено явно, то воно називається "коефіцієнтом довіри".Коефіцієнт довіри - це число, яке означає ймовірність або ступінь впевненості, з якою можна вважати даний факт або правило достовірним або справедливим.Багато правила ЕС є евристиками, тобто емпіричними правилами або спрощеннями, які ефективно обмежують пошук рішення. ЕС використовують евристики, так як завдання, які вона вирішує, важкі, не до кінця зрозумілі, не піддаються строгому математичному аналізу або алгоритмічного вирішення. Алгоритмічний метод гарантує коректне або оптимальне рішення задачі, тоді як евристичний метод дає прийнятне рішення в більшості випадків.

Знання в ЕС організовані так, щоб знання про предметну область відокремити від інших типів знань системи, таких як загальні знання про те, як вирішувати завдання або знання про те, як взаємодіяти з користувачем. Виділені знання про предметну область називаються базою знань, тоді як загальні знання про знаходження рішень задач називаються механізмом виведення. Програмні засоби, які працюють зі знаннями, організованими таким чином, називаються системами, заснованими на знаннях.БЗ містить

факти (дані) і правила (або інші уявлення знань), що використовують ці факти як основу для прийняття рішень. Механізм висновку містить:• інтерпретатор, що визначає як застосовувати правила для виведення нових знань на основі інформації, що зберігається в БЗ;• диспетчер, який встановлює порядок застосування цих правил.Такі ЕС отримали назву статичних ЕС і мають структуру, аналогічну. Ці ЕС використовуються в тих додатках, де можна не враховувати зміни навколишнього світу за час виконання завдання.Однак існує більш високий клас додатків, де потрібно враховувати динаміку зміни навколишнього світу за час виконання програми. Такі експертні системи отримали назву динамічних ЕС і їх узагальнена структура матиме вигляд, наведений на рис.4.У порівнянні зі статичної ЕС в динамічну вводиться ще два компоненти:• підсистема моделювання зовнішнього світу;• підсистема сполучення з зовнішнім світом. Динамічні ЕС здійснює зв'язку з зовнішнім світом через систему контролерів і датчиків. Крім того компоненти БЗ і механізму виведення істотно змінюються, щоб відбити тимчасову логіку відбуваються в реальному світі подій.До розряду таких динамічних середовищ розробки ЕС відноситься сімейство програмних продуктів фірми Gensym Corp. (США). Один з таких продуктів система G2 - базовий програмний продукт, що представляє собою графічну, об'єктно-орієнтоване середовище для побудови і супроводу експертних систем реального часу, призначених для моніторингу, діагностики, оптимізації, планування і управління динамічним процесом.

Основні режими роботи експертних систем

В роботі ЕС можна виділити два основні режими: режим придбання знань і режим рішення задачі (режим консультації або режим використання). У режимі придбання знань спілкування з ЕС здійснює експерт (за допомогою інженера знань).

Використовуючи компонент придбання знань, експерт описує проблемну область у вигляді сукупності фактів і правил. Іншими словами, "наповнює"

ЕС знаннями, які дозволяють їй самостійно вирішувати завдання з проблемної області.

Відзначимо, що цього режиму при традиційному підході до програмування відповідають етапи: алгоритмізації, програмування і налагодження, виконувані програмістом. Таким чином, на відміну від традиційного підходу в разі ЕС розробку програм здійснює не програміст, а експерт, який не володіє програмуванням.

У режимі консультацій спілкування з ЕС здійснює кінцевий користувач, якого цікавить результат і (або) спосіб його отримання. Необхідно відзначити, що в залежності від призначення ЕС користувач може:

− не бути фахівцем в даній галузі, і в цьому випадку він звертається до ЕС за результатом, який не вміє отримати сам;

− бути фахівцем, і в цьому випадку він звертається до ЕС з метою прискорення отримання результату, покладаючи на ЕС рутинну роботу.

Слід зазначити, що на відміну від традиційних програм ЕС при рішенні завдання не тільки виконують запропоновану алгоритмом послідовність операцій, але і сама попередньо формує її.

Добре побудована ЕС має можливість самонавчатися на розв'язуваних задачах, поповнюючи автоматично свою БЗ результатами отриманих висновків і рішень.

Відмінність експертних систем від традиційних програм

Особливості ЕС, що відрізняють їх від звичайних програм, полягають в тому, що вони повинні володіти:

1. Компетентністю, а саме:

− досягати експертного рівня рішень (тобто в конкретній предметній області мати той же рівень професіоналізму, що й експерти-люди).

− бути вмілої (тобто застосовувати знання ефективно і швидко, уникаючи, як і люди, непотрібних обчислень).

− мати адекватну робастної (тобто здатність лише поступово знижувати

якість роботи у міру наближення до границь діапазону компетентності або припустимої надійності даних).

2. Можливістю до символьних міркувань, а саме:

− представляти знання в символьному вигляді

− переформулювати символьні знання. На жаргоні штучного інтелекту символ - це рядок знаків, стосується вмісту деякого поняття. Символи об'єднують, щоб висловити відносини між ними. Коли відносини представлені в ЕС вони називаються символьними структурами.

3. Глибиною, а саме:

− працювати в предметній області, що містить важкі завдання;

− використовувати складні правила (тобто використовувати або складні конструкції правил, або велику їх кількість).

4. Самосвідомістю, а саме:

− дослідити свої міркування (тобто перевіряти їх правильність)

− пояснювати свої дії.

Існує ще одна важлива відмінність ЕС. Якщо звичайні програми розробляються так, щоб кожен раз породжувати правильний результат, то ЕС розроблені з тим, щоб вести себе як експерти. Вони, як правило, дають правильні відповіді, але іноді, як і люди, здатні помилятися.

Традиційні програми для вирішення складних завдань, теж можуть робити помилки. Але їх дуже важко виправити, оскільки алгоритми, що лежать в їх основі, явно в них не сформульовані. Отже, помилки нелегко знайти і виправити. ЕС, подібно людям, мають потенційну можливість вчитися на своїх помилках.

Технологія розробки експертних систем

Технологія їх розробки ЕС, включає в себе шість етапів: етапи ідентифікації, концептуалізації, формалізації, виконання, тестування, дослідної експлуатації. Розглянемо більш докладно послідовності дій, які необхідно виконати на кожному з етапів.

1) На етапі ідентифікації необхідно виконати наступні дії:

− визначити задачі, що виникають і цілі розробки,

− визначити експертів і тип користувачів.

2) На етапі концептуалізації:

− проводиться змістовний аналіз предметної області,

− виділяються основні поняття і їх взаємозв'язки,

− визначаються методи розв'язання задач.

3) На етапі формалізації:

− вибираються програмні засоби розробки ЕС,

− визначаються способи подання всіх видів знань,

− формалізуються основні поняття.

4) На етапі виконання (найбільш важливому і трудомісткий) здійснюється наповнення експертом БЗ, при якому процес придбання знань розділяють:

− на "витяг" знань з експерта,

− на організацію знань, що забезпечує ефективну роботу ЕС,

− на подання знань у вигляді, зрозумілому для ЕС.

Процес придбання знань здійснюється інженером по знаннях на основі діяльності експерта.

5) На етапі тестування експерт і інженер по знаннях з використанням діалогових і пояснювальних засобів перевіряють компетентність ЕС. Процес тестування триває до тих пір, поки експерт не вирішить, що система досягла необхідного рівня компетентності.

6) На етапі дослідної експлуатації перевіряється придатність ЕС для кінцевих користувачів. За результатами цього етапу можлива істотна модернізація ЕС.

Процес створення ЕС не зводиться до суворої послідовності цих етапів, так як в ході розробки доводиться неодноразово повертатися на більш ранні етапи і переглядати прийняті там рішення.

1.1.1 Лабораторна робота №1. Визначення інтелектуальної задачі

МЕТА: навчити відрізняти інтелектуальні завдання від не інтелектуальних.

ЗАДАЧІ: ознайомитися з поняттями «інтелект», «алгоритм» «інтелектуальна задача»; розглянути приклади інтелектуальних задач; порівняти інтелектуальні та не інтелектуальні задачі.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Інтелект — якість психіки, що складається із здатності адаптуватися до нових ситуацій, здатності до навчання на основі досвіду, розуміння і застосування абстрактних концепцій і використання своїх знань для управління навколишнім середовищем; Загальна здатність до пізнання і вирішення труднощів, яка об'єднує всі пізнавальні здібності людини: відчуття, сприйняття, пам'ять, уявлення, мислення, уява; здатність мозку вирішувати (інтелектуальні) задачі шляхом надбання, запам'ятовування і цілеспрямованого перетворення знань в процесі навчання на досвід й адаптації до різноманітних обставин.

У цьому визначенні під терміном «знання» мається на увазі не тільки та інформацію, яка надходить у мозок через органи почуттів.

Такого типу знання, отримані через органи чуття надзвичайно важливі, але недостатні для інтелектуальної діяльності. Справа в тому, що об'єкти оточуючого нас середовища мають властивість не тільки впливати на органи почуттів, але і знаходитися один з одним у певних відносинах. Ясно, що для того, щоб здійснювати в навколишньому середовищі інтелектуальну діяльність (або хоча б просто існувати), необхідно мати в системі знань модель цього світу. У цій інформаційній моделі навколишнього середовища реальні об'єкти, їх властивості і відносини між ними не тільки відображаються і запам'ятовуються, але й, як це зазначено в даному визначенні інтелекту, можуть подумки «цілеспрямовано перетворюватися». При цьому важливо те, що формування моделі зовнішнього середовища

відбувається «у процесі навчання на досвіді й адаптації до різноманітних зовнішнім обставинам».

Для пояснення відмінності інтелектуальної задачі від 23е інтелектуальної, необхідно розглянути термін «алгоритм»

Алгоритм — послідовність, система, набір систематизованих правил виконання процесу, який обов'язково призводить до вирішення певного класу задач після виконання кінцевого числа операцій.

Що ж стосується задач, алгоритми рішення яких уже встановлені, то, як зазначає відомий фахівець у галузі штучного інтелекту М. Мінський, «надмірно приписувати їм такі містичні властивості, як «інтелектуальність». Справді, після того, як такий алгоритм уже знайдений, процес вирішення відповідних завдань стає таким, що його можуть в точності виконати людина, обчислювальна машина (належним чином запрограмована) або робот, що не мають ні найменшого уявлення про сутність самої задачі. Потрібно тільки, щоб особа, що розв'язує задачу, була здатна виконувати ті елементарні операції, з яких складається процес, і, крім того, щоб вона педантично і акуратно керувалася запропонованим алгоритмом. Така особа, діючи, як кажуть у таких випадках, чисто машинально, може успішно вирішувати будь-яку задачу розглянутого типу.

Таким чином, справедливо стверджувати, що інтелектуальна задача — задача, розв'язання якої неможливо з використанням стандартних методів розв'язання, алгоритмізації. Так до інтелектуальних задач відносяться доведення теорем, розпізнавання образів, гра в шахи.

У свою чергу не інтелектуальна задача — задача, що вимагає для вирішення готових алгоритмів; прикладом не інтелектуальної задачі може бути будь-яка обчислювальна задача (рішення системи рівнянь, переклад чисел з однієї системи числення в іншу і т.д.)

Питання для самоконтролю:

1. Що називають «інтелектом»?

2. Що називають «інтелектуальним завданням»?

3. Що називають «алгоритм»?

4. Які існують види інтелектуальних завдань?

5. Чи можливий перехід завдання з розряду «інтелектуальних» в розряд «не інтелектуальних»? Наведіть приклади.

Завдання для самостійного виконання:

Визначте, які з наведених нижче завдань є інтелектуальними. Відповідь обґрунтуйте.

1. Прогнозування погоди.

2. Визначення несправності автомобіля в разі якщо автомобіль не заводиться.

3. Визначення стратегії гоночної яхти у регаті.

4. Формування навчального плану університету; розподіл навантаження викладачів.

5. Автоматизований переклад тексту.

6. Розрахунок оплати комунальних послуг.

7. Визначення раціону домашніх тварин.

8. Побудова графіків функцій.

9. Аналіз концентрації шкідливих речовин в повітрі.

10. Розрахунок траєкторії руху космічного тіла.

11. Зарахування абітурієнтів до ЗВО.

12. Постановка діагнозу пацієнта.

13. Визначення знака зодіаку.

14. Визначення відстані між планетами.

15. Вибір оптимальної моделі мобільного телефону для покупця.

16. Форма звіту: Заповнення наведеної таблиці.

Інтелектуальні задачі	**Не інтелектуальні задачі**

1.2 Основні засоби управління логічним виведенням

Висновок на знаннях

Найбільше поширення одержала продуційна модель подання знань. При її використанні база знань складається з набору правил, а програма, що управляє перебором правил, називається машиною висновку.

Визначення 1.8

Машина висновку (інтерпретатор правил) — це програма, що імітує логічний висновок експерта, що користується даною продуційною базою знань для інтерпретації даних, що надійшли в систему.

Звичайно вона виконує дві функції:

- перегляд існуючих даних (фактів) з робочої пам'яті (бази даних) і правил з бази знань і додавання (у міру можливості) у робочу пам'ять нових фактів;

- визначення порядку перегляду й застосування правил. Цей механізм управляє процесом консультації, зберігаючи для користувача інформацію про отримані висновки, і запитує в нього інформацію, коли для спрацьовування чергового правила в робочій пам'яті виявляється недостатньо даних.

У переважній більшості систем, заснованих на знаннях, механізм висновку являє собою невелику по об'єму програму й включає два компоненти – один реалізує безпосередньо висновок, інший – управляє цим процесом.

Дія компонента висновку засновано на застосуванні правила, називаного modus ponens: "Якщо відомо, що щиро твердження А, і існує правило виду "ЯКЩО А, ТО В", тоді твердження В також істинно".

Таким чином, правила спрацьовують, коли перебувають факти, що задовольняють їхній лівої частини: якщо щиро посилку, то повинне бути істинно й висновок.

Компонент висновку повинен функціонувати навіть при недоліку інформації. Отримане рішення може й не бути точним, однак система не

повинна зупинятися через те, що відсутня яка-небудь частина вхідної інформації.

Керуючий компонент визначає порядок застосування правив і виконує чотири функції:

Зіставлення— зразок правила зіставляється з наявними фактами.

Вибір — якщо в конкретній ситуації можуть бути застосовані відразу кілька

правил, то з них вибирається одне, найбільш підходяще за заданим

критерієм (дозвіл конфлікту).

Спрацьовування — якщо зразок правила при зіставленні збігся з будь-якими фактами з робочої пам'яті, те правило спрацьовує.

Дія — робоча пам'ять піддається зміні шляхом додавання в неї висновку правила, що спрацювало. Якщо в правій частині правила втримується вказівка на яку-небудь дію, то воно виконується (як, наприклад, у системах забезпечення безпеки інформації).

Інтерпретатор продукцій працює циклічно. У кожному циклі він переглядає всі правила, щоб виявити тих, посилки яких збігаються з відомими на даний момент фактами з робочої пам'яті. Після вибору правило спрацьовує, його висновок заноситься в робочу пам'ять, і потім цикл повторюється спочатку.

В одному циклі може спрацювати тільки одне правило. Якщо кілька правил успішно зіставлені з фактами, то інтерпретатор робить вибір за певним критерієм єдиного правила, що спрацьовує в даному циклі. Цикл роботи інтерпретатора схематично представлений на рис.1.2.

Рисунок 1.2 - Цикл роботи інтерпретатора

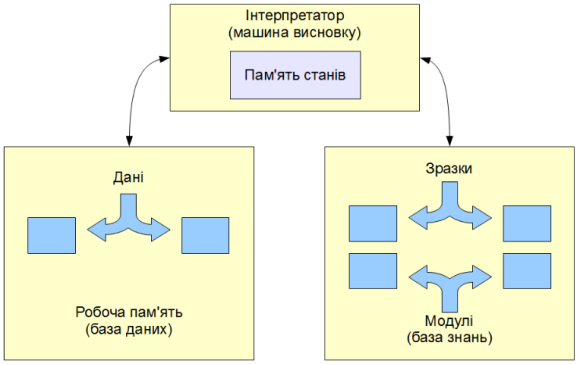

Рисунок 1.3 - Схема функціонування інтерпретатора

Інформація з робочої пам'яті послідовно зіставляється з посилками правил для виявлення успішного зіставлення. Сукупність відібраних правил становить так називану конфліктну множину.

Для дозволу конфлікту інтерпретатор має критерій, за допомогою якого він вибирає єдине правило, після чого воно спрацьовує. Це виражається в занесенні фактів, що утворять висновок правила, у робочу пам'ять або в зміні критерію вибору конфліктуючих правил. Якщо ж на закінчення правила входить назва якої-небудь дії, то воно виконується.

Робота машини висновку залежить тільки від стану робочої пам'яті й від складу бази знань. На практиці звичайно враховується історія роботи, тобто поводження механізму висновку в попередніх циклах. Інформація про поводження механізму висновку запам'ятовується в пам'яті станів (рис.1.4).

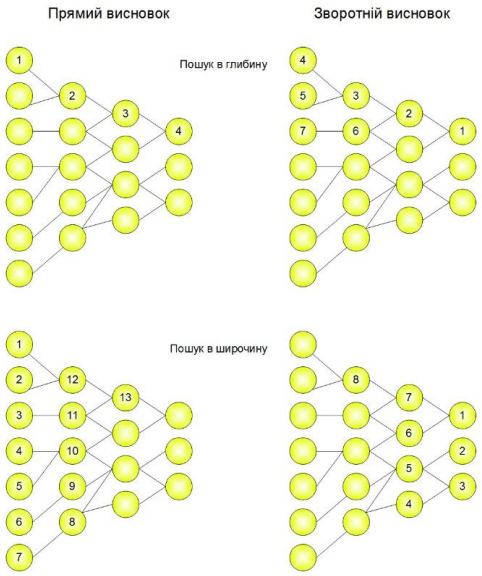

Рисунок 1.4 – Стратегій виведення

Звичайно пам'ять станів містить протокол системи.

У системах із прямим висновком по відомих фактах відшукується висновок, що із цих фактів треба (див. рис.3, ліва частина). Якщо такий висновок вдається знайти, то воно заноситься в робочу пам'ять. Прямий висновок часто називають висновком, керованим даними, або висновком, керованим антецедентами.

Існують системи, у яких висновок ґрунтується на сполученні згаданих вище методів - зворотного й обмеженого прямого. Такий комбінований метод одержав назву циклічного.

Нехай є фрагмент бази знань із двох правил:

- П1: Якщо "відпочинок - влітку" й "людина -активна", то "їхати в гори".

- П2: Якщо "любить сонце", те "відпочинок влітку".

Припустимо, у систему надійшли факти - "людина активна" й "любить сонце".

ПРЯМИЙ ВИСНОВОК— виходячи з фактичних даних, одержати рекомендацію.

- 1-й прохід.

•Крок 1.

Пробуємо П1, не працює (не вистачає даних "відпочинок - влітку").

•Крок 2.

Пробуємо П2, працює, у базу надходить факт "відпочинок - влітку".

- 2-й прохід.

• Крок 3.

Пробуємо П1, працює, активізується мета "їхати в гори", що і виступає як рада, що дає ЕС.

ЗВОРОТНИЙ ВИСНОВОК— підтвердити обрану мету за допомогою наявних правил і даних.

- 1-й прохід.

• Крок 1.

Ціль — "їхати в гори": пробуємо

П1— даних "відпочинок — влітку" ні, вони стають новою метою й шукається правило, де вона в лівої частини.

•Крок 2.

Ціль "відпочинок — улітку": правило

П2 підтверджує мета й активізує її.

- 2-й прохід.

•Крок 3. Пробуємо П1, підтверджується шукана мета.

Методи пошуку в глибину й завширшки

У системах, база знань яких нараховує сотні правил, бажаним є використання стратегії керування висновком, що дозволяє мінімізувати час пошуку рішення й тим самим підвищити ефективність висновку. До числа таких стратегій ставляться: пошук у глибину, пошук завширшки, розбивку на підзадачі й альфа-бета-алгоритм.

При пошуку в глибину в якості чергової підцілі вибирається та, котра відповідає наступний, більше детальному рівню опису задачі. Наприклад, що діагностує система, зробивши на основі відомих симптомів припущення про наявність певного захворювання, буде продовжувати запитувати

уточнювальні ознаки й симптоми цієї хвороби доти, поки повністю не спростує висунуту гіпотезу. При пошуку завширшки, навпроти, система спочатку проаналізує всі симптоми, що перебувають на одному рівні простору станів, навіть якщо вони ставляться до різних захворювань, і лише потім перейде до симптомів наступного рівня детальності. Розбивка на підзадачі має на увазі виділення підзадач, рішення яких розглядається як досягнення проміжних цілей на шляху до кінцевої мети.

Прикладом, що підтверджує ефективність розбивки на підзадачі, є пошук несправностей у комп'ютері - спочатку виявляється підсистема, що відмовила (живлення, пам'ять і т.д.), що значно звужує простір пошуку. Якщо вдається правильно зрозуміти сутність задачі й оптимально розбити її на систему ієрархічно зв'язаних целей-підцілей, то можна домогтися того, що шлях до її рішення в просторі пошуку буде мінімальний.

Альфа-бета-алгоритм дозволяє зменшити простір станів шляхом видалення галузей, не перспективних для успішного пошуку. Тому проглядаються тільки ті вершини, у які можна потрапити в результаті наступного кроку, після чого безперспективні напрямки виключаються. Альфа-бета-алгоритм знайшов широке застосування в основному в системах, орієнтованих на різні ігри, наприклад, у шахових програмах.

Об'єктно-орієнтований підхід, об'єднаний із правилами

Найбільш розвиненим способом подання знань в ЕС є об'єктно-орієнтований підхід, що є розвитком фреймового подання. У його основі лежать поняття об'єкт і клас. У предметної області, що цікавить розробника, як об'єкти можуть розглядатися конкретні предмети, а також абстрактні або реальні сутності. Об'єкт має індивідуальність і поводженням, має атрибути, значення яких визначають його стан.

Так, наприклад, конкретний покупець, роблячи замовлення, може виявитися в стані, коли грошей на його рахунку не вистачає для оплати, а його поводження в цьому випадку полягає у зверненні до банку за кредитом.

Кожен об'єкт є представником деякого класу однотипних об'єктів. Клас визначає загальні властивості для всіх його об'єктів. До таких властивостей ставляться склад і структура даних, що описують атрибути класу й відповідних об'єктів, і сукупність методів - процедур, що визначають взаємодію об'єктів цього класу із зовнішнім середовищем.

Об'єкти й класи мають характерні властивості, які активно використаються при об'єктно-орієнтованому підході й багато в чому визначають його переваги. До цих властивостей ставляться:

Інкапсуляція – приховання інформації. При об'єктно-орієнтованому програмуванні є можливість заборонити доступ до атрибутів об'єктів, доступ можливий тільки через його методи. Внутрішня структура об'єкта в цьому випадку схована від користувача, об'єкти можна вважати самостійними сутностями, відділеними від зовнішнього миру. Для того щоб об'єкт зробив деяку дію, йому ззовні необхідно послати повідомлення, що ініціює виконання потрібного методу. Інкапсуляція дозволяє змінювати реалізацію будь-якого класу об'єктів без побоювання, що це викличе небажані побічні ефекти в програмній системі. Тим самим спрощується - процес виправлення помилок і модифікації програм.

Спадкування – можливість створювати із класів нові класи за принципом «від загального до частки». Спадкування дозволяє новим класам, при збереженні всіх властивостей клас^-батьків, додавати свої риси, що відбивають їхня індивідуальність. З погляду програміста новий клас повинен містити тільки коди й дані для нових або методів, що змінюються.

Повідомлення, обробка яких не забезпечується власними методами класу, передається класу-батькові. Спадкування дозволяє створювати ієрархії класів й є ефективним засобом внесення змін і доповнень у програмні системи.

Поліморфізм – здатність об'єктів вибирати метод на основі типів даних, прийнятих у повідомленні.

Кожен об'єкт може реагувати по-своєму на те саме повідомлення. Поліморфізм дозволяє спростити вихідні тексти програм, забезпечує їхній розвиток за рахунок введення нових методів обробки.

Отже, об'єктно-орієнтований підхід полягає в поданні системи у вигляді сукупності класів й об'єктів предметного середовища. При цьому ієрархічний характер складної системи відбивається у вигляді ієрархії класів, а її функціонування розглядається як взаємодія об'єктів, з якими асоціюється, наприклад, продукційні правила. Асоціювання продуційних правил ЕС з ієрархією класів здійснюється за рахунок використання загальних правил, як префікс яких звичайно використається посилання на клас, до якого дане правило застосовне. У процедурній інтерпретації наявність префікса, що зв'язує продукцію із класом, викликає необхідність перебору всіх екземплярів зазначеного в префіксі класу і його підкласів і перевірки істинності умови для атрибутів кожного з екземплярів. Застосування об'єктно-орієнтованого підходу в системах інженерії знань виводить на перший план можливість природної декомпозиції задачі на сукупність підзадач, що представляють досить автономними агентами, що працюють зі знаннями. На сьогоднішній день це єдина практична можливість роботи в умовах експонентного росту складності (кількість взаємозв'язків), характерного для систем, що використають знання. Так практично всі інструментальні засоби для створення динамічних ЕС підтримують об'єктно-орієнтований підхід, об'єднаний із правилами.

Знання й дані. Подання знань

Якщо у вас є проблема або задача, яку не можна вирішити самостійно - ви звертаєтеся до знаючих людей, або до експертів, до тих, хто має ЗНАННЯ.

Термін "системи, засновані на знаннях" (knowledge-based systems) з'явився в 1976 році одночасно з першими системами, що акумулюють досвід і знання експертів.

Це були експертні системи (expert systems) MYCIN й DENDRAL [Shortliffe, 1976; Shortliffe Feigenbaum, Buchanan, 1978] для медицини й хімії. Вони ставили діагноз при інфекційних захворюваннях крові й розшифровували дані мас-спектрографічного аналізу.

Експертні системи з'явилися в рамках досліджень по штучному інтелекті (ШІ) (artificial intelligence) у той період, коли ця наука переживала серйозну кризу, і був потрібний істотний прорив у розвитку практичних додатків. Цей прорив відбувся, коли на зміну пошукам універсального алгоритму мислення й рішення задач дослідникам прийшла ідея моделювати конкретні знання фахівців-експертів.

Так у США з'явилися перші комерційні системи, засновані на знаннях, або експертні системи (ЕС). Ці системи по праву стали першими інтелектуальними системами, і дотепер єдиним критерієм інтелектуальності є наявність механізмів роботи зі знаннями. Так з'явився новий підхід до рішення задач штучного інтелекту — подання знань.

При вивченні інтелектуальних систем традиційно виникає питання - що ж таке знання й чим вони відрізняються від звичайних даних, десятиліттями оброблюваних на комп'ютерах.

Можна запропонувати кілька робочих визначень, у рамках яких це стає очевидним.

Визначення 2.1

Дані — це інформація, отримана в результаті спостережень або вимірів окремих властивостей (атрибутів), що характеризують об'єкти, процеси та явища предметної області. Інакше, дані - це конкретні факти, такі як температура повітря, висота будинку, прізвище співробітника, адреса сайту й ін.

При обробці на ПК дані трансформуються, умовно проходячи наступні етапи:

•D1 - дані як результат вимірів і спостережень;

•D2 - дані на матеріальних носіях інформації (таблиці, протоколи, довідники);

•D3 - моделі (структури) даних у вигляді діаграм, графіків, функцій;

•D4 - дані в комп'ютері мовою описи даних;

•D5 - бази даних на машинних носіях інформації.

Знання ж засновані на даних, отриманих емпіричним шляхом. Вони являють собою результат досвіду й розумової діяльності людини, спрямованої на узагальнення цього досвіду, отриманого в результаті практичної діяльності. Так, якщо озброїти людини даними про те, що в нього висока температура (результат спостереження або виміру), те цей факт не дозволить йому вирішити задачу видужання. А якщо досвідчений лікар поділиться знаннями про те, що температуру можна знизити жарознижуючими препаратами й рясним питвом, те це істотно наблизить рішення задачі видужання, хоча насправді потрібні додаткові дані й більше глибокі знання.

Визначення 2.2

Знання — це зв'язку й закономірності предметної області (принципи, моделі, закони), отримані в результаті практичної діяльності й професійного досвіду, що дозволяє фахівцям ставити й вирішувати задачі в даній області.

При обробці на ПК знання трансформуються аналогічно даним:

•Z1 - знання в пам'яті людини як результат аналізу досвіду й мислення;

•Z2 - матеріальні носії знань (спеціальна література, підручники, методичні посібники);

•Z3 - поле знань - умовний опис основних об'єктів предметної області, їхніх атрибутів і закономірностей, їх єднальних;

•Z4 — знання, описані на мовах подання знань (продукційні мови, семантичні мережі, фрейми);

•Z5 - база знань на машинних носіях інформації.

Часто використається й таке визначення знань:

Знання — це добре структуровані дані, або дані про дані, або метадані.

Ключовим етапом при роботі зі знаннями є формування поля знань (третій етап Z3), ця нетривіальна задача включає виявлення й визначення об'єктів і понять предметної області, їхніх властивостей і зв'язків між ними, а також подання їх у наочній й інтуїтивно зрозумілій формі. Цей термін уперше був уведений при практичній розробці експертної системи по психодіагностиці АВТАНТЕСТ і тепер широко використається розробниками ЕС.

Без ретельного пророблення поля знань не може бути мови про створення бази знань.

Істотним для розуміння природи знань є способи визначення понять. Один із широко застосовуваних способів заснований на ідеї інтенсіонала й екстенсіонала.

Визначення 2.3

Інтенсіонал поняття — це визначення його через співвіднесення з поняттям більш високого рівня абстракції із вказівкою специфічних властивостей.

Наприклад, інтенсіонал поняття "МЕБЛІ":

"предмети, призначені для забезпечення комфортного проживання людини й захаращуючи будинок".

Визначення 2.4

Екстенсіонал — це визначення поняття через перерахування його конкретних прикладів, тобто понять більше низького рівня абстракції.

Екстенсіонал поняття "МЕБЛІ": "Шафа, диван, стіл, стілець і т.д.".

Інтенсіонали формують знання про об'єкти, у те час як екстенсіонал поєднує дані. Разом вони формують елементи поля знань конкретної предметної області.

Для зберігання даних використаються бази даних (для них характерні великий об'єм і відносно невелика питома вартість інформації), для зберігання знань - бази знань (невеликого об'єму, але винятково дорогі інформаційні масиви).

База знань — основа будь-якої інтелектуальної системи, де знання описані на деякій мові подання знань, наближеній до природного.

Знання можна розділити на:

•глибинні;

•поверхневі.

Поверхневі — знання про видимі взаємозв'язки між окремими подіями й фактами в предметній області.

Глибинні — абстракції, аналогії, схеми, що відображують структуру й природу процесів, що протікають у предметній області. Ці знання пояснюють явища й можуть використатися для прогнозування поводження об'єктів.

Поверхневі знання: "Якщо ввести правильний пароль, на екрані комп'ютера з'явиться зображення робочого стола".

Глибинні знання: "Розуміння принципів роботи операційної системи й знання на рівні кваліфікованого системного адміністратора".

Сучасні експертні системи працюють, в основному, з поверхневими знаннями. Це пов'язане з тим, що на даний момент немає універсальних методик, що дозволяють виявляти глибинні структури знань і працювати з ними.

Рисунок 1.5 - Піраміда Н'юела.

Крім того, у підручниках по ШІ знання традиційно ділять на процедурні й декларативні. Історично первинними були процедурні знання, тобто знання, "розчинені" в алгоритмах. Вони управляли даними. Для їхньої зміни було потрібно змінювати текст програм. Однак з розвитком інформатики й програмного забезпечення все більша частина знань зосереджувала в структурах даних (таблиці, списки, абстрактні типи даних), тобто збільшувалася роль декларативних знань.

Рисунок 1.6 – Моделі подання знань

Сьогодні знання придбали чисто декларативну форму, тобто знаннями вважаються пропозиції, записані на мовах подання знань, наближених до природної мови й зрозумілих неспеціалістам.

Один з піонерів ШІ Алан Н'юел проілюстрував еволюцію засобів спілкування людини з комп'ютером як перехід від машинних кодів через символьні мови програмування до мов подання знань (рис. 1.6).

Моделі подання знань

Продуційна модель

МПЗ, засновані на правилах (rule-based), є найпоширенішими й більше 80% ЕС використають саме їх.

Визначення 1.5

Продуційна модель або модель, заснована на правилах, дозволяє представити знання у вигляді пропозицій типу "Якщо (умова), те (дія)".

Під "умовою" (антецедентом) розуміється деяка пропозиція-зразок, по якому здійснюється пошук у базі знань, а під "дією" (консеквентом) - дії, виконувані при успішному результаті пошуку (вони можуть бути проміжними, що виступають далі як умови, і термінальну або цільову, завершальну роботу системи).

Найчастіше висновок на такій базі знань буває прямій (від даних до пошуку мети) або зворотний (від мети для її підтвердження — до даних). Дані — це вихідні факти, що зберігаються в базі фактів, на підставі яких запускається машина висновку або інтерпретатор правил, що перебирає правила із продуційної бази знань.

Продукційна, модель так часто застосовується в промислових експертних системах, оскільки залучає розробників своєю наочністю, високої модульності, легкістю внесення доповнень і змін і простотою механізму логічного висновку.

Є велика кількість програмних засобів, що реалізують продуційний підхід (наприклад, мови високого рівня CLIPS й OPS 5; "оболонки" або "порожні" ЕС - EXSYS Professional і Карра, інструментальні системи KEE, ARTS, PIES), а також промислових ЕС на його основі (наприклад, ЕС, створених засобами G2.

Семантичні мережі

Термін "семантична" означає "змістовна", а сама семантика — це наука, що встановлює відносини між символами й об'єктами, які вони позначають, тобто наука, що визначає зміст знаків. Модель на основі семантичних мереж була запропонована американським психологом Куїліаном. Основною її перевагою є те, що вона більше інших відповідає сучасним поданням про організації довгострокової пам'яті людини.

Визначення 1.6

Семантична мережа — це орієнтований граф, вершини якого — поняття , а дуги — відносини між ними.

Як поняття звичайно виступають абстрактні або конкретні об'єкти, а відносини це зв'язку типу: "це" ("АКО — A-Kind-Of, "is" або "елемент класу"), "має частиною" ("has part"), "належить", "любить".

Можливо запропонувати кілька класифікацій семантичних мереж, пов'язаних з типами відносин між поняттями.

По кількості типів відносин:

•однорідні (з єдиним типом відносин);

•неоднорідні (з різними типами відносин).

По типах відносин:

•бінарні (у які відносини зв'язують два об'єкти);

•N-арні (у які є спеціальні відносини, що зв'язують більше двох понять).

Найбільше часто в семантичних мережах використаються наступні відносини:

•елемент класу (троянда це квітка);

•атрибутивні зв'язки /мати властивість (пам'ять має властивість — об'єм);

•значення властивості (кольори має значення— жовтий);

•приклад елемента класу (троянда, наприклад — чайна);

•зв'язку типу "частина-ціле" (велосипед включає кермо);

•функціональні зв'язки (обумовлені звичайно дієсловами "робить", впливає"...);

•кількісні (більше, менше, дорівнює...);

•просторові (далеко від, близько від, за, під, над...);

•часові (раніше, пізніше, протягом...);

•логічні зв'язки (і, або, не) і ін.

Мінімальний склад відносин у семантичній мережі такий:

• елемент класу або АКО;

• атрибутивні зв'язки /мати властивість;

• значення властивості.

Недоліком цієї моделі є складність організації процедури організації висновку на семантичній мережі.

Ця проблема зводиться до нетривіальної задачі пошуку фрагмента мережі, що відповідає деякої підмережі, що відбиває поставлений запит до бази.

На рис.1.7 зображений приклад семантичної мережі. Як вершини отут виступають поняття "людина", "Петренко", "ЗАЗ-1102", "автомобіль", "вид транспорту" й "двигун".

Для реалізації семантичних мереж існують спеціальні мережні мови, наприклад, NET, мова реалізації систем SIMER + MIR й ін. Широко відомі експертні системи, що використають семантичні мережі як мова подання знань - PROSPECTOR, CASNET, TORUS.

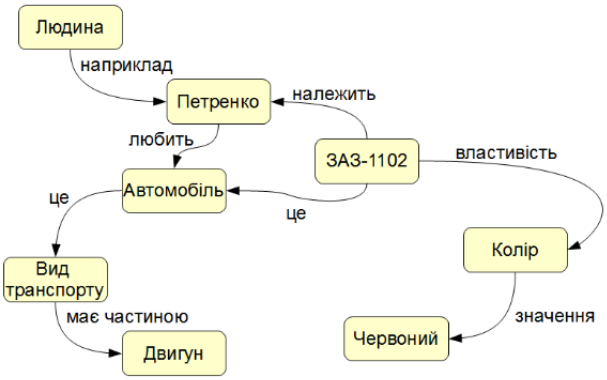

Рисунок 1.7 – Семантична мережа

Фрейми

Термін фрейм (від англ. Frame — "каркас" або "рамка") був запропонований Марвіном Мінським, одним з піонерів ШІ, в 70-і роки для позначення структури знань для сприйняття просторових сцен.

Ця модель, як і семантична мережа, має глибоке психологічне обґрунтування.

Визначення 1.7

Фрейм - це абстрактний образ для подання стереотипу об'єкта, поняття або ситуації. Інтуїтивно зрозуміло, що під абстрактним образом розуміється деяка узагальнена й спрощена модель або структура. Наприклад, проголошення вголос слова "кімната" породжує в почутому образ кімнати: "житлове приміщення із чотирма стінами, підлогою, стелею, вікнами й дверима, площею 6—20 м^2". Із цього опису нічого не можна забрати (наприклад, забравши вікна, ми одержимо вже прикомірок, а не кімнату), але в ньому є "дірки" або "слоти"— це незаповнені значення деяких атрибутів — наприклад, кількість вікон, кольори стін, висота стелі, покриття підлоги й ін.

У теорії фреймів такий образ кімнати називається фреймом кімнати. Фреймом також називається й формалізована модель для відображення образу.

Розрізняють фрейми-зразки або прототипи, що зберігаються в базі знань, і фрейми-екземпляри, які створюються для відображення реальних фактичних ситуацій на основі даних, що надходять,. Модель фрейму є досить універсальної, оскільки дозволяє відобразити все різноманіття знань про світ через:

• фрейму-структури, що використаються для позначення об'єктів і понять (позика, застава, вексель);

• фреймів-ролі (менеджер, касир, клієнт);

• фрейми-сценарії (банкрутство, збори акціонерів, святкування іменин);

• фреймів-ситуації (тривога, аварія, робочий режим пристрою) і ін.

Традиційно структура фрейму може бути представлена як список властивостей:(ІМ'Я ФРЕЙМУ:

(ім'я 1-го слоту: значення 1-го слоту),

(ім'я 2-го слоту: значення 2-го слоту),

.....

(ім'я N-го слоту: значення N-го слоту)).

Той же запис можна представити у вигляді таблиці (див. табл. 1), доповнивши її двома стовпцями.

Таблиця 1.1- Структура фрейму

Ім'я фрейму			
Ім'я слоту	Значення слоту	Спосіб одержання значення	Приєднана процедура

У таблиці додаткові стовпці (3-й й 4-й) призначені для опису способу одержання слотом його значення й можливого приєднання до того або іншого слоту спеціальних процедур, що допускається в теорії фреймів. Як значення слоту може виступати ім'я іншого фрейму, так утворяться мережі фреймів.

Існує кілька способів одержання слотом значень у фреймі-екземплярі:

•за замовчуванням від фрейму-зразка (Default-значення);

• через спадкування властивостей від фрейму, зазначеного в слоті АКО;

• по формулі, зазначеної в слоті;

• через приєднану процедуру;

• явно з діалогу з користувачем;

• з бази даних.

Найважливішою властивістю теорії фреймів є - запозичення з теорії семантичних мереж – так називане спадкування властивостей. І у фреймах, і в семантичних мережах спадкування відбувається по Ако-связям (A-Kind-Of = це). Слот АКО вказує на фрейм більше високого рівня ієрархії, звідки неявно успадковуються, тобто переносяться, значення аналогічних слотів.

Наприклад, у мережі фреймів на рис.4 поняття "учень" успадковує властивості фреймів "дитина" й "людина", які перебувають на більше високому рівні ієрархії. На питання " чи люблять учні солодке?" треба відповідь "так", тому що цією властивістю володіють всі діти, що зазначено у фреймі "дитина".

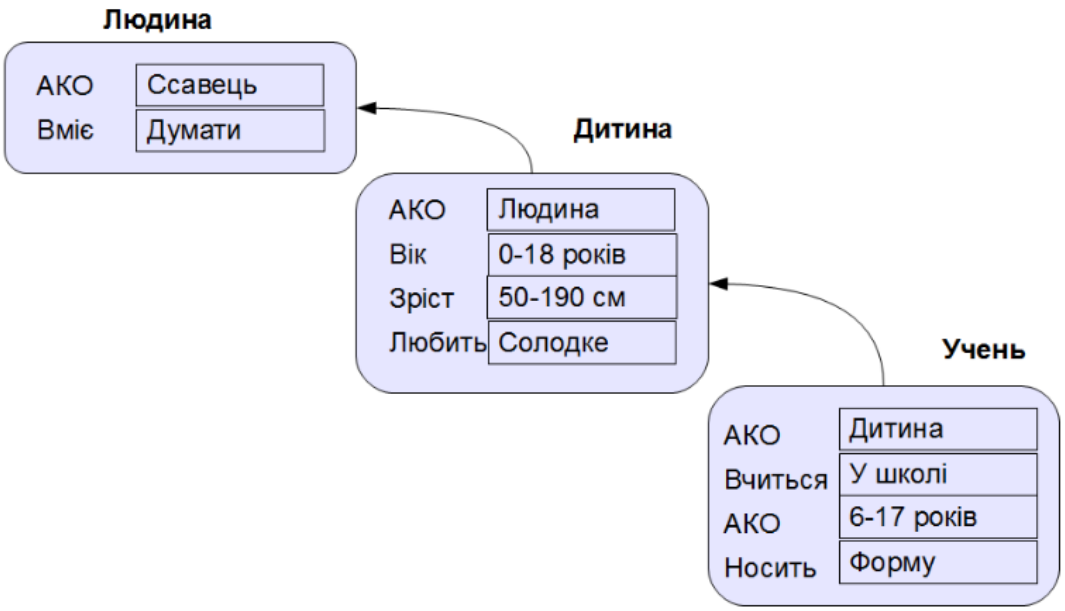

Рисунок 1.8 – Мережа фреймів

Спадкування властивостей може бути частковим: вік для учнів не успадковується із фрейму "дитина", оскільки зазначено явно у своєму власному фреймі.

Основною перевагою фреймів як моделі подання знань є те, що вона відбиває концептуальну основу організації пам'яті людини, а також її гнучкість і наочність.

Спеціальні мови подання знань у мережах фреймів FRL (Frame Representation Language), KRL (Knowledge Representation Language), фреймова "оболонка" Kappa й інші програмні засоби дозволяють ефективно будувати промислові ЕС. Широко відомі такі фрейм-ориентированные експертні системи, як ANALYST, МОДИС, TRISTAN, ALTERID.

Формальні логічні моделі

Традиційно в поданні знань виділяють формальні логічні моделі, засновані на класичному вирахуванні предикатів 1-го порядку, коли предметна область або задача описується у вигляді набору аксіом. Реальне вирахування предикатів 1-го порядку в промислових експертних системах практично не використається.

Ця логічна модель застосовна в основному в дослідницьких "іграшкових" системах, тому що пред'являє дуже високі вимоги й обмеження до предметної області. У промислових же експертних системах використаються різні її модифікації й розширення, виклад яких виходить за рамки цього курсу.

1.2.1 Лабораторна робота №2. Формування правил продукцій

МЕТА: навчити представляти фрагменти знань у вигляді продукційної моделі

ЗАДАЧІ:

1. розглянути приклади продукційних моделей для задач з різних предметних галузей;
2. навчитися складати продукційні моделі для представлення фрагментів знань.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Продукції (як і мережевими моделями) є найбільш популярними засобами подання знань в інформаційних системах. Звичайна форма продукції виглядає так: *ЯКЩО* А, *ТО* В.

Що розуміється у звичайному логічному сенсі, як знак логічного слідування В з істинного А. Можливі й інші інтерпретації продукції, наприклад, А описує деяку умову, необхідну , щоб можна було вчинити дію В. Продукційна модель або модель, заснована на правилах, дозволяє представити знання у вигляді пропозицій типу

«Якщо (умова), то (дія)».

Під умовою розуміється деяка пропозиція — зразок, за яким здійснюється пошук у базі знань, а під дією — дії, що виконуються при успішному результаті пошуку (вони можуть бути проміжними, виступаючими далі як умови, і термінальними або цільовими, такими що завершують роботу системи).

При використанні продукційної моделі база знань складається з набору правил, Програма, що управляє перебором правил, називається машиною виводу. Найчастіше висновок буває прямий (від даних до пошуку мети) або зворотний (від цілі для її підтвердження — до даних). Дані — це вихідні факти, на підставіяких запускається машина виводу.

Однак не слід ототожнювати правило-продукцію і відношення логічного слідування. Справа в тому, що інтерпретація продукції залежить від того, що знаходиться ліворуч і праворуч від знака логічного слідування. Часто під А розуміється деяка інформаційна структура (наприклад, фрейм), а під В — деяка дія, що полягає в її трансформації (перетворенні). Поняття продукції ширше логічного слідування. Як приклад розглянемо правило, взяте з бази знань експертної системи MYCIN, призначеної для діагностики інфекційних захворювань. (табл.1.1)

Таблиця 1.2 - Фрагмент правил діагностики системи MYCIN

ЯКЩО	**ТО**
місце виділення культури - кров **І** реакція мікроорганізму - грам, **І** форма мікроорганізму - паличка, **І** пацієнт відноситься до групи ризику,	з упевненістю (0,6) назва мікроорганізму — *pseudomoniaaeruginosa.*

Умова правила складається з чотирьох фактів, з'єднаних союзом "І". Якщо всі чотири факти мають місце, то вірним буде слідство правила. З кожним правилом зв'язується деяке число, що приймає значення в діапазоні від -1 до 1, що виражає ступінь достовірності наслідків і називається коефіцієнтом впевненості.

У загальному випадку продукційну модель можна представити в наступномувигляді:

N = <A, U, C, I, R>

N — ім'я продукції;

A — сфера застосування продукції; U — умова застосовності продукції; C — ядро продукції;

I — постумови продукції, актуалізуються при позитивній реалізації продукції;

R— коментар, неформальне пояснення(обґрунтування) продукції, час введення в базу знань та ін.

Переваги і недоліки продукційної моделі

Переваги продукційної моделі полягають у наступному.

– Переважна частина людських знань може бути записана у вигляді продукцій.

− Простота створення та розуміння окремих правил.

− Простота поповнення та модифікації бази знань (набору продукцій).

− Простота механізму логічного висновку.

− Розбиття системи продукцій на сфери (декомпозиція) дозволяє ефективновикористовувати ресурси і скоротити час пошуку рішення.

− Можливість реалізації немонотонного логічного виведення й обробки суперечливих фактів.

− Можливість паралельної і асинхронної обробки правил.

Недоліки продукційної моделі проявляються в наступному.

− Відсутність теоретичного обґрунтування в побудові продукційних систем. Восновному при їх побудові використовуються евристичні прийоми.

− При великому числі продукцій процедура перевірки несуперечності правил і коректності роботи системи стає вкрай складною. Саме тому число продукцій, з якими працюють реальні інформаційні системи, не перевищує тисячі.

− Можливість легкого внесення серйозних спотворень в базу знань, що призводять до неправильного функціонування системи (якщо в системі немає розвинених засобів перевірки цілісності бази знань).

Приклад

Нехай заданим є фрагмент бази знань з двох правил:

Правило 1:

ЯКЩО «відпочинок влітку» і «людина активна», ТО «їхати в гори».

Правило 2:

ЯКЩО «любить сонце», ТО «відпочинок влітку».

Припустимо, що в систему надійшли дані людина активна і любить сонце.

Крок 1: Пробуємо *Правило 1* — не працює, так як не вистачає даних

«відпочинок влітку»

Крок 2: Пробуємо *Правило 2* — працює, в базу надходить новий факт

«відпочинок влітку»

Крок 3: Знову пробуємо *Правило 1* — працює, активує мету «їхати в гори», якавиступає як порада, що видається системою.

Приклад продукційної моделі

Розглянемо продукційну модель знань на основі англійської вірша.

Новину вам цю Скажу по секрету

Лондонський міст обвалився!

***

Якщо з колод

Він був побудован

Ті колоди, напевно, згнили!

***

А якщо був він з каміння зведений,То каміння, мабуть, стерлось!

***

А якщо зі сталі Його споруджували

Те іржа сталь цю з'їла!

***

Але вихід я знаю!

І я пропоную

З Золота міст будувати!

***

А щоб усі зналиІ не крали

На міст гармату поставити Солдата й гармату поставити!

***

А щоб караул Всю ніч не заснув

Солдату вина дати й трубку!

Дана продукційна модель є прикладом інструкції з будівництва моста і його експлуатації, заснованої на правилах. У модельній реалізації у вигляді продукційної моделі вірш буде виглядати наступним чином:

Правило 1. *ЯКЩО* з колод *ТО* згниє

Правило 2. *ЯКЩО* зі сталі *ТО* заіржавіє

Правило 3. *ЯКЩО* з каменів *ТО* зітреться

Правило 4. *ЯКЩО* із золота *ТО* вкрадуть

Правило 5. *ЯКЩО* поставити охорону *ТО* не вкрадуть

Правило 6. *ЯКЩО* охорона засне *ТО* вкрадуть

Правило 7. *ЯКЩО* охороні дати вина й трубку *ТО* не засне

Тепер спробуємо за допомогою даної моделі дізнатися, які дії необхідно зробити, щоб побудувати такий міст, який буде стояти досить довго.

Якщо ми говоримо, що збираємося побудувати міст із золота і поставити охорону, дана продукційна модель видасть наступний результат — *Правило 6* говорить про те, що існує ймовірність, що охорона засне і міст вкрадуть, тому нам доведеться додати ще одну умову — дати охорони вина і трубку. Тепер, маючи всі достатні умови, ми зможемо побудувати міст, який буде стоятидосить довго.

Питання для самоконтролю:

1. Що називають «продукційною моделлю»?
2. Що називають «коефіцієнтом впевненості»?
3. Як виглядає продукційна модель в загальному вигляді?
4. У чому переваги і недоліки продукційної моделі?

Завдання для самостійного виконання:

Розробити набір правил для підбору краватки.

Знання, надані консультантом з питань моди (експертом):

1. Краватка потрібна, коли Ви одягаєте офіційний костюм або діловий костюм вробочий день.

2. Якщо Ви збираєтеся поїхати в діловому костюмі на уїк-енд, то Ви можете надіти краватку, але це необов'язково.

3. Якщо Ви носите спортивний піджак, то краватка необхідна.

4. До офіційного костюма підійде чорна краватка-метелик або біла краватка.

5. До блакитного або сірого ділового костюма підійде червона краватка в смужку або однотонна червона.

6. Якщо Ваш костюм зеленого або коричневого кольору, тоді Вам підійде руда смугаста краватка або коричнева смугаста краватка, в крайньому випадку зелена візерунчаста краватка.

Форма звіту: сформувати набір правил, представлених за зразком:

Правіло N: *ЯКЩО* <Передумова> *ТО* <Висновок>.

1.2.2 Лабораторна робота №3. Формування семантичних мереж

МЕТА: навчити представляти фрагменти знань у вигляді семантичних мереж.

ЗАДАЧІ:

1. розглянути приклади семантичних мереж для задач з різних предметних областей;

2. навчити класифікувати семантичні мережі за основними ознаками;

3. навчитися складати семантичні мережі для подання фрагментів знань;

4. закріпити вміння складати семантичні мережі за допомогою комп'ютерногосередовища Explain.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Термін "семантична" означає "смислова", а сама семантика — це наука, що встановлює відносини між символами і об'єктами, які вони позначають, тобто наука, що визначає сенс знаків. Модель на основі семантичних мереж була запропонована американським психологом Куілліаном. Основною її перевагою є те, що вона більше за інших відповідає сучасними уявленнями проорганізацію довготривалої пам'яті людини.

Семантична мережа — це орієнтований граф, вершини якого — поняттяя, а дуги — відносини між ними.

Можна запропонувати кілька класифікацій семантичних мереж, пов'язаних з типами відносин між поняттями:

- за кількістю типів відносин:

однорідні (з єдиним типом відносин);

неоднорідні (з різними типами відносин).

- за типами відносин:

бінарні (в яких відносини пов'язують два об'єкта);

N-арні (в яких є спеціальні відносини, що зв'язують більше двох понять).

Семантична мережа підтримує 4 типи відносин:

– **IS (АКО)** — «є». A IS B — сутність A відноситься до класу сутностей B.Наприклад, «кішка є домашня тварина».

– **ISAPART** - «є частина». A ISAPART B — сутність A є частиною B, входитьдо B. Наприклад, «голова є частина тулуба».

– **HAS** — «приватна ознака». A HAS B — B є приватним ознакою A. Якщо A входить в яку-небудь іншу сутність C, ознака B не передається C. Наприклад, «волосатість є ознака голови, голова є частина тіла — звідси не випливає, щовсе тіло вкрите волоссям» .

– **PROP** — «загальна властивість». A PROP B — B є загальною властивістю A, яке переноситься на всі сутності, в яких входить A. Наприклад, «голова покрита шкірою, голова є частина тіло, і тіло також покрито шкірою».

Мінімальний склад відносин в семантичної мережі такий: елемент класу або АКО; атрибутивні зв'язку / мати властивість;значення властивості.

Недоліком цієї моделі є складність організації процедури виведення на семантичної мережі. Ця проблема зводиться до нетривіальною задачі пошуку фрагмента мережі, відповідного деякої підмережі, що відбиває поставлений запит до бази.

Переваги і недоліки семантичних мереж

Переваги семантичних мереж проявляються у наступному:

– універсальність, що досягається за рахунок вибору відповідного набору відносин. В принципі за допомогою семантичної мережі можна описати будь- яку складну ситуацію, факт або предметну область;

– наочність системи знань, представленої графічно;

– близькість структури мережі, що представляє систему знань, до семантичної структурі фраз природною мовою;

– відповідність сучасним уявленням про організацію довготривалої пам'яті людини (рис.1.9).

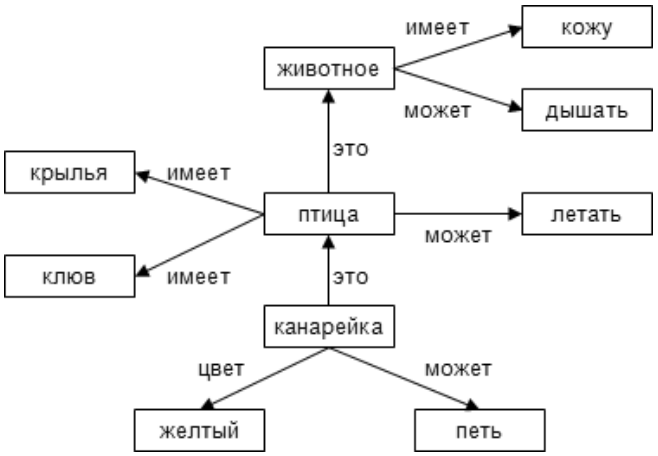

Рисунок 1.9 – Приклад семантичної мережі «Чи може канарейка літати?»

З психології відомо, що люди при запам'ятовуванні часто використовують асоціації і намагаються ієрархічно організувати свої знання.

Спробуйте відповісти якомога швидше на питання про різні властивості птахів, такі як «Канарейка — це птах?», «Канарейка може співати?» Або «Канарейка може літати?».

Хоча відповіді на ці питання, можливо, прості, але час реакції при відповіді на питання «Чи може канарейка літати?» Більше, ніж на питання «Чи може канарейка співати?». Психологи пояснюють цю різницю в часі відповіді тим, що люди запам'ятовують інформацію на самому абстрактному рівні. Замість того, щоб запам'ятовувати конкретні властивості для кожного птаха (канарки літають, дрозди літають, ластівки літають), люди запам'ятовують, що канарейки — птахи, а птиці (зазвичай) мають властивість літати. Таким чином, спроба згадати, чи може канарейка літати, займає більше часу, ніж спогад, чи може канарейка співати. Це відбувається через те, що для отримання відповіді людина повинна довше подорожувати по ієрархії структур пам'яті.

Недоліки семантичних мереж:

− мережева модель не дає (точніше, не містить) ясного уявлення про структурупредметної області, тому формування і модифікація такої моделі скрутні;

− мережеві моделі являють собою пасивні структури, для обробки яких необхідний спеціальний апарат формального виводу;

− проблема пошуку рішення в семантичної мережі зводиться до задачі пошуку фрагмента мережі, відповідного підмережі, що відбиває поставлений запит. Це, в свою чергу, обумовлює складність пошуку рішення в семантичних мережах;

− представлення, використання і модифікація знань при описі систем реального рівня складності виявляється трудомісткою процедурою, особливо при наявності множинних відносин між її поняттями.

Останні твердження розберемо на прикладі «Федір дав книгу Маші» Простіше нікуди?

Зобразимо мережу з Федором і Машею та їх зв'язок (через книгу), спрямовану від Федора до Маші. Але, оскільки всі ці об'єкти мають властивості, то припишемо Федору властивості "хороший і сильний", а книзі і Маші — "дуже гарна і цікава".

Але ж є й багато інших зв'язки, які теж слід відобразити в мережі: сімейні (у Федора дружина і троє дітей), дипломатичні (з тещею), агентурні (Федір шпигун, але не любить про це хвалитися), телепатичні (у Маші ще з одним сором'язливим хлопчиком), виробничі (у Феді з оборонним КБ, а у Маші з начальником) та ін. ».

Приклад для реалізації на комп'ютері:

Будемо вважати, що система «Хлібний магазин» складається з таких елементів: хліб, продавець, покупець, прилавок, автомобіль, шофер, вантажник, гроші, чек. Побудувати семантичну мережу, в якій вершинами будуть перераховані об'єкти, а дугами — відносини між ними.

Хід роботи:

1. Запускаємо середу Explain. Створимо нову схему (ПКМ-Файл-Нова схема), присвоївши їй ім'я «Хлібний магазин»

2. Розмістимо відомі нам елементи (виконати подвійне клацання мишею, ввести назву) (рис.1.10)

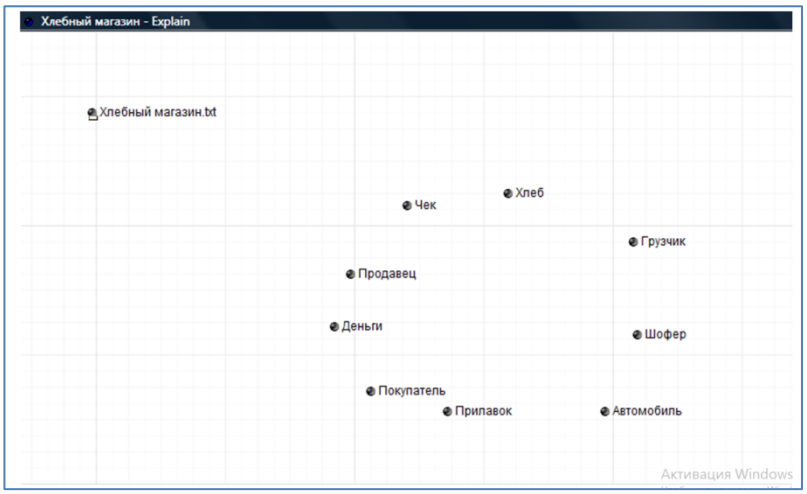

Рисунок 1.10 – Схема введення даних

3. Встановимо зв'язки між об'єктами:

– Продавець: отримує гроші, продає хліб, вручає чек.

– Покупець: дає гроші, отримує хліб, отримує чек.

– Шофер: водить автомобіль.

– Автомобіль: перевозить хліб.

– Вантажник: вивантажує хліб.

– Автомобіль: перевозить хліб.

– Хліб: лежить на прилавку

4. Розмістимо зв'язки в середовищі Explain (Утримуючи праву кнопку мишіпровести лінію від однієї точки до іншої).

5. Підписати лінії (ЛКМ по стрілкою, потім ввести підпис).

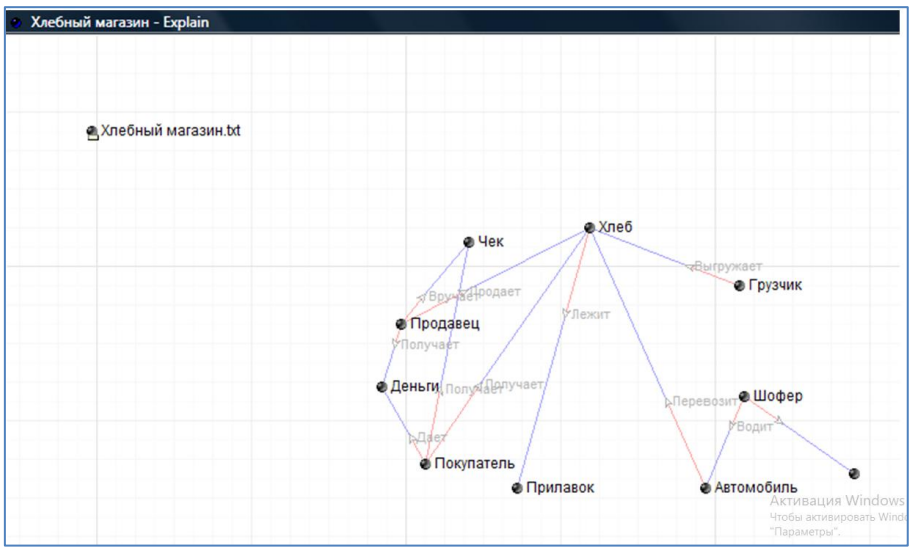

Рисунок 1.11 – Результуюча схема

Питання для самоконтролю:

1. Що називають «семантичної мережею»?

2. З яких елементів складається семантична мережа?

3. Які існують класифікації семантичних мереж?

4. Які типи відносин підтримуються семантичними мережами?

5. У чому переваги і недоліки семантичних мереж?

Завдання для самостійного виконання:

Представити у вигляді семантичної мережі структуру органів влади у Франції в часи Третьої республіки (1875-1940 рр.).

Органи влади включають в себе центральні органи та органи місцевого самоврядування (муніципалітети). Центральні органи — це президент, уряд і Національні збори (парламент) з двох палат (нижня — Палата депутатів, верхня

— Сенат). Виборці безпосередньо обирають органи місцевого самоврядування та Палату депутатів. Вибори Сенату - багатоступеневі. Муніципалітети вибирають колегію вибірників, а вони вже вибирають сенаторів. Сенат і Палата депутатів на спільному засіданні обирають президента. Президент призначає уряд, але повинен при цьому враховувати думку Палати депутатів, оскільки вона має право висловити уряду недовіру (тобто відправити його у відставку). Уряд призначає в кожен департамент префекта, який має право скасувати будь- яке рішення місцевої влади.

Форма звіту: графічне представлення семантичної мережі, виконане в середовищі Explain.

1.2.3 Лабораторна робота №4. Робота з фреймами

МЕТА: навчити представляти фрагменти знань у вигляді фреймової моделі.

ЗАДАЧІ:

1. розглянути приклади фреймових моделей;
2. вивчити компоненти фреймової моделі;
3. навчити складати фреймові моделі для представлення фрагментів знань.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Фреймова модель представлення знань була запропонована М. Мінським в 1979році і є розвитком семантичних мереж.

Фрейм (англ. *frame*) - абстрактний образ для представлення деякого стереотипу сприйняття. Кожен фрейм має власну назву і список слотів і їх значень.

Значеннями можуть бути дані будь-якого типу, а також назва іншого фрейма. Таким чином, фрейми утворюють мережу. Крім того, існує зв'язок між фреймами типу АКО (a kind of), яка вказує на фрейм більш високого рівня ієрархії, звідки неявно успадковуються список і значення слотів. При цьому можливе множинне спадкування — перенесення властивостей від декількох прототипів. Будь який фрейм може бути представлений таким чином:

(ІМ'Я ФРЕЙМА:(Ім'я 1-го слоту: значення 1-го слоту),(Ім'я 2-го слоту: значення 2-го слоту),

...

(Ім'я N-го слоту: значення N-го слоту)).

Табличне представлення слоту виглядає наступним чином (табл. 1.2):

Таблиця 1.2 - Структура фрейму

ІМ’Я ФРЕЙМУ			
Ім’я слоту	Значення слоту	Спосіб отриманнязначення	Приєднана процедура \Демон

Ім'я фрейму служить для ідентифікації фрейму в системі і має бути унікальним. Фрейм являє собою сукупність слотів, число яких може бути довільним.

Кількість слотів в кожному фреймі встановлюється проектувальником системи, при цьому частина слотів визначається самою системою для виконання специфічних функцій (системні слоти), прикладами яких є: слот-покажчик предка даного фрейма (IS-A), слот-покажчик дочірніх фреймів, слот для введення імені користувача, слот для введення дати визначення фрейму, слот для введення дати зміни фрейму і т.д.

Ім'я слота має бути унікальним в межах фрейму. Зазвичай ім'я слота являє собою ідентифікатор, який наділений певною семантикою. Як ім'я слота може виступати довільний текст.

Наприклад, <Ім'я слота> = Головний герой роману Ф. М. Достоєвського «Ідіот», <Значення слота> = Князь Мишкін. Імена системних слотів зазвичай зарезервовані, в різних системах вони можуть мати різні значення.

Значення слота повинно відповідати зазначеному типу даних і умові успадкування.

Спосіб отримання значення визначає, як саме встановлюється значення конкретного слота. Існує кілька способів отримання значень слота (табл. 2), вибір способу залежить від властивостей самих даних.

Таблиця 1.3 - Спосіб отримання значень слотів

Спосіб	**Опис**
За замовчуванням від прототипу (предка)	Слоту присвоюється значення, визначене за замовчуванням під фреймі-прототипі, деякі стандартнізначення.

Через успадкування	Відрізняється від першого способу тим, що значення задано в спеціальному слоті батьківського фрейма, поєднаного з поточним зв'язком AKO.
За формулою	Слоту призначається формула, результат обчислення якої є значенням слоту.
Через приєднану процедуру	Слоту призначається процедура, що дозволяє отримати значення слота алгорітмічно.
Із зовнішніх джерел даних	При використанні моделі в інтелектуальних системах дані, що є значеннями слотів, можуть надходити з баз даних, від системи датчиків, від користувача.

Демоном називається процедура (табл.1.4), що автоматично запускається при виконанні деякої умови. Демони автоматично запускаються при зверненні до відповідного слоту. Типи демонів пов'язані з умовою запуску процедури. Демон з умовою IF-NEEDED запускається, якщо в момент звернення до слоту його значення не було встановлено. Демон типу IF-ADDED запускається при спробі зміни значення слота. Демон IF-REMOVED запускається при спробі видалення значення слота. Можливі також інші типи демонів. Демон є різновидом пов'язаної процедури.

Таблиця 1.4 - Найбільш поширені демони

Демон	Подія	Опис
IF-REMOVED	якщо видалено	Виконується, коли інформація видаляється з слота.
IF-ADDED	якщо додано	Виконується, коли нова інформація записується у слот.
IF-NEEDED	на вимогу	Виконується, коли запитується інформація з пустого слота.
IF-DEFAULT	за замовчуванням	Виконується, коли встановлюється значення за замовчуванням.

Приєднана процедура запускається за повідомленням, переданим з іншого фрейму. Демони і приєднані процедури є процедурними знаннями, об'єднанимиразом з декларативними в єдину систему. Ці процедурні знання є засобами управління виводу у фреймових системах, причому з їх допомогою можна реалізувати будь-який механізм виведення. Подання таких знань і заповнення ними інтелектуальних систем — дуже нелегка справа, яка вимагає додаткових витрат праці і часу розробників. Тому проектування фреймових систем виконується, як правило, фахівцями, які мають високий рівень кваліфікації в галузі штучного інтелекту.

Існує кілька видів фреймів, які дозволяють описати предметну область і вирішувати завдання. У табл.1.5 представлені найбільш поширені типи фреймів, вказані типи знань, які вони відображають, а також приклади фреймів даного типу з різних предметних областей.

Таблиця 1.5 - Типи фреймів

Тип фрейму	**Тип знання**	**Опис**	**Приклад**
За пізнавальним призначенням			
Фрейми-прототипи (шаблони, зразки)	інтенсіональні	відображають знання про абстрактністереотипних поняттях, які є класами якихось конкретних об'єктів	людина, автомобіль
Фрейми-екземпляри (приклади)	екстенсіональні	відображають знання про конкретніфакти предметної області	Іванченко І. І.,ЗАЗ-2110
За функціональним призначенням			
Фрейми-структури (об'єкти)	декларативні	відображають абстрактні і конкретні предмети і поняття предметної області (містять набір характеристик, що описує об'єкт або поняття)	кредит, застава, вексель, людина, лекція
Фрейми-операції	процедурні	відображають різні процеси перетворення або використання об'єктів предметної області (містять набір характеристик процесу)	процеси отримання кредиту, синтезу пристроїв
Фрейми-ситуації	прагматичні	відображають типові ситуації, в яких можуть перебувати фрейми об'єкти і	аварія, тривога,

		фрейми ролі (містять набір характеристик, що ідентифікують ситуацію)	робочий режим пристрою
Фрейми-сценарії	технологічні	відображають розвиток ситуації, типову структуру для деякої дії, поняття, події, відображає динаміку (містять набір характеристик, дозволяють забезпечити розвиток системи за даним сценарієм)	банкрутство, здача іспиту
Фрейми-ролі	функціональні		менеджер, касир, клієнт, студент

Слід зазначити близькість понять, що використовуються у фреймовій і об'єктно-орієнтованої моделях представлення знань, а також в базах даних.

Таблиця 1.6 - Основні поняття фреймовой і об'єктно-орієнтованої моделей

Фреймова модель	**Об’єктно-орієнтованамодель**	**База даних**
Фрейм (фрейм-зразок)	Класс	Таблиця
Фрейм-екземпляр	Об’єкт (экземпляр класу)	Рядок
Слот	Атрибут	Стовбчик
Демон	Подія (з методом її обробки)	Тригер

Приєднана процедура	Метод	Процедура щозберігаєтьс

Переваги і недоліки фреймових моделей

Переваги фреймових моделей:

− фрейми дозволяють організувати чітко виражену ієрархію знань.

− процедурні вкладення є важливою властивістю фреймів, так як вони дозволяють тісно пов'язати процедурні та декларативні знання про об'єкт, тобтоз'єднати інформаційну, функціональну та поведінкові складові об'єкта в єдине ціле;

− розвинена процедура спадкування значень;

− дозволяють представляти складні об'єкти не у вигляді великої семантичної структури, а у вигляді єдиної сутності.

Недоліком фреймовой моделі є відсутність спеціального механізму управління висновком, у зв'язку з чим, розробники повинні реалізувати даний механізм за допомогою приєднаних процедур.

Приклад розв’язання задачі

Задача. Побудувати фреймову модель подання знань в предметній області «Ресторан» (відвідування ресторану).

Опис процесу розв’язання. Для побудови фреймової моделіподання знань необхідно виконати наступні кроки:

1. Визначити абстрактні об'єкти і поняття предметної області, необхідні для вирішення поставленого завдання. Оформити їх у вигляді фреймів-прототипів (фреймів-об'єктів, фреймів-ролей).
2. Задати конкретні об'єкти предметної області. Оформити їх у вигляді фреймів-екземплярів (фреймів-об'єктів, фреймів-ролей).
3. Визначити набір можливих ситуацій. Оформити їх у вигляді фреймів- ситуацій (прототипи). Якщо існують прецеденти по ситуацій в предметної області, додати фрейми-екземпляри (фрейми-ситуації).
4. Описати динаміку розвитку ситуацій (перехід від одних до інших)

через набір сцен. Оформити їх у вигляді фреймів-сценаріїв.

5. Додати фрейми-об'єкти сценаріїв і сцен, які відображають дані конкретного завдання.

Розв’язання.

1) Ключові поняття даної предметної області — *ресторан*, той, хто відвідує ресторан (*клієнт*) і ті, хто його обслуговують (для простоти обмежимося тільки *офіціантами*). У обслуговуючого персоналу і клієнтів є спільні характеристики, тому доцільно виділити загальне абстрактне поняття — *людина*. Тоді фрейми «Ресторан» і «Людина» є прототипами-зразками, а фрейми «Офіціант» і «Клієнт» — прототипами-ролями. Також потрібно визначити основні слоти фреймів — характеристики, що мають значення для розв'язуваної задачі.

ЛЮДИНА			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Стать	Чоловіча абожіноча	Із зовнішніхджерел	
Вік	Від 0 до 120 років	Із зовнішніхджерел	

РЕСТОРАН			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Назва		Із зовнішніхджерел	
Адреса		Із зовнішніхджерел	
Час роботи		Із зовнішніхджерел	

Спеціалізація		Із зовнішніхджерел	
Клас	Середній абовищий	Із зовнішніхджерел	

Фрейми-нащадки містять всі слоти своїх батьків, вони явно прописуються тільки в разі зміни будь-якого параметра.

ОФІЦІАНТ (*АКО* ЛЮДИНА)			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Вік	Від 18 до 55 років	Із зовнішніх джерел	
Стаж роботи		Із зовнішніхджерел	
Заробітна платня		Із зовнішніхджерел	
Графік роботы		Із зовнішніхджерел	
Місто роботы	Фрейм-об’єкт	Із зовнішніхджерел	

КЛІЄНТ (*АКО* ЛЮДИНА)			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Вид оплати	Готівка абокартка	За замовчуванням (готівка)	
Статус	Звичайний абоVIP	За замовчуванням (звичайний)	

Форма замовлення	Замовлення є або нема	За замовчуванням (замовлення нема)	
Чайоі		Із зовнішніхджерел	

2) Фрейми-зразки описують конкретну ситуацію: які ресторани є в місті, як саме організовується відвідування, хто є відвідувачем, хто працює в обраному ресторані та ін..

Тому визначимо такі фрейми-зразки, що є спадкоємцями фреймів-прототипів:

КАФЕ-РЕСТОРАН «СМАКОТА» (*АКО* РЕСТОРАН)			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Назва	Смакота	Із зовнішніхджерел	
Адреса	м. Одеса вул.Буніна 25	Із зовнішніхджерел	
Час роботи	9:00 — 23:00	Із зовнішніхджерел	
Спеціалізація	Піцерия	Із зовнішніхджерел	
Клас	Середній	Із зовнішніхджерел	

КАФЕ-РЕСТОРАН «СМАЧНА ЇЖА» (*АКО* РЕСТОРАН)			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Назва	Смачна їжа	Із зовнішніхджерел	

Адреса	м. Одеса вул. Грушевського 14	Із зовнішніхджерел	
Час роботи	9:00 — 23:00	Із зовнішніхджерел	
Спеціалізація	Паб	Із зовнішніх джерел	
Клас	Вищій	Із зовнішніх джерел	

СЕРГІЙ (*АКО* ОФІЦІАНТ)			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Вік	27	Із зовнішніхджерел	
Стать	Чоловіча	Із зовнішніхджерел	
Стаж роботи	5	Із зовнішніхджерел	
Заробітна платня	7000	Із зовнішніхджерел	
Графік роботи	Через день з 18:00 дд 23:00	Із зовнішніхджерел	
Місце роботи	**КАФЕ «СМАЧНАЇЖА»**	Із зовнішніхджерел	

МАРИНА (*АКО* ОФІЦІАНТ)			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Вік	24	Із зовнішніхджерел	

Стать	Жіноча	Із зовнішніхджерел	
Стаж роботи	2	Із зовнішніхджерел	
Заробітна платня	8200	Із зовнішніхджерел	
Графік роботи	Кожного дня з 9:00 до 14:00	Із зовнішніхджерел	
Місце роботи	Кафе-ресторан «Смакота»	Із зовнішніхджерел	
ПЕТРО (*АКО* КЛІЄНТ)			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Стать	Чоловіча	Із зовнішніхджерел	
Вік	19	Із зовнішніхджерел	
Спосіб оплати	Готівка	За замовчуванням (готівка)	
Статус	Звичайний	За замовчуванням (звичайний)	
Форма замовлення	Замовлення немає	За замовчуванням (замовлення немає)	
Чайові	7% від суми замовлення	Із зовнішніхджерел	

3) Фрейми-ситуації описують можливі ситуації. У ресторані клієнт потрапляє вкілька типові ситуацій: замовлення і оплата.

Можливі й інші не типові ситуації: клієнт подавився, у клієнта немає готівки для оплати рахунку і т.д. Розглянемо типові ситуації (їх може бути більше):

ЗАМОВЛЕННЯ			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Перелік блюд		Із зовнішніхджерел	IF-ADDED (змінює слот «Перелік цін»)
Перелік цін		Приєднана процедура	IF-ADDED (змінює слот «Сума замовник»)
Сума замовлення		Приєднана процедура	
Прийняв замовлення	Фрейм-зразок	Із зовнішніхджерел	
Зробив замовлення	Фрейм-зразок	Із зовнішніхджерел	
ОПЛАТА			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Вид платежу		Із зовнішніхджерел	IF-ADDED (змінює

			слот «Чайові»)
Чайові		Приєднана процедура	
Сплатив	Фрейм-зразок	Приєднана процедура	
Замовлення	Фрейм-образец	Із зовнішніхджерел	IF-ADDED (змінює слот «Сплатив »)

4) Ситуації виникають після настання якихось подій, виконання умов і можуть слідувати одна за одною. Динаміку предметної області можна відобразити в фреймах-сценаріях. Їх може бути безліч, опишемо найбільш загальний і типовий сценарій відвідування ресторану:

ВІДВІДУВАННЯ РЕСТОРАНУ			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Відвідувач	Фрейм-об’ект	Із зовнішніхджерел	
Ресторан	Фрейм-об’ект	Із зовнішніх	
		джерел	

Офіціант	Фрейм-объект	Приєднана процедура (визначає заобраним рестораном)	IF-ADDED, IF-REMOVED (змінюють слот «Офіціант»)
Сцена 1	Вхід, вибір	Із зовнішніхджерел	
Сцена 2	Замовлення	Із зовнішніхджерел	
Сцена 3	Прийом їжі	Із зовнішніхджерел	
Сцена 4	Сплата	Із зовнішніхджерел	
Сцена 5	Вихід	Із зовнішніхджерел	

5) Нехай в рамках нашого завдання Петро відвідав ресторан «Смачна їжа».Тоді фрейми будуть заповнені таким чином:

ВІДВІДУВАННЯ «СМАЧНОЇ ЇЖІ» **(**АКО **ВІДВІДУВАННЯ РЕСТОРАНУ)**			
Ім’я слоту	**Значення слоту**	**Спосіб Отримання даних**	**Демон**
Відвідувач	**ПЕТРО**	Із зовнішніхджерел	
Ресторан	**КАФЕ «СМАЧНА ЇЖА»**	Із зовнішніх джерел	
Офіціант	**СЕРГІЙ**	Приєднана процедура (визначає по обраному ресторану)	IF-ADDED, IF-REMOVED (змінює слот «Офіціант»)

Сцена 1	Вхід, вибір	Із зовнішніхджерел	
Сцена 2	**ЗАМОВЛЕННЯ ПЕТРА**	Із зовнішніхджерел	
Сцена 3	Прийом єжі	Із зовнішніхджерел	
Сцена 4	**СПЛАТАПЕТРА**	Із зовнішніхджерел	
Сцена 5	Вихід	Із зовнішніхджерел	

ЗАМОВЛЕННЯ ПЕТРА (АКО **ЗАМОВЛЕННЯ)**			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Перелік блюд	Відбивна, темне пиво	Із зовнішніхджерел	IF-ADDED (змінює слот «Перелік цін»)
Перелік цін	25; 7	Приєднана процедура	IF-ADDED (змінює слот «Сума замовник»)
Сума замовлення	32	Приєднана процедура	
Прийняв замовлення	**СЕРГІЙ**	Із зовнішніхджерел	
Зробив замовлення	**ПЕТРО**	Із зовнішніхджерел	

СПЛАТА ПЕТРА (АКО **СПЛАТА)**			
Ім’я слоту	**Значення слоту**	**Спосіб отримання даних**	**Демон**
Спосіб платежу	Готівка	Із зовнішніхджерел	IF-ADDED (змінює слот «Чайові»)
Чайові	2,25	Приєднана процедура	
Сплатив	**ПЕТРО**	Приєднана процедура	
Замовлення	**ЗАМОВЛЕННЯ ПЕТРА**	Із зовнішніхджерел	IF-ADDED (змінює слот «Сплатив»)

Взаємозв'язок різних видів фреймів відображається графічно у вигляді графу (рис.1.12)

Використання фреймової моделі аналогічно семантичної, тільки в процесі отримання відповіді крім вершин враховуються і слоти. Наприклад, отримати відповідь на питання «Хто працює офіціантом в ресторані "Смачна їжа"?» можна наступним чином: із запиту зрозуміло, що необхідно знайти фрейм «Ресторан "Смачна їжа"» і простежити зв'язок з фреймом «Сергій», який є нащадком фрейму «Офіціант».

Також можна знайти слот «Місце роботи» і перевіривши його значення у фреймах спадкоємців фрейму «Офіціант» визначити, що офіціантом в ресторані "Смачна їжа" працює Сергій.

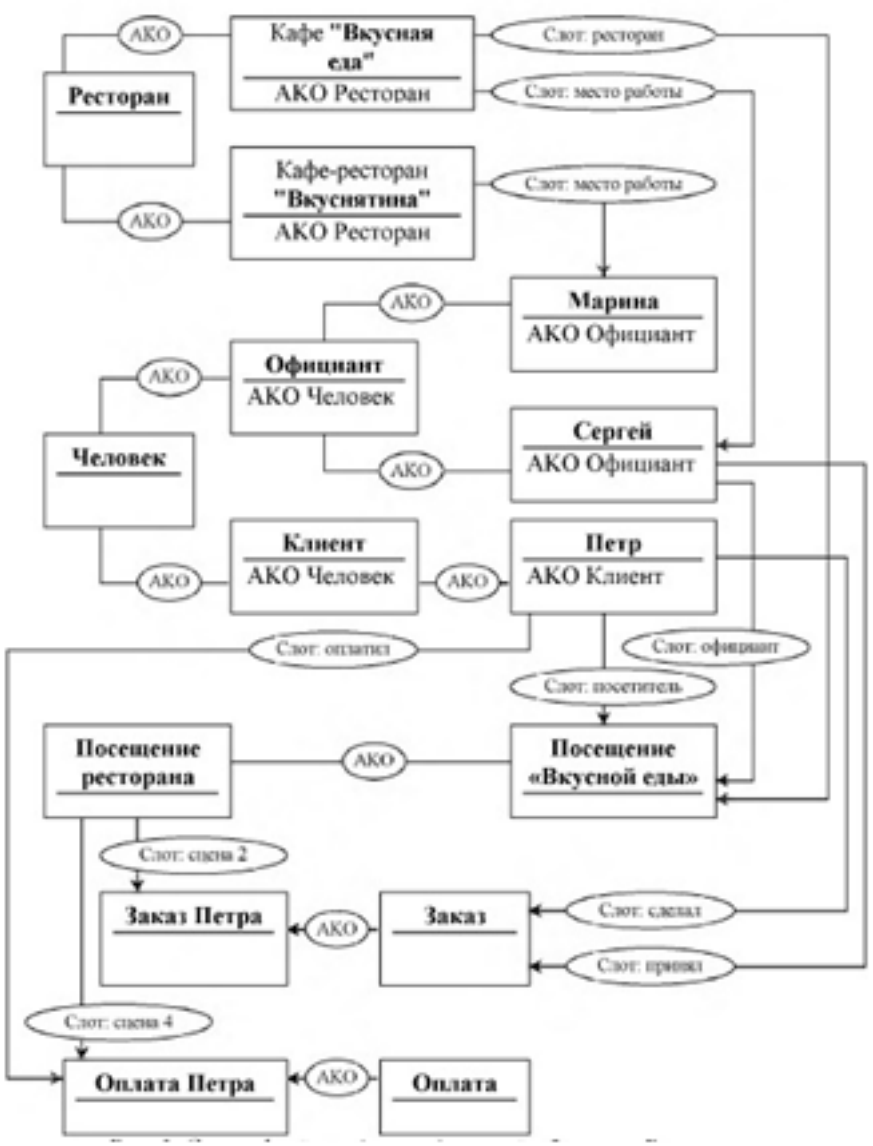

Рисунок 1.12 - Схема фреймів для предметної області «Ресторан»

Завдання для самостійного виконання:

Побудувати фреймову модель подання знань в області «Університет» (навчальний процес).

Форма звіту: подання фреймової моделі у вигляді набору таблиць.

1.2.4 Лабораторна робота №5. Формування формальних моделей

МЕТА: навчити формувати формальні моделі знань.

ЗАДАЧІ: ознайомити з поняттям «формальна модель»; розглянути приклади формальних моделей.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Формальна (логічна) модель подання знань — модель в уявленні знань, основна ідея побудови якої — розгляд всієї інформації, необхідної для вирішення прикладної задачі, як сукупності фактів і тверджень, які представляються як сукупність формули в деякій логіці.

Знання відображаються сукупністю таких формул, а отримання нових знань зводиться до реалізації процедур логічного висновку. В основі логічних моделей представлення знань лежить поняття формальної теорії, що задається кортежем:

S = <B, F, A, R>, де:

B — рахункове безліч базових символів (алфавіт);F — множина формул;

A — виділена підмножина апріорі істинних формул (аксіом);

R — кінцева безліч відносин між формулами, звана правилами виведення.

Для побудови формальних моделей часто використовуються природні мови. Приміром, Коперник сформулював геліоцентричну модель світу таким чином:

- Земля обертається навколо Сонця і навколо власної осі;
- Орбіти всіх планет проходять навколо Сонця.

Крім цього для побудови формальних моделей використовуються формальні мови, одним з яких є мова математики. Так, наприклад, Ньютон формалізував закон всесвітнього тяжіння за допомогою відомої математичної формули. У широкому сенсі прикладами формальних моделей можуть служити всі видиформул, таблиці, графи, схеми, карти і т.д.

Приклад. Складемо логічну схему персонального комп'ютера. Перерахуємо елементи ПК, встановивши взаємозв'язок між ними:

Елемент	**Пов’язаний з…**
Процесор	Системна плата \ Системна шина

Оперативна пам’ять	Системна плата \ Системна шина
Контролер клавіатури	Системна плата \ Системна шина
Контролер монітора	Системна плата \ Системна шина
Контролери додаткових пристроїв	Системна плата \ Системна шина
Контролер портів	Системна плата \ Системна шина
Контролер дисків	Системна плата \ Системна шина
Жорсткий диск	Контролер дисків
Дисководи гнучких дисків	Контролер дисків
Клавіатура	Контролер клавіатури
Монітор	Контролер монітора
Додаткові пристрої	Контролери додаткових пристроїв

Наведені вище відомості представимо у вигляді схеми (рис.1.13)

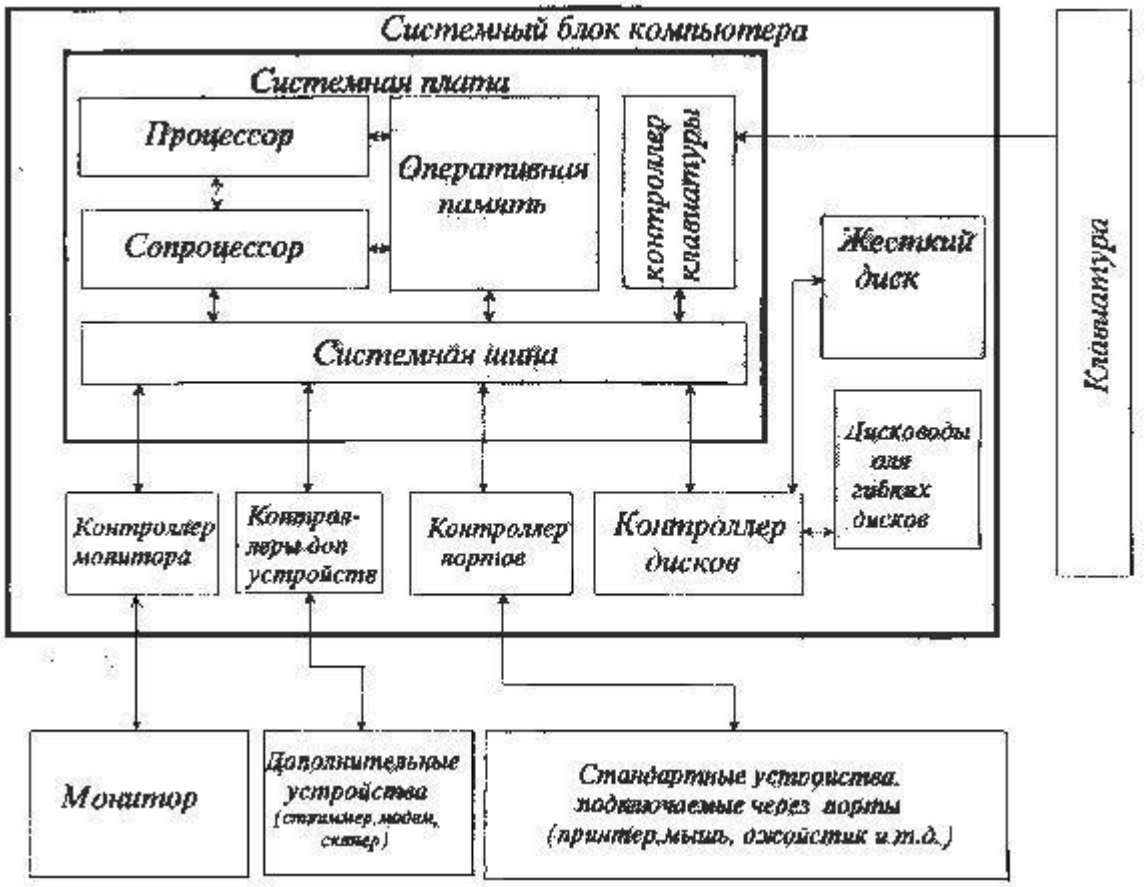

Рисунок 1.13 - Логічна схема ПК

Переваги формальних (логічних) моделей:

– У якості «фундаменту» тут використовується класичний апарат математичної логіки, методи якої досить добре вивчені і формально обґрунтовані.

– Існують досить ефективні процедури виведення, в тому числі реалізовані в мові логічного програмування Пролог, що використовують механізми автоматичного доведення теорем для пошуку і логічно осмисленого виведення інформації

– У базах знань можна зберігати лише безліч аксіом, а всі інші знання отримувати з них за правилами виведення, а також дані, факти та інші відомостіпро людей, предмети, події та процеси.

Недоліки формальних (логічних) моделей:

– "Закритість" формальних моделей, їх негнучкість. Модифікація та розширення тут завжди пов'язані з перебудовою всієї моделі, що для практичних систем складно і трудомістко. У них дуже складно враховувати зміни, що відбуваються. Тому формальні моделі, як моделі подання знань, використовуються в тих предметних областях, які добре локалізуються і мало залежать від зовнішніх факторів.

Питання для самоконтролю:

1. Що називають «Формальною (логічної) моделлю»?
2. Наведіть приклади формальних моделей?
3. Назвіть переваги і недоліки формальних моделей.

Завдання для самостійного виконання:

Сформувати формальну модель гри в шахи.

Форма звіту: Заповнення даних таблиць.

ФІГУРИ		
Назва фігури	*Початкове положення*	*Спосіб ходу*

ПІДСУМОК ГРИ	
Термін	*Пояснення*

1.2.5 Лабораторна робота №6

«Пошук в глибину»

МЕТА: ознайомити з алгоритмом пошуку в глибину, навчити застосовувати даний алгоритм для вирішення задач пошуку в просторі станів, аналізувати його ефективність.

ЗАДАЧІ:

1. розглянути приклади завдань, для вирішення яких використовується пошук вглибину;

2. розглянути алгоритм пошуку в глибину на просторі станів, представленого увигляді графа (дерева пошуку);

3. самостійно скласти дерево пошуку, використовуючи алгоритм пошуку в глибину.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Непоінформовані пошук (також сліпий пошук, метод грубої сили, англ. Uninformed search, blind search, brute-force search) — стратегія пошуку рішень у просторі станів, при якому не використовується додаткова інформація про стани, крім тієї, яка представлена у визначенні завдання.

В результаті неінформованого пошуку вдається сформувати наступників і відрізняти цільовий стан від нецільового.

Пошук в глибину (depth-first search, DFS) — стратегія пошуку рішень у просторі станів, при якій завжди розгортається найглибший вузол в поточній периферії дерева пошуку. При пошуку в глибину аналізується перший за списком наступник поточного вузла, потім - його перший наступник і т. д. (рис.1.14) Розгорнуті вузли видаляються з периферії, тому надалі пошук «поновлюється» з наступного самого верхнього вузла, в якому залишилися ще недосліджені приймачі.

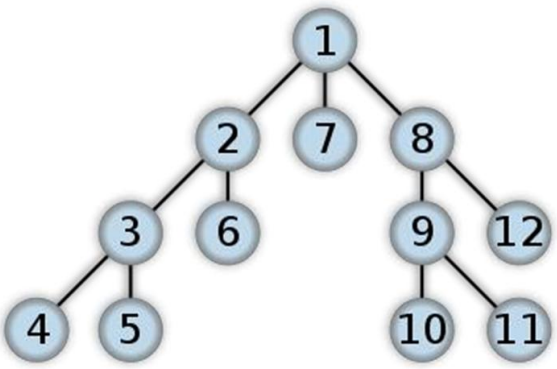

Рисунок 1.14 - Схема пошуку в глибину

Приклад 1 «Задача про розсіяного студента».

Збираючись на іспит з основ штучного інтелекту студент Іванченко усвідомив, що загубив залікову книжку. Іванов точно знає, що залікову книжку загубив десь вдома (рис.1.15). Для раціонального пошуку втраченого предмета Іванченко застосовує знання, отримані на відвіданих заняттях з основ ШІ.

Примітка: Залікова книжка була втрачена в кухні (на рис.1.15 відзначена квадратом).

Пошуки Іванченко почав у вітальні (на рис.1.15 відзначена овалом).

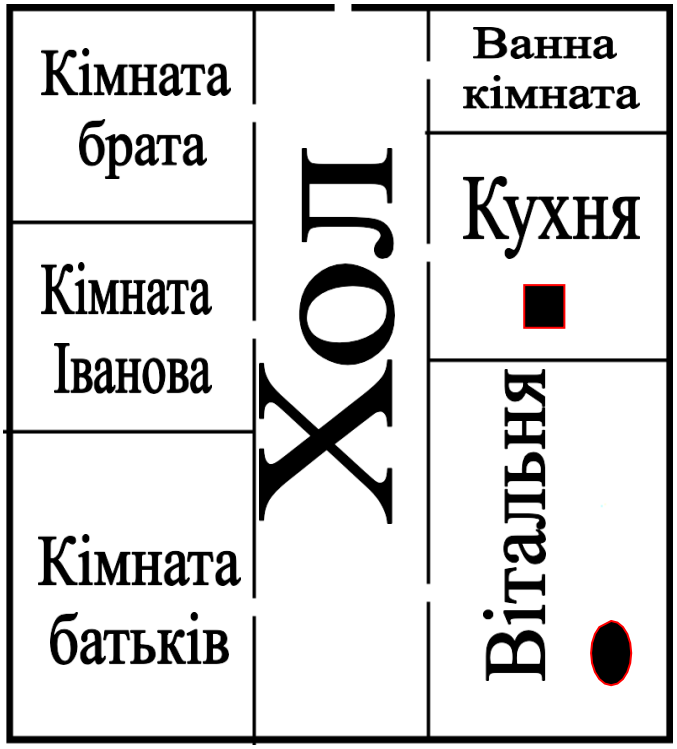

Рисунок 1.15 - План квартири Іванченко

План пошуку:

1. Студент знаходиться у вітальні (початок пошуку). Залікова книжка не знайдена.

2. Студент переміщається в хол. Залікова книжка не знайдена.

3. З холу переміщається в кімнату батьків. Залікова книжка не знайдена.

4. Повернувся з кімнати батьків в хол.

5. З холу переміщається в свою кімнату. Залікова книжка не знайдена.

6. Повернувся зі своєї кімнати в хол.

7. З холу переміщається в кімнату брата. Залікова книжка не знайдена.

8. Повернувся з кімнати брата в хол.

9. Переміщується з холу в кухню. Залікова книжка знайдена !!!

10. Відтворимо пройдений шлях у вигляді графа (рис.1.16), на якому вершини —назви кімнат. Таке зображення інакше називають «пошуковим деревом»

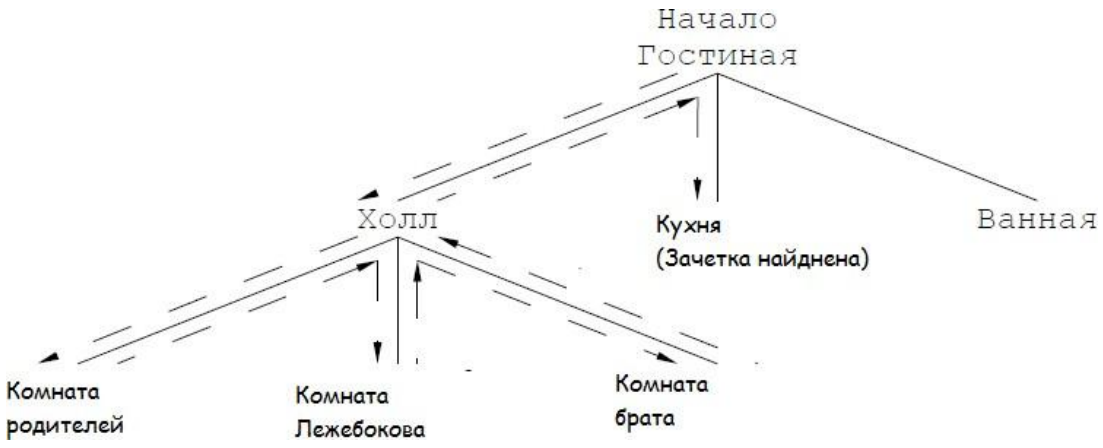

Рисунок 1.16 - Дерево пошуку. Пройдений маршрут

Переваги і недоліки пошуку в глибину

Недоліком пошуку в глибину є те, що в ньому може бути зроблений неправильний вибір і перехід в тупикову ситуацію, пов'язану з проходженням вниз по дуже довгому (або навіть нескінченного) шляху,

притому, що інший варіант міг би привести до вирішення, що знаходиться недалеко від кореня дерева пошуку.

Питання для самоконтролю:

1. Що називають «поінформованим пошуком»?
2. Що називають «пошуком в глибину»?
3. Що називають «пошуковим деревом»?
4. Які існують недоліки пошуку в глибину?

Завдання для самостійного виконання:

Обійти лабіринт у пошуках виходу (рис.1.17), використовуючи метод пошуку в глибину. Тупики та ходи лабіринту позначені латинськими літерами. Вхід позначений літерою **A**, вихід позначений літерою **X**.

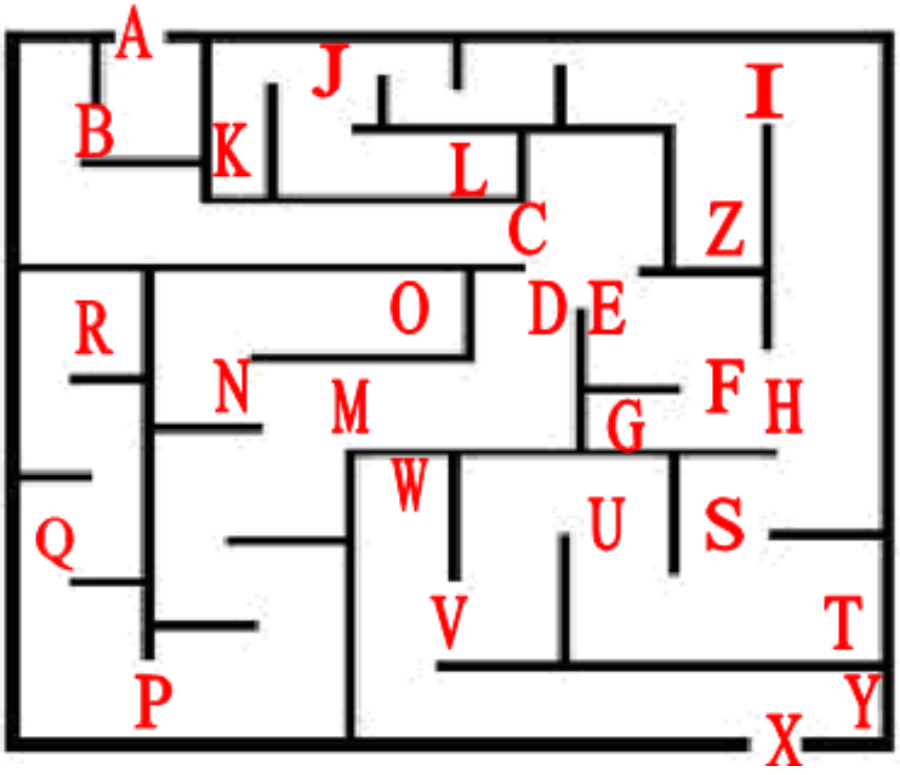

Рисунок 1.17 – Схема лабіринту

Форма звіту: Маршрути лабіринту відтворити у вигляді списку вершин. Простір станів відтворити у вигляді графа

1.2.6 Лабораторна робота №7. Пошук в ширину

МЕТА: ознайомити з алгоритмом пошуку в ширину, навчити застосовувати даний алгоритм для вирішення завдань пошуку в просторі станів, аналізувати його ефективність.

ЗАДАЧІ:

1. розглянути приклади завдань, для вирішення яких використовується пошук вширину;

2. розглянути алгоритм пошуку в ширину на просторі станів, представленого увигляді графа (дерева пошуку);

3. самостійно скласти дерево пошуку, використовуючи алгоритм пошуку в ширину.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Непоінформовані пошук (також сліпий пошук, метод грубої сили, англ. Uninformed search, blind search, brute-force search) - стратегія пошуку рішень у просторі станів, при якому не використовується додаткова інформація про стани, крім тієї, яка представлена у визначенні завдання.

В результаті неінформованого пошуку вдається сформувати наступників і відрізняти цільовий стан від нецільового.

Пошук в ширину працює шляхом послідовного перегляду окремих рівнів графа, починаючи з вузла-джерела u.

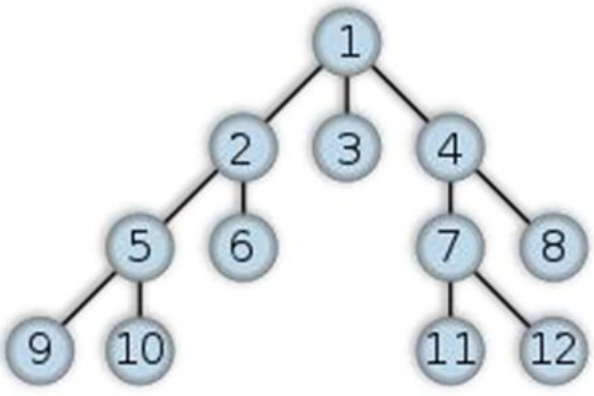

Рисунок 1.18 - Схема пошуку в ширину

Розглянемо всі ребра *(u,v),* що виходять з вузла *u*. Якщо черговий узол є цільовим вузлом, то пошук завершується. В іншому випадку вузол додається в чергу. Після того, як будуть перевірені всі ребра, що виходять з вузла *u*, з черги витягується наступний вузол *v*, і процес повторюється (рис.1.18).

Стратегія пошуку в ширину передбачає перехід в першу чергу до вершин, найближчий до стартової вершини. Внаслідок процес пошуку має тенденцію розвиватися більше в ширину, ніж в глибину (рис.1.19).

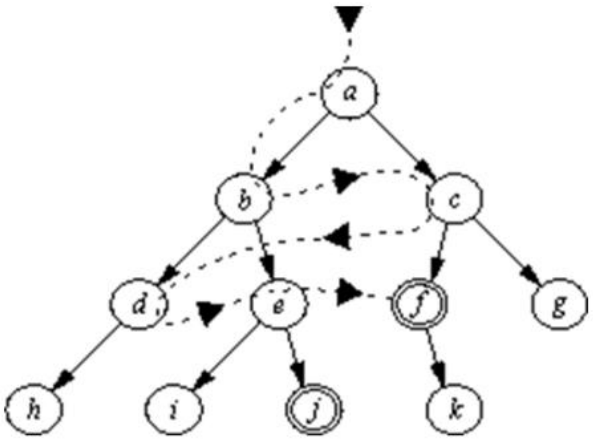

Рисунок 1.19 - Послідовність відвідування вершин за пошуком в ширину

Простий простір станів: a —стартова вершина, f і j — цільові вершини. Застосування стратегії пошуку в ширину дає наступний порядок проходу по вершинах: a, b, c, d, e, f. Більш короткий рішення [a, c, f] знайдено раніше, ніж довший [a, b, e, j]

Пошук в ширину реалізується не так легко, як пошук в глибину. Причина полягають у тому, що нам доводиться зберігати всі безліч альтернативних вершин-кандидатів, а не тільки одну вершину, як при пошуку в глибину. Більше того, якщо ми бажаємо отримати за допомогою процесу пошуку вирішальний шлях, то одного безлічі вершин недостатньо. Тому ми будемо зберігати не безліч вершин-кандидатів, а безліч шляхів-кандидатів. Таким чином, ланцюг істинна тільки тоді, коли існує шлях з безлічі кандидатів Шляху, який може бути продовжений аж до цільової вершини. Цей продовжений шлях і є розв’язанням задачі. Для того, щоб виконати пошук в

ширину при заданій множині шляхів- кандидатів, потрібно дотримуватись наступних правил:

− якщо голова першого шляху — це цільова вершина, то взяти цей шлях як рішення, інакше

− видалити перший шлях з безлічі кандидатів і породити безліч всіх можливих продовжень цього шляху на один крок; безліч продовжень додати в кінець безлічі кандидатів, а потім виконати пошук в ширину з отриманим новим безліччю.

*Приклад 1 «Гра в «Вісімки».*Початковий стан:

2	8	3
1	6	4
7		5

Переміщаючи за хід одну з «фішок» вліво, вправо, вгору або вниз привестиігрове поле до кінцевого стану:

1	2	3
8		4
7	6	5

*Розв'язання:*Зобразимо можливі варіанти першого ходу

2	8	3
1	6	4
	7	5

2	8	3
1		4
7	6	5

2	8	3
1	6	4
7	5	

Зобразимо можливі варіанти другого і подальших ходів у вигляді графа (рис.1.20)

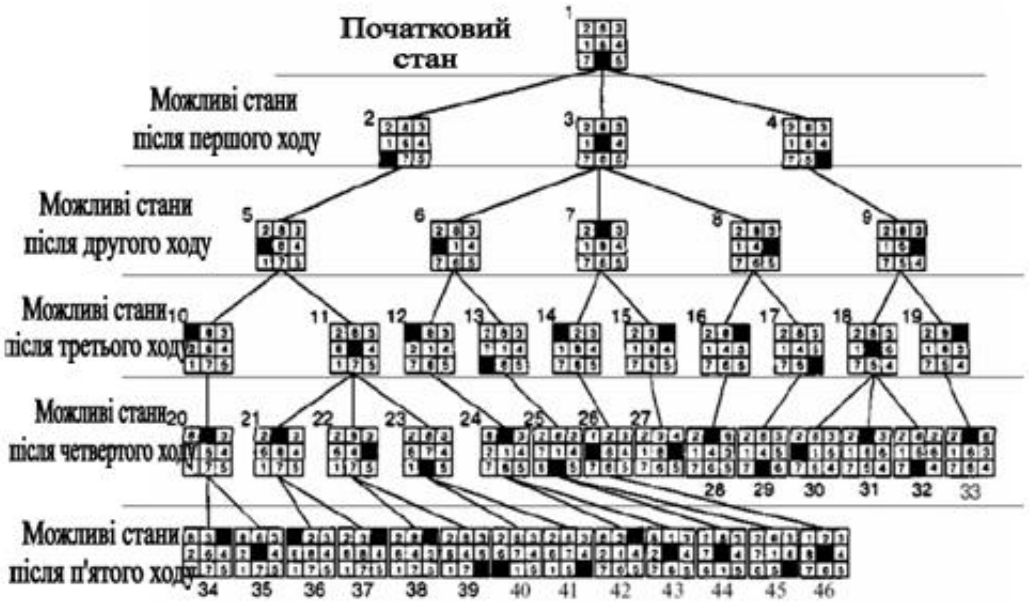

Рисунок 1.20 - Граф простору станів при пошуку в ширину

Як видно на малюнку, побудова графа завершилась на 46 вузлі, який є цільовимстаном. Шлях: 1-3-7-14-26-46.

Переваги і недоліки пошуку в глибину

Перевагою пошуку в ширину є можливість, у процесі перебору станів, відшукати саме просте (коротке) рішення, якщо воно існує. Однак у випадку високого коефіцієнта розгалуження можлива ситуація «комбінаторного вибуху», що приводить до істотного збільшення часу пошуку рішення, що є *недоліком пошуку в ширину*

Питання для самоконтролю:

1. Що називають «поінформованим пошуком»?
2. Що називають «пошуком в ширину»?
3. Які існують переваги і недоліки пошуку в ширину?
4. Порівняйте алгоритм пошуку в ширину і алгоритм пошуку у глибину.

Завдання для самостійного виконання:

Між населеними пунктами A, B, C, D, E, F побудовані дороги (рис. 1.21) Визначтедовжину найкоротшого шляху між пунктами A і F.

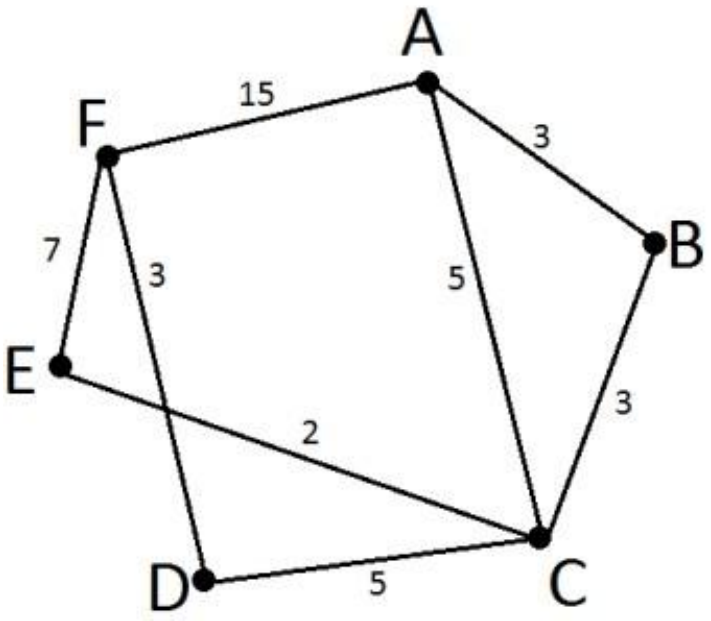

Рисунок 1.21 - Схема доріг

Форма звіту: Всі можливі маршрути представити у вигляді списку вершин, отриманого за допомогою пошуку в ширину. Простір станів представити у вигляді графа.

1.3 Нечітке логічне виведення

Невизначеності в ЕС та проблеми, що з ними пов'язані

У житті часто доводиться оцінювати гіпотези для яких є неповна або недостатня інформація. Іноді важко зробити точні оцінки, але, не дивлячись на невизначеність ми приймаємо розумні рішення. Щоб ЕС були корисними, вони теж повинні вміти це робити. Класичним прикладом цього завдання є медична діагностика. Завжди існують деякі сумніви в чіткості прояву симптомів того або іншого захворювання. Сумніви в наявності у пацієнта конкретного захворювання зберігаються навіть в тому випадку, коли всі його симптоми чітко виражені.

Як же проявляється і враховується невизначеність в експертних системах? Розглянемо найпростішу ситуацію. Нехай використовується правило

якщо (A), то (B)

і припустимо ніякі інші правила і посилки не мають відношення до даної ситуацій. Де ж виникає невизначеність? В ЕС вона може бути двох типів:

• невизначеність в істинності самої посилки (наприклад, якщо ступінь впевненості в тому, що А істинно становить 90%, то які значення прийме В)

• невизначеність самого правила (наприклад, ми можемо сказати, що в більшості випадків, але завжди, якщо є А, тобто також і В)

Ще більш складна ситуація виникає в разі, якщо правило має вигляд:

якщо (А и В), то С

де ми можемо з деякою мірою бути впевнені як в істинності кожної з посилок (А, В), а тим більше їх спільного прояви, так і в істинності самого висновку. Існують чотири важливі проблеми, які виникають при проектуванні і створенні ЕС з невизначеними знаннями:

• Як кількісно виразити ступінь визначеності при встановленні істинності (або хибності) деякої частини даних?

• Як висловити ступінь підтримки укладення конкретної посилкою?

• Як використовувати спільно дві (або більше) посилки, незалежно впливають на висновок?

• Як бути в ситуації, коли потрібно обговорити ланцюжок виведення для підтвердження укладення в умовах невизначеності?

Перш за все розглянемо можливості використання теорії ймовірності при введенні в умовах невизначеності.

Теорія суб’єктивних ймовірностей

Основне поняття ймовірності настільки природно, що воно відіграє значну роль в повсякденному житті. Розмови, що стосуються ймовірності дощу або хорошого врожаю в городі часто зустрічаються в нашому житті. Поняття ймовірності було розроблено кілька століть назад. Але вже тисячі років людина використовує такі слова, як "може бути", "шанс", "удача" або інші їх еквіваленти в розмовній мові.

Однак математична теорія ймовірностей була сформульована відносно недавно (близько 1660 року). Імовірність події класично визначається як відношення випадків в яких дана подія відбувається до загальної кількості спостережень.

Однак можливі й інші визначення. В даний час існує декілька інтерпретацій теорії ймовірностей. Розглянемо три найбільш домінуючих погляду.

Об'єктивістський погляд. Полягає в тому, що розглядає ймовірність відносини результатів до всіх спостереженнями протягом тривалого часу. Іншими словами цей підхід заснований на законі великих чисел, що гарантує те, що при наявності досить великої кількості спостережень частота випадків, цікавить події буде прагнути до об'єктивної ймовірності.

Персоніфікований, суб'єктівістській або заснований на судженнях погляд. Полягає в тому, що імовірнісна міра розглядається як ступінь довіри того, як окрема особистість судить про істинність деякого висловлювання. Цей погляд постулює, що дана особа має в певному сенсі ставлення до цієї події. Але це не заперечує можливості того, що дві прийнятні особистості можуть мати різні ступені довіри для одного і того ж судження. Термін "байесовкій" часто використовується як синонім суб'єктивної ймовірності.

Необхідний або логічний. Характеризується тим, що імовірнісна міра розширюється на безліч тверджень, що мають логічний зв'язок таку, що істинність одного з них може виводитися з іншого. Іншими словами ймовірність вимірює ступінь доказовості логічно вивіреного ув'язнення. Такий погляд можна розглядати як розширення звичайної логіки.

Ці імовірнісні інтерпретації використовують і різні схеми виведення. Однак існує всього дві школи імовірнісних розрахунків: школа Паскаля (або загальноприйнята), школа Бекона (або індуктивна). Розрахунки за Паскалем використовують байєсовські правила для перевірки і обробки заходів довіри. Обчислення за Беконом використовують правила логіки для доведення або

спростування гіпотез. Таким чином, загальноприйняті ймовірності (за Паскалем) не можуть бути отримані з індуктивних ймовірностей (за Беконом) і, навпаки. Об'єктивістський і суб'єктивний погляди використовують розрахунки за Паскалем. Ті, хто підтримують логічні висновки, використовують розрахунки за Беконом.

Існують ЕС, побудовані на обох з цих напрямків. Однак в ЕС бази знань накопичують людські знання, тому для представлення знань експертів з урахуванням ймовірностей найбільш підходящими є інтерпретація на основі суб'єктивних довір. В результаті чого і більшість сучасних ЕС, що використовують теорію ймовірностей, є "байесовськими".

Байесовське оцінювання

Перед тим, як ввести теорему Байеса розглянемо деякі фундаментальні поняття теорії ймовірностей. Нехай А деяка подія реального світу. Сукупність усіх елементарних подій називається вибірковим простором або простір подій (Ω). Імовірність події А, позначається *p(A)* і кожна імовірнісна функція р повинна задовольняти трьом аксіомам:

1. Імовірність будь-якої події А є невід’ємною, тобто

$$p(A) \geq 0 \quad для \quad \forall A \in \Omega$$

2. Ймовірність всіх подій вибіркового простору дорівнює 1, тобто

$$p(\Omega) = 1.$$

3. Якщо k подій A_1, A_2, … , A_k є взаємно незалежними (тобто не можуть підійти одночасно), то ймовірність, принаймні, одного з цих подій дорівнює сумі окремих ймовірностей, або

$$p(A_1 \cup A_2 \cup \quad ... \quad \cup A_k) \quad = \quad \sum_{i=1}^{k} p(A_i)$$

Аксіоми 1 і 2 можна об'єднати, що дає $1 \geq p(A) \geq 0 \quad для \quad \forall A \in \Omega$.

Це твердження показує, що ймовірність будь-якої події знаходиться між 0 і 1. За визначенням, коли p (A) = 0, то подія A ніколи не станеться. У тому випадку і коли p (A) = 1, то подія A має відбутися обов'язково.

Доповнення до A, що позначається (¬A), містить сукупність всіх подій в Ω за винятком A. Оскільки A та ¬A є взаємонезалежними (тобто A ∪ ¬A = Ω), то з аксіоми 3 слідує

$$p(A) + p(\neg A) = p(A \square\square \neg A) = p(\square) = 1 .$$

Переписуючи це рівність у вигляді p (¬A) = 1 - p (A), ми отримує шлях для отримання p (¬A) з p (A).

Припустимо тепер, що B ∈ Ω деяке інше подія. Тоді ймовірність того, що станеться A за умови, що сталося B записується у вигляді p (A | B) і називається умовною ймовірністю події A при заданому подію B.

Імовірність того, що обидві події A і B відбудуться p (A∩B) називається спільної ймовірністю подій A і B. Умовна ймовірність p (A | B) дорівнює відношенню спільної ймовірності p (A∩B) до ймовірності події B, за умови, що вона не дорівнює 0, т. е.

$$p(A|B) = \frac{p(A \cap B)}{p(B)}$$

Аналогічно умовна ймовірність події B за умови A, позначається p (B | A) дорівнює:

$$p(B|A) = \frac{p(B \cap A)}{p(A)}$$

Таким чином,

$$p(B \cap A) = p(B|A) \times p(A) .$$

Так, як спільна ймовірність комутативна (тобто від перестановки місць сума не змінюється), то

$$p(A \cap B) = p(B \cap A) = p(B|A) \times p(A) .$$

Підставляючи це рівність в раніше отриманий вираз для умовної ймовірності $p(A|\ B)$ отримаємо правило Байеса

$$p(A|B) = \frac{p(B|A) \times p(A)}{p(B)} \ .$$

У ряді випадків наше знання того, що відбулася подія B, не впливає на ймовірність події A (або навпаки A на B). Іншими словами, ймовірність події A не залежить від того, що сталося чи ні подія B, так що

$$p(A \mid B) = p(A) \quad \text{та} \quad p(B \mid A) = p(B) \ .$$

У цьому випадку говорять, що події A і B є незалежними.

Особливості роботи з нечіткими знаннями

При формалізації знань існує проблема, що утрудняє використання традиційного математичного апарата. Це проблема опису понять, що оперують якісними характеристиками об'єктів (багато, мало, сильний, дуже сильний і т.п.). Ці характеристики звичайно розмиті й не можуть бути однозначно інтерпретовані, однак містять важливу інформацію (наприклад, "одним з можливих ознак грипу є висока температура").

Крім того, у задачах, розв'язуваних інтелектуальними системами, часто доводиться користуватися неточними знаннями, які не можуть бути інтерпретовані як повністю щирі або помилкові (логічні true/false або 0/1). Існують знання, вірогідність яких виражається деякою проміжною цифрою, наприклад 0,7. Як, не руйнуючи властивості розмитості й неточності, представляти подібні знання формально? Для дозволу таких проблем на початку 70-х років XX століття американський математик Лотфи Заде запропонував формальний апарат нечіткої (fuzzy) алгебри й нечіткої логіки. Пізніше цей напрямок одержав широке поширення і поклало початок однієї з галузей ШІ за назвою м'які обчислення (soft computing).

Л. Заде ввів одне з головних понять у нечіткій логіці - поняття лінгвістичної змінної. Лінгвістична змінна (ЛП) — це змінна, значення якої визначається набором вербальних (тобто словесних) характеристик деякої властивості.

Наприклад, ЛП "ріст" визначається через набір {карликовий, низький, середній, високий, дуже високий}.

Основи теорії нечітких множин

Значення лінгвістичної змінної (ЛП) визначаються через так називані нечіткі множини (НМ), які у свою чергу визначені на деякому базовому наборі значень або базовій числовій шкалі, що має розмірність. Кожне значення ЛП визначається, як нечітка множина (наприклад, НМ "низький ріст").

Нечітка множина визначається через деяку базову шкалу В и функцію приналежності НМ — μ (х), *xi* В Д приймаючого значення на інтервалі [0; 1].

Таким чином, нечітка множина В — це сукупність пар виду (х, μ (х)). Часто зустрічається й такий запис:

$B=\Sigma i=1n\ xi\mu(\ xi\ ),$

де *xi* — *i-me* значення базової шкали.

Функція приналежності визначає суб'єктивний *ступінь упевненості* експерта в тім, що дане конкретне значення базової шкали відповідає обумовленому НМ.

Цю функцію не варто плутати з імовірністю, що носить об'єктивний характер і підкоряється іншим математичним залежностям.

Наприклад, для двох експертів визначення НМ "висока" для ЛП "ціна автомобіля" в умовних одиницях може істотно відрізнятися залежно від їх соціального й фінансового становища.

"Висока_ціна_автомобіля_1" = {50000/1 + 25000/0.8 +10000/0.6 + 5000/0.4}

"Висока_ціна_автомобіля_2" = {25000/1 + 10000/0.8 +5000/0.7 + 3000/0.4}

Нехай перед нами коштує задача інтерпретації значень ЛП "вік", таких як "молодий" вік, "похилий" вік або "перехідний" вік. Визначимо "вік" як ЛП (рис. 1.22).

Тоді "молодий", "похилий", "перехідний" будуть значеннями цієї лінгвістичної змінної. Більш повно, базовий набір значень ЛП "вік" наступний:

B = {дитячий, дитячий, юний, молодий, зрілий,

похилий, старечий}.

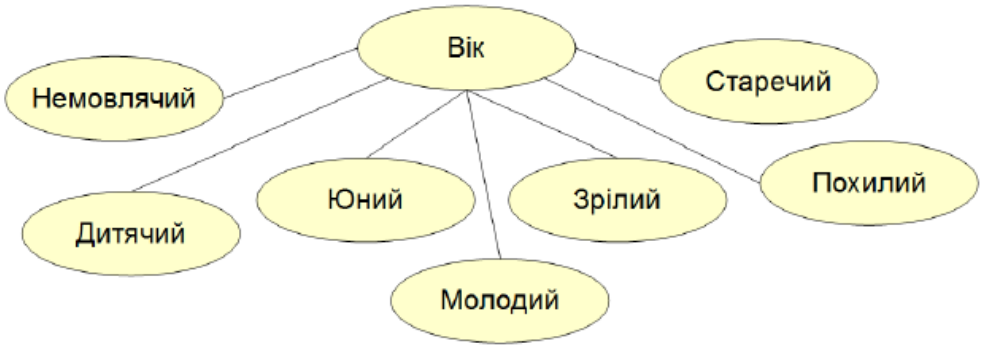

Рисунок 1.22 - Лінгвістична змінна "вік" і нечіткі множини, що визначають її значення

Для ЛП "вік" базова шкала - це числова шкала від 0 до 120, що позначає кількість прожитого років, а функція приналежності визначає, наскільки ми впевнені в тім, що дана кількість років можна віднести до даної категорії віку. На мал. 1 відбито, як ті самі значення базової шкали можуть брати участь у визначенні різних НМ.

Наприклад, визначити значення НМ "дитячий" можна так:

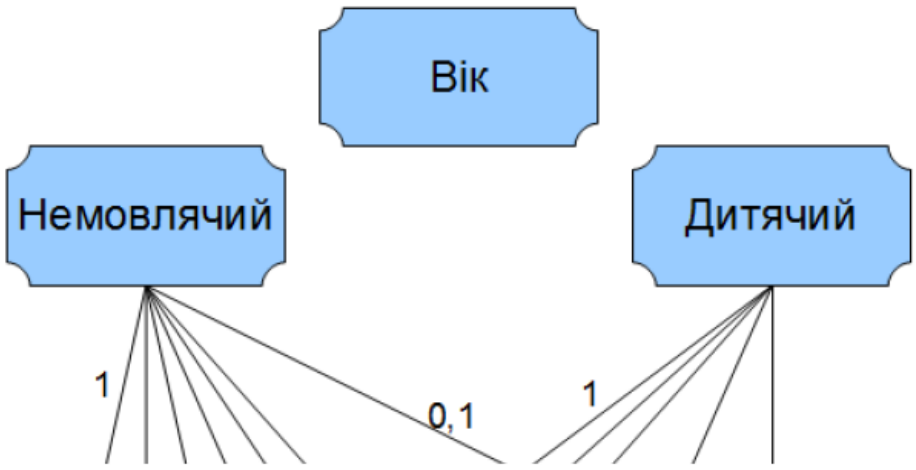

Рисунок 1.23 - Формування нечітких множин

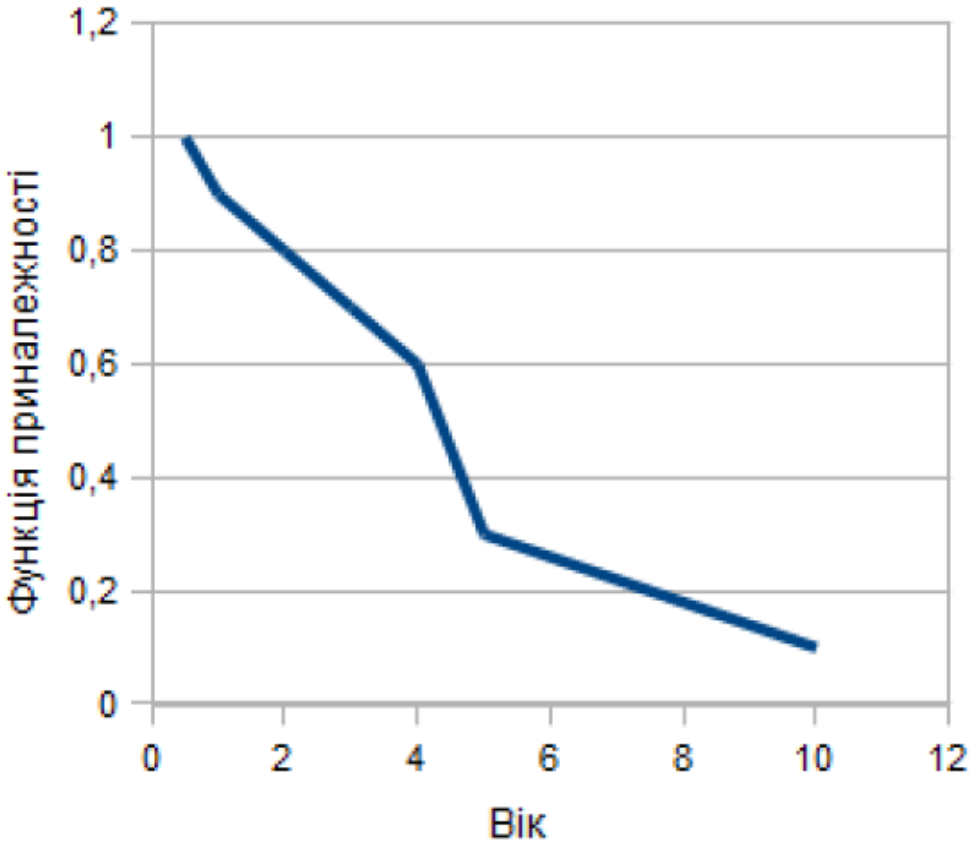

Рисунок 1.24 - Графік функції приналежності нечіткій множині «немовлячий вік»

Рис. 1.24 ілюструє оцінку НМ якимсь усередненим експертом, що дитини до напівроку з високим ступенем упевненості відносить до дитин (μ= 1). Діти до чотирьох років зараховуються до дитин теж, але з меншим ступенем упевненості (0,5 <μ< 0,9), а в десять років дитини називають так тільки в дуже рідких випадках - приміром, для дев'яностолітньої бабусі й 15 років може вважатися дитинством. Таким чином, нечіткі множини дозволяють при визначенні поняття враховувати суб'єктивні думки окремих індивідуумів.

Операції з нечіткими знаннями

Для операцій з нечіткими знаннями, вираженими за допомогою лінгвістичних змінних, існує багато різних способів. Ці способи є в основному евристиками.

Ми не будемо зупинятися на цьому питанні докладно, укажемо лише для приклада визначення декількох операцій. Наприклад, операція "АБО" часто задається так :

$\mu(x)=\max(\mu1(x) , \mu2(x)$ (так називана логіка Заде)

або так:

$\mu (x)=\mu1 (x)+\mu2 (x-\mu1(x)*\mu2(x)$ (імовірнісний підхід).

Посилення йди ослаблення лінгвістичних понять досягається введенням спеціальних квантифікаторів. Наприклад, якщо поняття "старечий вік" визначається як то поняття "дуже старечий вік" розпізнається як *con (A) = A2* $=\Sigma iXi\mu i^2$ т. е. дуже старечий вік визначиться так.

Для висновку на нечітких множинах використаються спеціальні відносини й операції над ними.

Одним з перших застосувань теорії НМ

{60*0,6+70*0,8+80*0,9+90*1 }

{60*0,36+70*0,64+80*0,81+90*1}

стало використання коефіцієнтів упевненості для висновку рекомендацій медичної системи MYC1N. Цей метод використає кілька евристичних прийомів. Він став прикладом обробки нечітких знань, що вплинули на наступні системи.

У цей час у більшість інструментальних засобів розробки систем, заснованих на знаннях, включені елементи роботи із НМ, крім того, розроблені спеціальні програмні засоби реалізації так називаного нечіткого висновку, наприклад "оболонка" FuzzyCLIPS.

1.3.1 Лабораторна робота №8. Формування нечітких функцій приналежності

МЕТА: навчити формувати нечіткі функції приналежності

ЗАДАЧІ:

1. розглянути поняття «нечітка логіка», «нечітка множина»;
2. розглянути види функцій належності;
3. розглянути приклади використання нечітких функцій приналежності.

ТЕОРЕТИЧНІ ВІДОМОСТІ

На практиці зручно використовувати ті функції приналежності, які допускають аналітичне подання у вигляді деякої простої математичної функції. Це спрощує не тільки відповідні чисельні розрахунки, але і скорочує обчислювальні ресурси, необхідні для зберігання окремих значень цих функцій приналежності.

Класифікація функцій приналежності

Кусково-лінійні приналежності:

В якості першого типу функцій належності розглянемо функції, які, як випливає з їх назви, складаються з відрізків прямих ліній, утворюючи безперервну або кусочно-безперервну функцію. Найбільш характерним прикладом таких функцій є "трикутна" (рис.1.25(а)) і "трапецивидна" (рис.1.25(б)) функції приналежності. У нашому випадку кожна з цих функцій задана науніверсумі X = [0, 10], в якості якого обраний замкнутий інтервал дійсних чисел. У загальному випадку вибір універсуму може бути довільним, і не обмежений ніякими правилами.

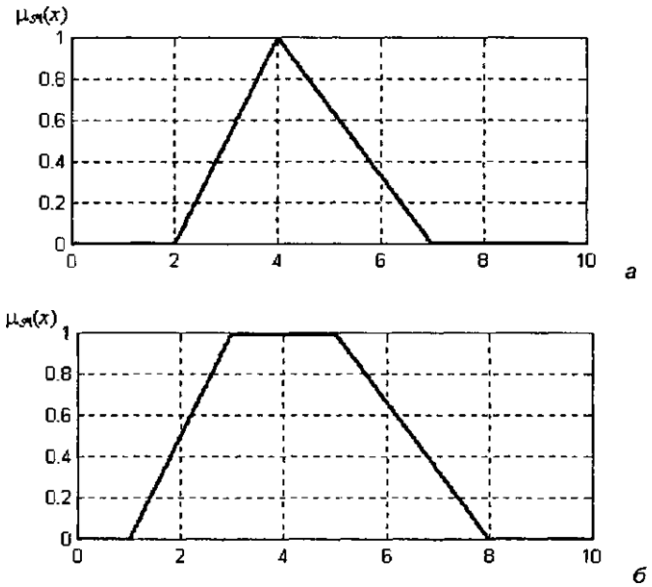

Рисунок 1.25 - Графіки функцій приналежності трикутної (а) і трапецієподібної (б) форми

Перша з цих функцій приналежності в загальному випадку може бути задана аналітично наступним виразом:

$$f_{\Delta}(x;,a,b,c)=\begin{Bmatrix}0, & x\le a\\ \frac{x-a}{b-a}, & a\le x\le b\\ \frac{c-x}{c-b}, & b\le c\le x\\ 0, & c\le x\end{Bmatrix}$$

де a, b, c — деякі числові параметри, що приймають довільні дійсні значення і впорядковані відношенням: **a≤b≤c**.

Стосовно конкретної функції, зображеної на мал.1, а, значення параметрів рівні: a = 2, b = 4, c = 7. Як неважко помітити, параметри a і c характеризують основу трикутника, а параметр b — його вершину. Як можна помітити, ця функція приналежності породжує нормальну опуклу унімодальну нечітку множину з носієм — інтервалом (a, c), межами (a, c) \ {b}, ядром {b} і модою b.

Трапецієвидна функція приналежності в загальному випадку може бути задана аналітично наступним виразом:

$$f_{\mathrm{T}}(x;,a,b,c,d)=\begin{Bmatrix}0, & x\le a\\ \frac{x-a}{b-a}, & a\le x\le b\\ 1, & b\le x\le c\\ \frac{d-x}{d-c}, & c\le x\le d\\ 0, & d\le x\end{Bmatrix}$$

де a, b, c, d — деякі числові параметри, що приймають довільні дійсні значенняі впорядковані відношенням: **a≤b≤c≤d**.

Стосовно конкретної функції, зображеної на мал.1 б, значення параметрів рівні: a = 1, b = 3, c = 5, d = 8. Як неважко помітити, параметри a і d характеризують нижня основа трапеції, а параметри b і c — верхнє підставу трапеції. При цьомудана функція приналежності породжує нормальну опуклу нечітку множину з носієм — інтервалом (a, d), межами (a, b) (c, d) і ядром [b, c].

Ці функції використовуються для завдання таких властивостей множин, які характеризують невизначеність типу: "приблизно дорівнює", "середнє значення", "розташований в інтервалі", "подібний до об'єкту", "схожий на предмет" та ін. Вони також служать для представлення нечітких чисел і інтервалів, які будуть розглянуті далі.

Z-подібні і S-подібні функції приналежності отримали свою назву по виду кривих, які представляють їх графіки. Перша з функцій цієї групи називається Z-подібної кривої або сплайн-функцією і в загальному випадку може бути задана аналітично наступним виразом:

$$f_{z_1}(x;,a,b)=\begin{Bmatrix}1, & x\le a\\ \frac{1}{2}+\frac{1}{2}\cos\left(\frac{x-a}{b-a}\pi\right), & a\le x\le b\\ 0, & x>b\end{Bmatrix}$$

де a, b - деякі числові параметри, що приймають довільні дійсні значення і впорядковані відношенням: a<b. Графік цієї функції для деякої нечіткої множини **Ã** і універсуму X = [0, 10] зображено на рис.1.26, при цьому

$$f_{z_2}(x;,a,b)=\begin{Bmatrix}1, & x\le a\\ 1-2\left(\frac{x-a}{b-a}\right)^2, & a<x\le\frac{a+b}{2}\\ 2\left(\frac{b-x}{b-a}\right)^2, & \frac{a+b}{2}<x<b\\ 0, & b\le x\end{Bmatrix}$$

значення параметрів відповідно рівні a = 3, b = 6. Сплайн-функція може бути також задана іншим виразом:

де a, b — деякі числові параметри, що приймають довільні дійсні значення і впорядковані ставленням: a <b. Графік цієї функції для деякого нечіткої множини **Ã** і універсуму X = [0, 10] зображений на рис.1.26 б, при цьому значення параметрів відповідно рівні a = 3, b = 6.

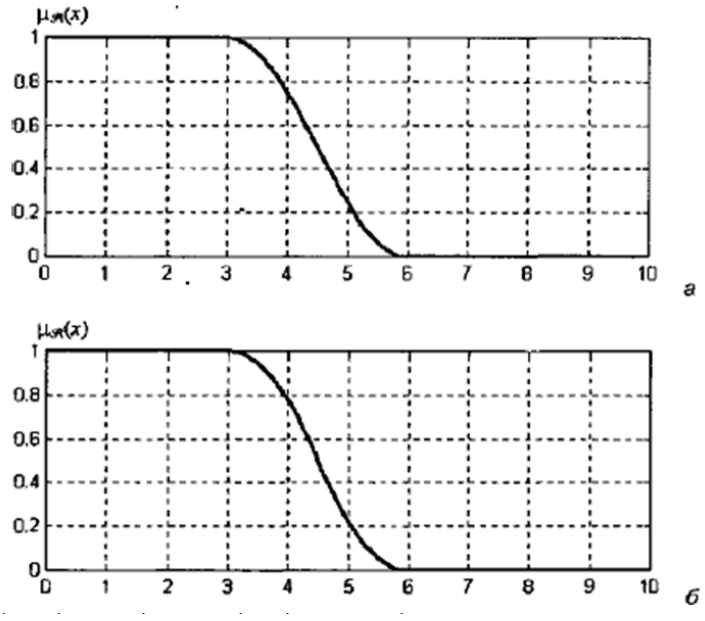

Рисунок 1.26 - Графіки Z-образних функцій приналежності fz1 і fz2 для значень параметрів a = 3, b = 6»

Дані функції приналежності породжують нормальні опуклі нечіткі множини з ядром (-∞, a] і носієм (-∞, b). Ці функції використовуються для представлення таких властивостей нечітких множин, які характеризуються невизначеністю типу: "мала кількість", "невелике значення", "незначна величина", "низька собівартість продукції", "низький рівень цін або доходів", "низька процентна ставка" та багатьох інших. Спільним для всіх таких ситуацій є слабка ступінь прояву того чи іншого якісного або кількісного ознаки. Особливість нечіткого моделювання при цьому полягає в поданні відповідних нечітких множин за допомогою незростаюча (монотонно убувають) функцій приналежності. Друга з функцій даної групи називається S-подібної кривої або сплайн-функцією і в загальному випадку може бути задана аналітично наступним виразом:

$$f_{S1}(x;,a,b)=\begin{Bmatrix} 0, & x<a \\ \frac{1}{2}+\frac{1}{2}\cos\left(\frac{x-b}{b-a}\pi\right), & a\le x\le b \\ 1, & x>b \end{Bmatrix}$$

де a, b - деякі числові параметри, що приймають довільні дійсні значення і впорядковані відношенням: a <b. Графік цієї функції для деякого нечіткої множини **Ã** і універсуму X = [0, 10] зображено на рис.1.27 (а), при цьому значення параметрів відповідно рівні a = 3, b = 6.

Дані функції приналежності породжують нормальні опуклі нечіткі множини з ядром [b, + ∞) і носієм (a, + ∞).

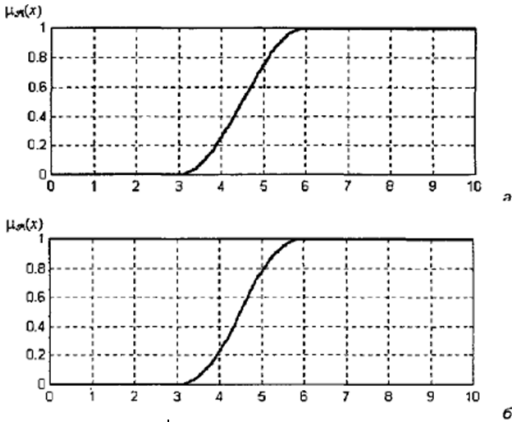

Рисунок 1.27 - Графіки S-образних функцій приналежності fS1 і fS2 для значень параметрів a = 3, b = 6

До типу S-образних і одночасно Z-образних функцій приналежності може бути віднесена так звана сигмоїдальна функція (сигмоид), яка в загальному випадку задається аналітично наступним виразом:

$$f_{S3}(x; a, b) = \frac{1}{1 + e^{-a(x-b)}}$$

тут a, b- деякі числові параметри, що приймають довільні дійсні значення і впорядковані ставленням: a <b, a e - основа натуральних логарифмів, яке ініціює завдання відповідної експоненційної функції. При цьому у разі a> 0 може бути отримана S-подібна функція приналежності, а в разі a <0 - Z-подібна функція приналежності. Графіки цієї функції для деякого нечіткої множини **Ã** і універсуму X = [0, 10] зображені на рис.1.28. При цьому S-подібної функції приналежності відповідають значення параметрів a = 3, b = 6 (рис. 1.28, а), а Z- подібної функції приналежності відповідають значення параметрів a=-3, b=6 (рис.1.28б). Дані функції приналежності породжують субнормального опуклі нечіткі множини з носієм і кордоном **R** і точкою переходу b.

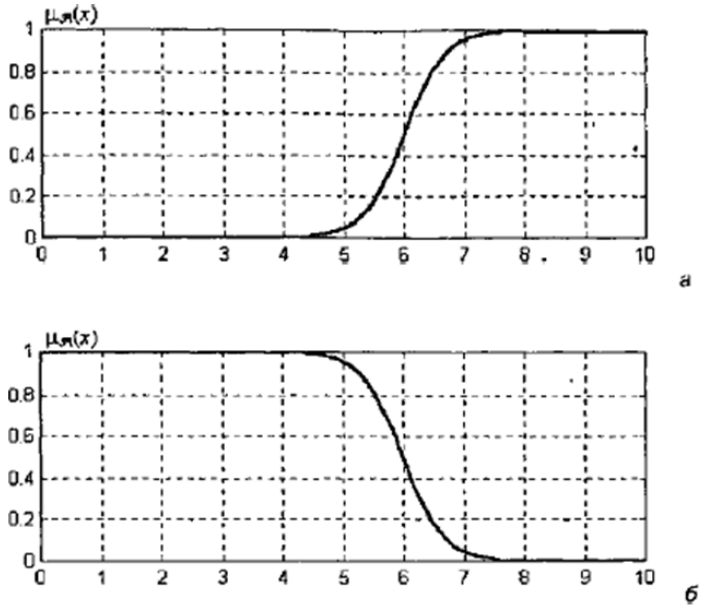

Рисунок 1.28 - Графіки сигмоїдальної функції приналежності fS3 для значень параметрів a= 3, b = 6(а) і a = -3, b = 6 (б)

Розглянуті S-подібні функції використовуються для представлення таких нечітких множин, які характеризуються невизначеністю типу: "велика кількість", "велике значення", "значна величина", "високий рівень доходів і цін", "висока норма прибутку", "висока якість послуг "," високий сервіс обслуговування "та багатьох інших. Спільним для всіх таких ситуацій є висока ступінь прояву того чи іншого якісного або кількісного ознаки. Особливість нечіткого моделювання при цьому полягає в поданні відповідних нечітких множин за допомогою неубиваючих (монотонно зростаючих) функцій приналежності. В якості окремих випадків Z- і 5-образних кривих зручно розглядати так звану лінійну Z-подібну функцію (рис.1.29а) і лінійну S-подібну функцію (рис.1.29б). Перша з цих функцій в загальному випадку може бути заданааналітично наступним виразом:

$$f_{\downarrow}(x;a,b)=\begin{Bmatrix}1, & x\le a\\ \dfrac{b-x}{b-a}, & a<x<b\\ 0, & b\le x\end{Bmatrix}$$

де a, b — деякі числові параметри, що приймають довільні дійсні значення і впорядковані ставленням: a <b. Графік цієї функції для деякого нечіткої множини **Ã** і універсуму X = [0,10] зображений на мал.5а, при цьому

$$f_{\uparrow}(x;a,b)=\begin{Bmatrix}0, & x\le a\\ \dfrac{x-a}{b-a}, & a<x<b\\ 1, & b\ge x\end{Bmatrix}$$

значення параметрів відповідно дорівнюють a = 3, b = 6. Друга з цих функцій в загальному випадку може бути задана аналітично наступним виразом:

де a, b — деякі числові параметри, що приймають довільні дійсні значення і впорядковані ставленням: a <b. Графік цієї функції для деякого нечіткої множини **Ã** і універсуму X = [0, 10] зображений на рис.1.29б, при цьому значення параметрів відповідно також дорівнюють a = 3, b = 6.

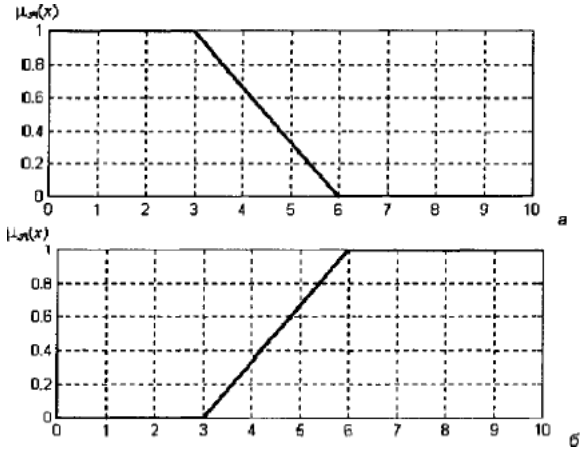

Рисунок 1.29 - Графіки лінійної Z-подібної функції (а) і лінійної S-подібної функції (б) приналежності для значень параметрів a = 3, b = 6»

Дані функції приналежності породжують нормальні опуклі нечіткі множини з межами (a, b). Слід зауважити, що дані лінійні Z- і S-подібні функції можуть бути використані для побудови розглянутих вище трикутної і трапецієподібної функцій приналежності. Зокрема трикутна функція приналежності виходить як композиція лінійної Z-подібною і лінійної S-подібної функцій за такою формулою:

$$f_{\Delta}(x; a, b, c) = \min_{x \in X}\{f_{\uparrow}(x; a, b), f_{\downarrow}(x; b, c)\}$$

де a, b, c — деякі числові параметри, що приймають довільні дійсні значення і впорядковані ставленням: a≤b≤c. У виразі вище використовується операція взяття мінімуму (позначена знаком min) з усіх значень, зазначених у фігурних дужках через кому. При цьому якщо відповідні функціональні значення залежать від деякої незалежної змінної (у нашому випадку від x), то під знаком мінімуму явно вказується діапазон або безліч значень цієї змінної (в нашому випадку - універсум). Трапецієвидна функція приналежності виходить як композиція двох лінійних Z-подібною і S-подібної функцій за такою формулою:

$$f_{T}(x; a, b, c, d) = \min_{x \in X}\{f_{\uparrow}(x; a, b), f_{\downarrow}(x; c, d)\}$$

де a, b, c, d - деякі числові параметри, що приймають довільні дійсні значення і впорядковані ставленням: a≤b≤c≤d.

П-подібні функції приналежності

До даного типу функцій належності можна віднести цілий клас кривих, які за своєю формою нагадують дзвін, згладжену трапецію або букву "П". Перша з подібних функцій так і називається — П-подібна функція, і в загальному випадку задається аналітично наступним виразом:

$$f_{\Pi}(x; a, b, c, d) = f_S(x; a, b) \cdot f_Z(x; c, d)$$

де a, b, c, d — деякі числові параметри, що приймають довільні дійсні значення і впорядковані ставленням: a≤b≤c≤d, а знак "●" позначає звичайне арифметичнедобуток значень відповідних функцій.

При цьому можуть бути використані будь-які з розглянутих вище Z- і S-образних функцій. Зокрема, якщо використовувати функції fSl і fZ1 то отримаємо П-функцію fП1, графік якої для деякого нечіткої множини і універсуму X = [0, 10] зображений на мал. 3.6а. При цьому значення параметрівдля функції fS1 рівні a = 1, b = 4, а для функції fZ1 - з = 5, d = 9. Якщо ж використовувати функції fS2 і fZ2, то отримаємо П-функцію fП2, графік якої для деякого нечіткої множини $\tilde{\mathbf{A}}$ і універсуму X = [0, 10] зображений на мал.6б для тих же значень параметрів. Очевидно, цей тип функцій приналежностіпороджує нормальні опуклі нечіткі множини з носієм (a, d) і ядром [b, c].

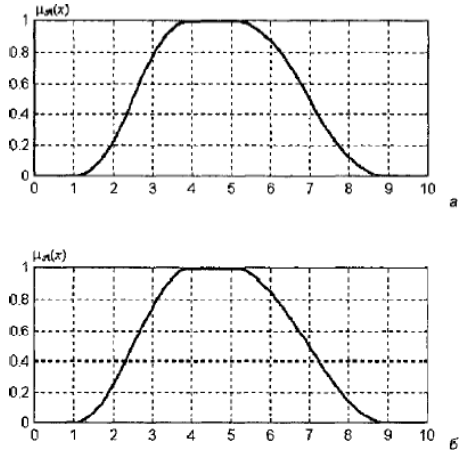

Рисунок 1.30 - Графіки П-подібних функцій приналежності fП1 (а) і fП2 (б) для значень параметрів a = 1, b = 4, c = 5, d = 9

Наступна функція цього типу П-подібних функцій визначається як добуток двох сигмоїдальних функцій і в загальному випадку може бути задана аналітично наступним виразом:

$$f_{П3}(x; a, b, c, d) = f_{S3}(x; a, b) \cdot f_{S3}(x; c, d),$$

де a, b, c, d - деякі числові параметри, що приймають довільні дійсні значення, причому a> 0, c <0, і впорядковані ставленням: a≤b <| c | ≤d. Знак "●" позначає арифметичне добуток значень відповідних функцій, а функція | x | - модуль дійсного числа. До П-подібним функцій відноситься також так звана *колоколоподібна (bell-shaped) функція,* яка в загальному випадку задається аналітично наступним виразом:

$$f_{П4}(x;, a, b, c, d) = \frac{1}{1 + \left|\frac{x - c}{a}\right|^{2b}},$$

де a, b, c — деякі числові параметри, що приймають довільні дійсні значення і впорядковані ставленням: a <b <c, причому параметр b> 0. Тут функція | x | позначає модуль дійсного числа

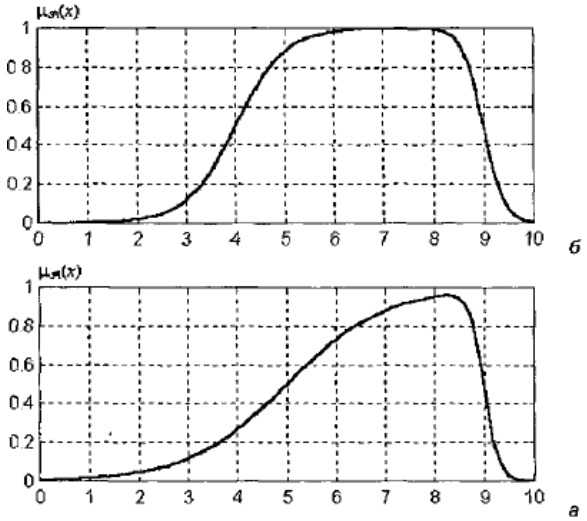

Рисунок 1.31 - Графіки П-подібних функцій приналежності fП3 для значень параметрів a = 1, b = 5, c= -7, d = 9 (а) і для значень параметрів a = 2, b = 4, c = -5, c / = 9 (б)

Нарешті, останньою з розглянутих функцій даного типу є добре відома в теорії ймовірностей функція щільності нормального розподілу в припущенні, яка в нашому випадку задається аналітично наступним виразом:

$$f_{П5}(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}.$$

де ϭ і з — числові параметри, при цьому квадрат першого з них ϭ2 в теорії ймовірностей називається *дисперсією розподілу*, а другий параметр с — *математичним очікуванням*. Вочевидь, ці останні типи функцій приналежностіпороджують нормальні опуклі нечіткі множини.

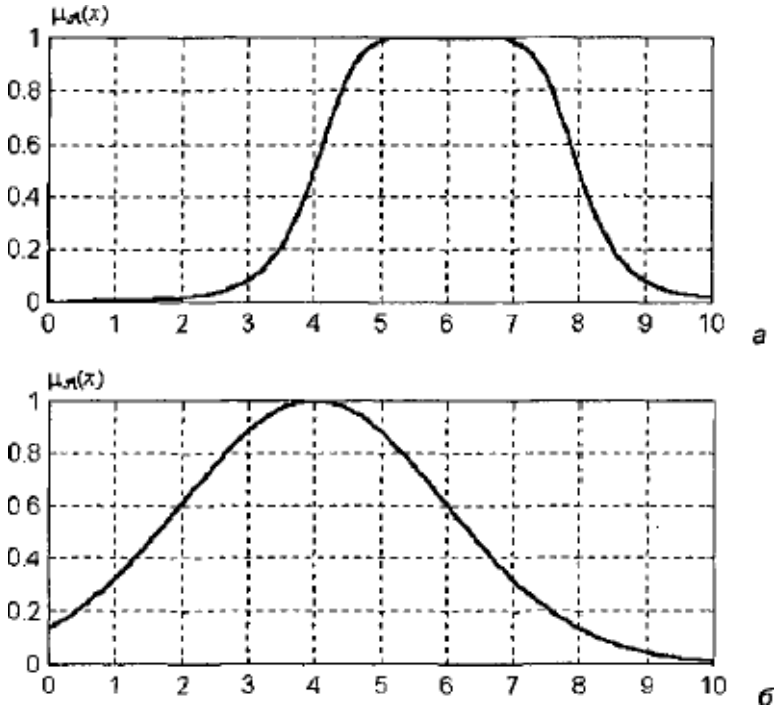

Рисунок 1.32 - Графіки П-подібних функцій приналежності fП4 для значень параметрів a = 2, b = 3, c-6 (а) і fП5 для значень параметрів з = 2, с = 4 (б)

Процес побудови або завдання нечіткої множини на основі деякого відомого заздалегідь кількісного значення вимірного ознаки отримав навіть спеціальну назву — *фазифікація* або приведення до нечіткості. Мова йде про те, що хоча іноді нам буває відомо деяке значення вимірної величини, ми визнаємо той факт, що це значення відомо неточно, можливо з погрішністю або випадковою помилкою. При цьому, чим менше ми впевнені в точності

вимірювання ознаки, тим більшим буде інтервал носія відповідного нечіткого безлічі. Слід пам'ятати, що в більшості практичних випадків абсолютна точність вимірювання є лише зручною абстракцією для побудови математичних моделей.

Саме з цієї причини фазифікації дозволяє більш адекватно представити об'єктивно присутню неточність результатів фізичних вимірювань.

Питання для самоконтролю:

1. Що називають «нечіткою логікою»?
2. Що називають «нечітким безліччю»?
3. У яких випадках використовуються «трикутні» і «трапецієподібні» функції?
4. У яких випадках використовується S-образні і Z-подібні функції?
5. Що називають «фазифікацією»?

Завдання для самостійного виконання:

Побудуйте графік функції приналежності нечітких множин:

1. висока вартість мобільного телефону;
2. гаряча кава;
3. освітленість приміщення;
4. порівняння з фотороботом;
5. недостатній рівень доходів;
6. висока тривалість життя.

Форма звіту: побудувати графіки відповідних функцій

1.3.2 Лабораторна робота №9. Операції над нечіткими множинами

МЕТА: навчити здійснювати операції над нечіткими множинами.

ЗАДАЧІ: ознайомити з логічними і арифметичними операціями; розглянути графічний вигляд операцій.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Операції з нечіткими знаннями

Для операцій з нечіткими знаннями, вираженими за допомогою лінгвістичних змінних, існує багато різних способів. Ці способи є в основному евристиками.

Ми не будемо зупинятися на цьому питанні докладно, укажемо лише для приклада визначення декількох операцій. Наприклад, операція "АБО" часто задається так :

$\mu(x)=\max(\mu1(x) , \mu2(x)$ (так називана логіка Заде)

або так:

$\mu (x)=\mu1 (x)+\mu2 (x-\mu1(x)*\mu2(x)$ (імовірнісний підхід).

Посилення йди ослаблення лінгвістичних понять досягається введенням спеціальних квантифікаторів. Наприклад, якщо поняття "старечий вік" визначається як то поняття "дуже старечий вік" розпізнається як *con (A) = A2* $=\Sigma iXi\mu i^2$ т. е. дуже старечий вік визначиться так.

Для висновку на нечітких множинах використаються спеціальні відносини й операції над ними.

Одним з перших застосувань теорії НМ

{60*0,6+70*0,8+80*0,9+90*1 }

{60*0,36+70*0,64+80*0,81+90*1}

стало використання коефіцієнтів упевненості для висновку рекомендацій медичної системи MYC1N. Цей метод використає кілька евристичних прийомів. Він став прикладом обробки нечітких знань, що вплинули на наступні системи.

У цей час у більшість інструментальних засобів розробки систем, заснованих на знаннях, включені елементи роботи із НМ, крім того, розроблені спеціальні програмні засоби реалізації так називаного нечіткого висновку, наприклад "оболонка" FuzzyCLIPS.

Питання для самоконтролю:

1. Які два типи операцій допустимі для нечітких множин?

2. Дайте визначення алгебраїчного твору, алгебраїчної суми, диз'юнктивнійсуми.

3. Які логічні операції для нечітких множин Вам відомі? Вкажіть їх особливості.

Завдання для самостійного виконання:

Дано множини:

E = {2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000, 10000},

A – нечітка множина «висока зарплата»,

B — нечітка множина «низька зарплата»,

A = 0,1 / 3000 + 0,2 / 4000 + 0,4 / 6000 + 0,5 / 7000 + 0,7 / 9000 + 0,8 / 10000,

B = 0,8 / 2000 + 0,7 / 3000 + 0,6 / 4000 + 0,5 / 5000 + 0,4 / 6000 + 0,2 / 8000 + 0,1 /10000.

Знайдіть $A \vee B, A \wedge B, \neg A.$

Форма звіту: представити рішення у вигляді графіків та надати їх інтерпретацію.

1.3.3 Лабораторна робота №10. Нечітке логічне виведення

МЕТА: навчити здійснювати нечітке логічне виведення**.**

ЗАДАЧІ:

1. ознайомити з поняттям «нечіткий логічний висновок»;
2. ознайомити з етапами нечіткого логічного висновку, алгоритмами нечіткого логічного виведення та їх особливостями;
3. розглянути приклади завдань в яких використовується механізм нечіткого логічного висновку.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Поняття нечіткого виводу займає центральне місце в нечіткої логіки і в теорії нечіткого управління. Говорячи про нечіткої логіки в системах управління, можна дати наступне визначення системи нечіткого виводу.

Система нечіткого виводу — це процес отримання нечітких висновків про необхідному управлінні об'єктом на основі нечітких умов або передумов, що представляють собою інформацію про поточний стан об'єкта.

Цей процес поєднує в собі всі основні концепції теорії нечітких множин: функції приналежності, лінгвістичні змінні, методи нечіткої імплікації і т.п. Розробка і застосування систем нечіткого виводу включає в себе ряд етапів, реалізація яких виконується на основі розглянутих раніше положень нечіткої логіки.

Формування бази правил систем нечіткого виводу:

Найбільш часто база нечітких продукційних правил представляється у формі узгодженого щодо використовуваних лінгвістичних змінних структурованого тексту:

ПРАВІЛО_1: ЯКЩО «Условіе_1» ТО «Заключеніе_1» (F 1 т),

...

ПРАВІЛО_n: ЯКЩО «Условіе_n» ТО «Заключеніе_n» (F n),

де F і ∈ [0; 1] є коефіцієнтом визначеності або ваговим коефіцієнтом відповідного правила. Узгодженість списку означає, що в якості умов і висновків правил можуть використовуватися тільки прості і складові нечіткі висловлювання, з'єднані бінарними операціями «І», «АБО», при цьому в кожному з нечітких висловлювань мають бути визначені функції приналежності значень терм-множин для кожної лінгвістичної змінної. Як правило, функції приналежності окремих термів представляють трикутними або трапецеїдальними функціями. Для найменування окремих термів прийнято використовувати скорочення (табл. 1.3).

Таблиця 1.3 - Загальноприйняті скорочення для значень основних термів лінгвістичних змінних в системах нечіткого виведення

Символическое обозначение	Англоязычная нотация	Русскоязычная нотация
NB	Negative Big	Отрицательное большое
NM	Negative Middle	Отрицательное среднее
NS	Negative Small	Отрицательное малое
ZN	Zero Negative	Отрицательное близкое к нулю
Z	Zero	Нуль, близкое к нулю
ZP	Zero Positive	Положительное близкое к нулю
PS	Positive Small	Положительное малое
PM	Positive Middle	Положительное среднее
PB	Positive Big	Положительное большое

Приклад. Мається наливна ємність (бак) з безперервним керованим припливом рідини і безперервним некерованим витратою рідини. База правил системи нечіткого виведення, відповідна знань експерта про те, який необхідно вибрати приплив рідини щоб рівень рідини в баку залишався середнім, буде виглядати наступним чином:

ПРАВИЛО <1>: ЯКЩО «рівень рідини малий» І «витрата рідини великі» ТО «приплив рідини Великий»

ПРАВИЛО <2>: ЯКЩО «рівень рідини малий» І «витрата рідини середня» ТО

«приплив рідини Великий»

ПРАВИЛО <3>: ЯКЩО «рівень рідини малий» І «витрата рідини мала»ТО «приплив рідини Великий»

ПРАВИЛО <4>: ЯКЩО «рівень рідини середній» І «витрата рідини велика» ТО

«приплив рідини Великий»

ПРАВИЛО <5>: ЯКЩО «рівень рідини середній» І «витрата рідини середня»ТО «приплив рідини Середній»

ПРАВИЛО <6>: ЯКЩО «рівень рідини середній» І «витрата рідини мала» ТО «приплив рідини Середній»

ПРАВИЛО <7>: ЯКЩО «рівень рідини великий» І «витрата рідини велика» ТО

«приплив рідини Середній»

ПРАВИЛО <8>: ЯКЩО «рівень рідини великий» І «витрата рідини середня»ТО «приплив рідини Малий»

ПРАВИЛО <9>: ЯКЩО «рівень рідини великий» І «витрата рідини мала» ТО «приплив рідини Малий»

Отримані правила представимо у вигляді таблиці (табл. 1.4):

Таблиця 1.4 - Подання правил у вигляді скорочень

Расход \ Уровень	ZP	PM	PB
ZP	ZP	ZP	ZP
PM	PM	PM	ZP
PB	PB	PB	PM

Фазифікація. Фазифікація (введення нечіткості) — це установка відповідності між чисельним значенням вхідної змінної системи нечіткого виводу і значенням функції приналежності відповідного їй терма лінгвістичної змінної. На етапі фазифікації значенням всіх вхідним змінним системи нечіткого виведення, отриманим зовнішнім по відношенню до системи нечіткого виводу способом, наприклад, за допомогою датчиків, ставляться у відповідність конкретні значення функцій приналежності відповідних лінгвістичних термів, які використовуються в умовах (антецедент) ядер нечітких продукційних правил, складових базу нечітких продукційних правил системи нечіткого виводу. Фазифікація вважається виконаною, якщо знайдені ступеня істинності μ A (x) всіх елементарних логічних висловлювань виду «β ЕСТЬ ά», що входять до антецеденти нечітких продукційних правил, де ά — деякий терм з відомою функцією приналежності μ A (x), а — чітке чисельне значення, що належить універсуму лінгвістичної змінної β.

Приклад. Для ілюстрації виконання цього етапу розглянемо приклад процесу фазифікації трьох нечітких висловлювань: "швидкість автомобіля мала", "швидкість автомобіля середня", "швидкість автомобіля висока" для вхідних лінгвістичної змінної β1 — швидкість руху автомобіля. Їм відповідають нечіткі висловлювання першого виду: "β1 є α1", "β1 є α2", "β1 є α3". Припустимо, що поточна швидкість автомобіля дорівнює 55 км/год, тобто α1=55 км/год. Тоді фазифікація першого нечіткого висловлювання дає в результаті число 0, яке означає його ступінь істинності і виходить підстановкою значення a1=55 км/год в якості аргументу функції приналежності терма α1 (рис. 1.33 (а)). Фазифікація другого нечіткого висловлювання дає в результаті число 0.67 (наближене значення), яке означає його ступінь істинності і виходить підстановкою значення a1=55 км/год в якості аргументу функції приналежності терма α2 (рис. 1.33 (б)). Фазифікація третього нечіткого висловлювання дає в результаті число 0, яке означає його ступінь істинності і виходить підстановкою значення a1 = 55 км/год в якості аргументу функції приналежності терма α3. Результат фазифікації представлений на рис. 1.33.

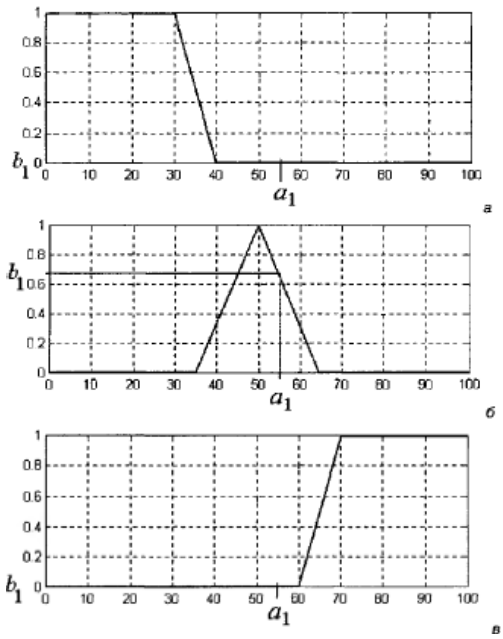

Рисунок 1.33 - Приклад фазифікації вхідної лінгвістичної змінної «швидкість автомобіля» для трьох нечітких висловлювань

Агрегування (Aggregation)

Агрегування являє собою процедуру визначення ступеня істинності умов по кожному з правил системи нечіткого виводу. Формально процедура агрегування виконується наступним чином. До початку цього етапу передбачаються відомими значення істинності всіх підумов системи нечіткого виведення, т. Е. Безліч значень B = {bi}. Далі розглядається кожне з умов правил системи нечіткого виводу. Якщо умова правила являє собою нечітке висловлювання виду 1 або 2, то ступінь його істинності дорівнює відповідному значенню bi .

Етап агрегування вважається закінченим, коли будуть знайдені всі значення bi "для кожного з правил Rk, що входять у розглянуту базу правил P системи нечіткого виводу. Цю множину значень позначимо через B "= {b1 ", b2 ", ..., bn "}

Приклад. Для ілюстрації виконання цього етапу розглянемо приклад процесу агрегування двох нечітких висловлювань: "швидкість автомобіля середня" I"кава гаряча" і "швидкість автомобіля середня" АБО "кава гаряча" для вхідних лінгвістичної змінної β1 — швидкість руху автомобіля і β2 — температура кави. Припустимо, що поточна швидкість автомобіля дорівнює 55 км/год, тобто A1 = 55 км/год, а температура кави дорівнює a2 = 70 ° C. Тоді агрегування першого нечіткого висловлювання з використанням операції нечіткої кон'юнкції дає в результаті число b1" = 0.67 (наближене значення), яке означає його ступінь істинності і виходить як мінімальне зі значень 0.67 і 0.8 (мал.3а). Агрегування другого нечіткого висловлювання з використанням операції нечіткої диз'юнкції дає в результаті число b2" = 0.8, яке означає його ступінь істинності і виходить як максимальне з значень 0.67 і 0.8 (рис.1.34).

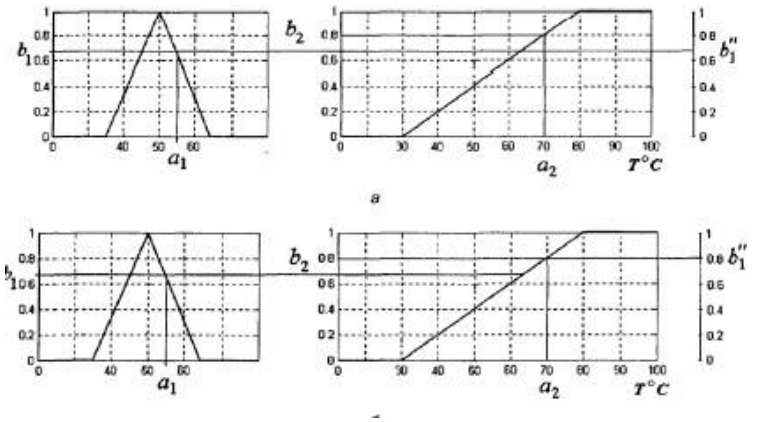

Рисунок 1.34 - Параметри агрегування підумови для двох нечітких висловлювань «швидкість автомобіля середня» І «температура кави висока»(а) та «швидкість автомобіля середня» АБО «температура кави висока»(б)»

Активізація (Activation)

Активізація в системах нечіткого виведення являє собою процедуру або процес знаходження ступеня істинності кожного з підвисновків правил нечітких продукцій. Активізація в загальному випадку багато в чому аналогічна композиції нечітких відносин, але не тотожна їй. Оскільки в системах нечіткого виводу використовуються лінгвістичні змінні, то формули для нечіткої композиції втрачають своє значення. Насправді при формуванні бази правил системи нечіткого виводу задаються вагові коефіцієнти F_i для кожного правила (за замовчанням передбачається, якщо ваговий коефіцієнт не заданий явно, то його значення дорівнює 1).

Формально процедура активізації виконується наступним чином. До початку цього етапу передбачаються відомими значення істинності всіх умов системи нечіткого виведення, тобто множина значень B "= {b_i"} і значення вагових коефіцієнтів F_i для кожного правила. Далі розглядається кожне з висновків правил системи нечіткого виведення. Якщо висновок правила являє собою нечітке висловлювання виду 1 або 2, то ступінь його істинності дорівнює алгебраїчному добутку відповідного значення b_1" на ваговий коефіцієнт F_i. Якщо ж висновок складається з декількох підвисновків,

причому лінгвістичнізмінні в підвисновках попарно не рівні один одному, то ступінь істинності кожного з підвисновків дорівнює алгебраічному добутку відповідного значенняb_i" на ваговий коефіцієнт F_i. Таким чином, знаходяться всі значення $c_к$ ступенів істинності підвисновків для кожного з правил R_k, що входять у розглянуту базу правил Р системи нечіткого виводу. Це безліч значень позначимо через $C=\{c_1, c_2, ..., c_q\}$, де q — загальна кількість підвисноків в базі правил.

Етап активізації вважається закінченим, коли для кожної з вихідних лінгвістичних змінних, що входять в окремі підвисновкі правил нечітких продукцій, будуть визначені функції приналежності нечітких множин їх значень, тобто сукупність нечітких множин: $C_1, C_2, ..., C_q$, де q — загальна кількість підвисновків в базі правил системи нечіткого виводу.

Приклад. Для ілюстрації виконання цього етапу розглянемо приклад процесу активізації ув'язнення в наступному правилі нечіткої продукції (це правило, навряд чи має цільове застосування і використовується формальним чином): ЯКЩО "швидкість автомобіля середня" ТО "кава гаряча". Вхідною лінгвістичною змінною в цьому правилі є β_1 — швидкість руху автомобіля, а вихідний змінної є β_2 — температура кави. Припустимо, що поточна швидкість автомобіля дорівнює 55 км/год, тобто a_1 = 55 км/год.

Оскільки агрегування умови цього правила дає в результаті b_1"=0.67, а ваговий коефіцієнт дорівнює 1 (за замовчуванням), то значення 0.67 буде використовуватися в якості c1 для отримання результату активізації.

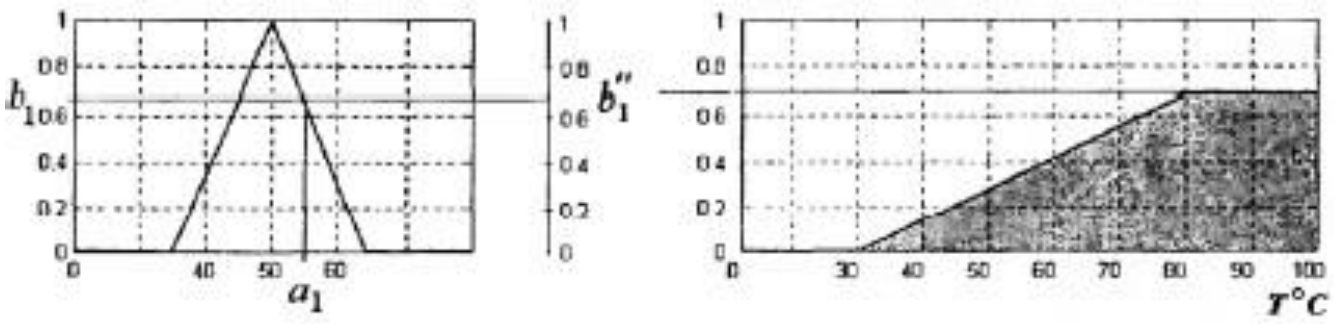

Рисунок 1.35 - Приклад активізації висновків для правил нечіткої продукції

Акумуляція. Акумуляція або акумулювання в системах нечіткого

виведення являє собою процедуру або процес знаходження функції приналежності для кожної з вихідних лінгвістичних змінних безлічі $W=\{w_1, w_2, ..., w_s\}$. Мета акумуляції полягає в тому, щоб об'єднати або акумулювати всі ступені істинності висновків (підвисновків) для отримання функції приналежності кожної з вихідних змінних. Причина необхідності виконання цього етапу полягає в тому,що підвисновки, що відносяться до однієї і тієї ж вихідної лінгвістичної змінної, належать різним правилам системи нечіткого виводу.

Формально процедура акумуляції виконується наступним чином. До початку цього етапу передбачаються відомими значення істинності всіх підвисновківдля кожного з правил R_k, що входять у розглянуту базу правил P системи нечіткого виведення, у формі сукупності нечітких множин: $C_1, C_2, ..., C_q$, де q— загальна кількість підвисновків в базі правил. Далі послідовно розглядається кожна з вихідних лінгвістичних змінних $w_j \in W$ і співвідношені до неї нечіткі множини: $C_{j1}, C_{j2}, ..., C_{jq}$. Результат акумуляції для вихідної лінгвістичної змінної w_j визначається як об'єднання нечітких множин $C_{j1}, C_{j2}, ..., C_{jq}$ за однією з формул. Етап акумуляції вважається закінченим, коли для кожної з вихідних лінгвістичних змінних будуть визначені підсумкові функції приналежності нечітких множин їх значень, тобто сукупність нечітких множин: $C_1', C_2', ..., C_s'$ де s — загальна кількість вихідних лнгвістичних змінних в базі правил системинечіткого виводу

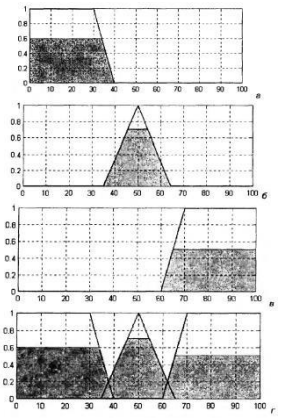

Рисунок 1.35 - Приклад акумуляції висновків для вихідної лінгвістичної змінної« швидкість рухуавтомобіля

Дефазіфікація. Дефазіфікації в системах нечіткого виведення являє собою процедуру або процес знаходження звичайного (не нечіткого) значення для кожної з вихідних лінгвістичних змінних безлічі $W = \{w_1, w_2, ..., w_s\}$.

Мета дефазіфікації полягає в тому, щоб, використовуючи результати акумуляції всіх вихідних лінгвістичних змінних, отримати звичайне кількісне значення (crisp value) кожної з вихідних змінних, яке може бути використане спеціальними пристроями, зовнішніми по відношенню до системи нечіткого виводу.

Дійсно, застосовувані в сучасних системах управління пристрої та механізми здатні сприймати традиційні команди у формі кількісних значень відповідних керуючих змінних. Саме з цієї причини необхідно перетворити нечіткі множини в деякі конкретні значення змінних. Тому дефазіфікації називають також приведенням до чіткості. Формально процедура дефазіфікації виконується наступним чином. До початку цього етапу передбачаються відомими функції приналежності всіх вихідних лінгвістичних змінних у формі нечітких множин:

$$C_1', C_2', ..., C_s',$$

де s - загальна кількість вихідних лінгвістичних змінних в базі правил системи нечіткого виводу. Далі послідовно розглядається кожна з вихідних лінгвістичних змінних $w_j \in W$ відноситься до неї нечітка множина C_j'. Результат дефазіфікації для вихідної лінгвістичної змінної w_j, визначається у вигляді кількісного значення $y_i \in R$, одержуваного за однією з розглянутих нижче формул. Етап дефазіфікації вважається закінченим, коли для кожної з вихідних лінгвістичних змінних будуть визначені підсумкові кількісні значення у формі деякого дійсного числа, тобто у вигляді $y_1, y_2, ..., y_s$, де s — загальна кількість вихідних лінгвістичних змінних в базі правил системи нечіткого виводу.

Методи дефазіфікації представлені в табл. 1.5.

Таблиця 1.5 - Методи дефазіфікації

Назва методу	Формула
Метод центру ваги	$y = \frac{\int_{Min}^{Max} x * \mu(x)dx}{\int_{Min}^{Max} \mu(x)dx}$
Метод центру ваги для одноточкових множин	$y = \frac{\sum_{i=1}^{n} x_i * \mu(x_i)}{\sum_{i=1}^{n} \mu(x_i)}$
Метод центру площі	$\int_{Min}^{u} \mu(x)dx = \int_{u}^{Max} \mu(x)dx$
Метод лівого модального значення	y = *min{xm}*
Метод правого модального значення	y = *max{xm}*

Питання для самоконтролю:

1. Що називають «нечітким логічним виведенням»?
2. Які етапи нечіткого логічного виведення Вам відомі?
3. Які алгоритми нечіткого логічного виведення Вам відомі?

Завдання для самостійного виконання:

Сформувати нечітку модель управління змішувачем води при прийнятті душа, використовуючи алгоритм Мамдамі. (Вважати температуру води при останньому вимірі такою, що дорівнює 55 градусів).

Змістовна постановка задачі.

При прийнятті душа на вхід змішувача подається холодна і гаряча вода за відповідними магістральних трубопроводах. Найбільш комфортні умови для душа створюються при наявності на виході змішувача теплої води постійної температури. Оскільки під час прийняття душу може спостерігатися

нерівномірний витрата води, температура води на виході змішувача буде коливатися, приводячи до необхідності ручного зміни подачі холодної або гарячої води. Завдання полягає в тому, щоб зробити регулювання температури води автоматичною, забезпечуючи постійну температуру води на виході змішувача.

Досвід прийняття душу дозволяє сформулювати кілька евристичних правил, якими застосовуємо в разі регулювання температури води на виході змішувача:

1. ЯКЩО вода гаряча, ТО слід повернути вентиль крана гарячої води на великий кут вправо.

2. ЯКЩО вода не дуже гаряча, ТО слід повернути вентиль крана гарячої водина невеликий кут вправо.

3. ЯКЩО вода тепла, ТО залишити вентиль крана гарячої води без впливу.

4. ЯКЩО вода прохолодна, ТО слід повернути вентиль крана гарячої води наневеликий кут вліво.

5. ЯКЩО вода холодна, ТО слід повернути вентиль крана гарячої води на великий кут вліво.

Ця інформація буде використовуватися при побудові бази правил системи нечіткого виведення, яка дозволяє реалізувати дану модель нечіткого управління.

Форма звіту: формування моделі управління поведінкою змішувача при температурі 55 градусів.

Теми для самостійного опрацювання

1. Поняття штучного інтелекту. Опрацювання літератури з існуючого стану за основними напрямами штучного інтелекту. Добір прикладів застосування інтелектуальних систем.

2. Основні засоби управління логічним виведенням. Реалізація процедур логічного виведення для отримання результатів консультування за основними стратегіями. Трасування виконання програми.

3. Нечітке логічне виведення. Складання функцій належності та їх аналіз. Реалізація процедур нечіткого логічного виведення.

Розділ 2

РЕАЛІЗАЦІЙНІ ОСНОВИ СТВОРЕННЯ ТА ВИКОРИСТАННЯ ЕКСПЕРТНИХ СИСТЕМ

2.1 Архітектура та особливості експертних систем

Класифікація експертних систем

Інтелектуальними називають системи, основним ядром яких є база знань або модель предметної області, що описана мовою високого рівня наближеною до природної. Вона називається м**овою представлення знань (МПЗ).**

Переважно інтелектуальні системи застосовують для вирішення складних задач, що пов’язані із застосуванням знань спеціалістів – експертів і де логічна обробка інформації переважає над обчислювальною. Наприклад:

- прийняття рішень в складних ситуаціях;
- розуміння природної мови;
- визначення діагнозу та методу лікування;
- аналіз візуальної інформації та інше.

Прикладні інтелектуальні системи використовують в численних застосуваннях. Щорічний дохід від продажів програмних та апаратних засобів ШІ вже перевищує 1 млрд. дол..

Поширеними інтелектуальними системами є експертні системи.

Експертна система – це клас інтелектуальних систем, що орієнтований на застосуванні досвіду висококваліфікованих спеціалістів в галузях, де якість прийняття рішень традиційно залежить від рівня експертизи (медицина, хімія, фармакологія, економіка, геологія, юриспруденція). Висновки та рекомендації ЕС є прозорими, та їх можна інтерпретувати на високому рівні.

ЕС спроможні вирішувати неформалізовані задачі, яким притаманні особливості:

- помилковість, неоднозначність, неповнота і суперечність початкових даних, знань про ПО і задачу, що вирішується;

- велика розмірність простору пошуку;
- дані і знання динамічно змінюються.

ЕС різняться від традиційних систем обробки даних використанням символьного способу представлення даних, логічним виведенням і евристичним пошуком.

База даних (робоча пам'ять) призначена для збереження початкових і проміжних даних задачі

База знань – сукупність знань про предметну область (ПО), що записана у вигляді правил мовою, яка є зрозумілою для користувача та експерту.

Машина виведення (інтерпретатор правил) – це програма, що моделює процес роздумів експерта на основі знань у БЗ.

Компонента придбання знань автоматизує процес наповнення ЕС знаннями, що отримані під час діалогу з експертом або інженером знань

Пояснювальна компонента відслідковує процес отримання висновку. Виводить довідкову інформацію про задіяні фрагменти БЗ. Полегшує тестування системи і підвищує довіру до отримання висновку.

Діалогова компонента організовує спілкування з користувачем як під час отримання висновку, так і під час придбання знань (рис.2.1)

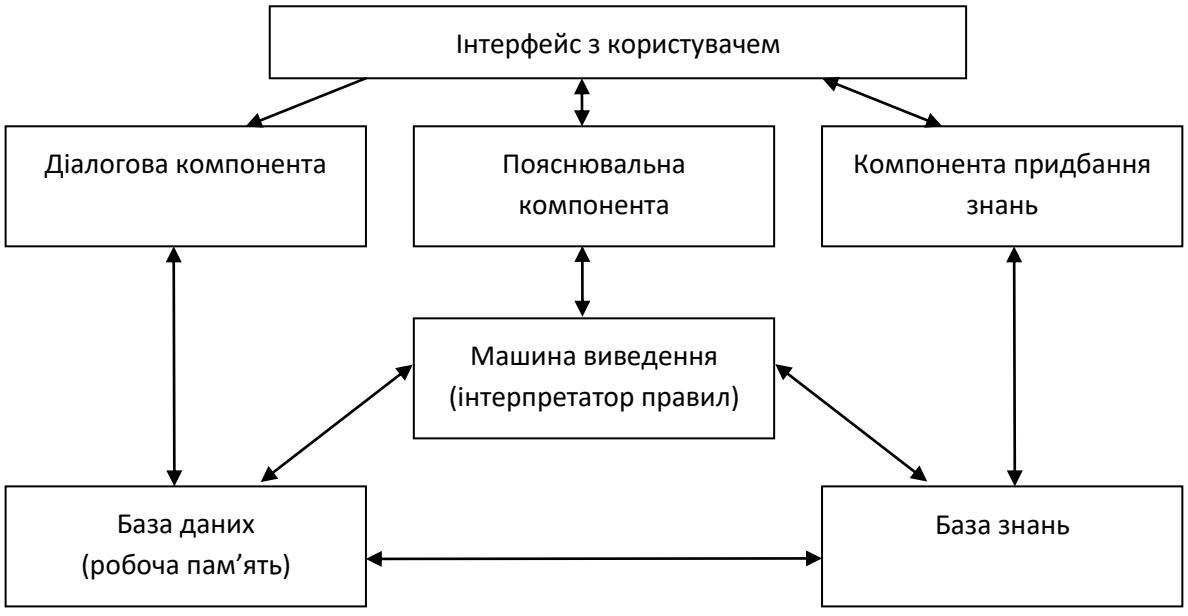

Рисунок 2.1 - Структура експертної системи

Для створення ЕС потрібні:

- **Експерт з ПО.** Визначає знання (дані і правила), що є характерними для ПО, забезпечує повноту і достовірність введення до ЕС знань.
- **Інженер по знаннях.** Допомагає експерту виявити і структурувати знання, що є необхідними для ЕС і визначає спосіб представлення знань
- **Програміст.** Розробляє інструментальні засоби, що містять основні компоненти ЕС, створює зручний інтерфейс користувача.

На відміну від традиційних підходів, розробку ЕС здійснює не програміст, а інженер по знаннях, що не вміє програмувати.

ЕС працює в 2 режимах:

- режим придбання знань;
- режим використання знань.

Режим придбання знань

Експерт описує ПО у вигляді сукупності даних і правил.

Дані – це об’єкти та їх значення

Правила визначають способи маніпулювання даними

Інженер по знаннях за допомогою компоненти придбання знань наповнює ЕС знаннями, що надає можливості вирішувати задачі

Режим використання

Здійснює користувач, якого цікавить висновок і спосіб його отримання. Залежно від призначення ЕС, користувач може бути нефахівцем в ПО або фахівцем (для пришвидшення рутинної роботи).

У режимі використання дані за допомогою діалогової компоненти надходять до БД (робочої пам’яті). На підставі вхідних даних з робочої пам’яті, загальних даних про ПО і правил з БЗ, машина виведення формує рішення задачі. Якщо висновок є незрозумілим, тоді користувач може отримати довідкову інформацію про кроки логічного виведення.

За зв'язком з реальним часом ЕС бувають:

ЕС статичного типу. Використовують там, де можна не враховувати зміни навколишнього середовища під час вирішення задачі. Вміст БД та БЗ з часом не змінюється.

ЕС динамічного типу. Працюють в режимі реального часу з неперервною обробкою даних, що надходять (рис.2.2).

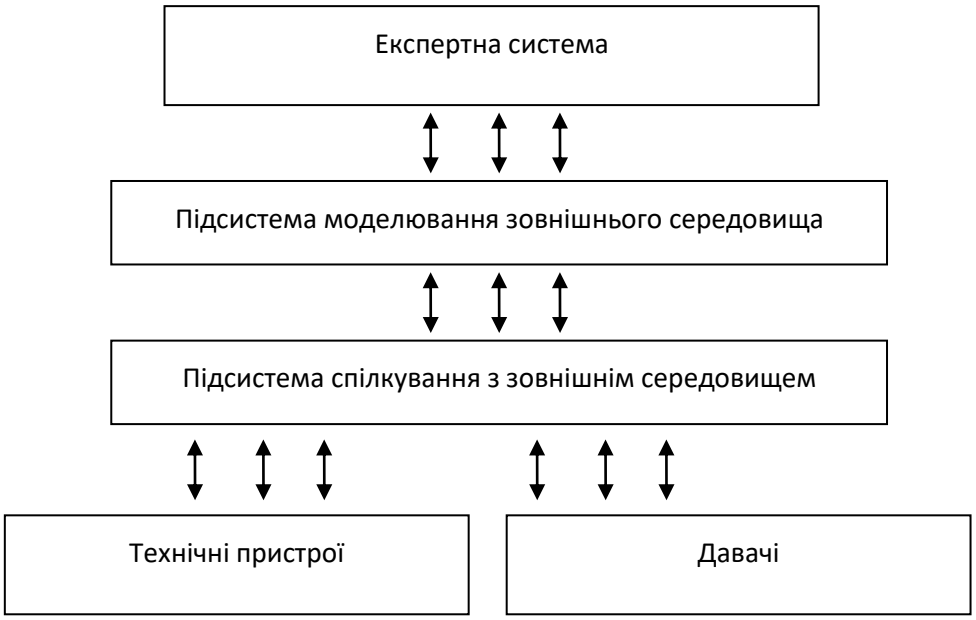

Рисунок 2.2 – Архітектура динамічної ЕС

До складу архітектури динамічної ЕС входять 2 компоненти:

- підсистема моделювання зовнішнього середовища;
- підсистема спілкування з зовнішнім середовищем.

БД та БЗ зазнають істотних змін для врахування всіх подій реального середовища.

ЕС доцільно створювати за наступних умов:

- вирішення задачі потребує логічного виведення;
- задача відноситься до «зрозумілої» і структурованої ПО;
- існує група досвідчених експертів ПО, які можуть пояснити природною мовою методи вирішення задачі, які вони використовують в своїй професійній діяльності;
- експерти погоджуються між собою в оцінці отриманого рішення.

ЕС доцільно використовувати, коли:

- виіршення задачі принесе значний ефект;
- використання людини-експерта є неможливим (наприклад, космічні станції);
- при залученні експерта відбувається неприпустима втрата часу або інформації;
- вирішення задачі в умовах, шкідливих для людини (наприклад, небезпечне виробництво).

Під час створення ЕС існує певна технологія розробки, що містить наступні етапи:

- Ідентифікація проблеми
- Концептуалізація (структуризація) знань
- Формалізація знань
- Створення прототипу системи
- Дослідна експлуатація

Ідентифікація проблеми

Осмислення задач, що будуть розв’язуватися майбутньою ЕС та формування до неї вимог, зокрема:

1. Визначення ПО та ідентифікація задачі
2. Визначення учасників проекту та їх ролі
3. Визначення підходу до рішення задачі
4. Визначення ресурсів і цілей, складання плану розробок

1. Визначення ПО та ідентифікація задачі

Вибір ПО, що є важливою частиною розробки, від якої залежить ефективність ЕС

Для ідентифікації проблеми визначають:

- Джерела знань (книги, методики)
- Аналогічні ЕС
- Класи вирішуваних задач

Ідентифікація задачі містить неформальний опис, де вказуються:

- Загальні характеристики задачі
- Підзадачі всередині даної задачі
- Ключові поняття (об’єкти)
- Бажаний вигляд рішення
- Існуючі знання

2. Визначення учасників проекту та їх ролі

Мінімально колектив розробник має такий склад:

- 1 експерт
- 1-2 інженери по знаннях
- 1 програміст

Але, у міру необхідності можуть залучатися і більше учасників, наприклад, експерти, щоб не було протиріччя знань.

3. Визначення підходу до рішення задачі

Інженер та експерт працюють у тісному контакті. Початковий неформальний опис задачі експертом використовується інженером по знання для вибору термінів і ключових понять.

Експерт пояснює опис задачі, шляхи її вирішення та процес логічного виведення рішення. Відбувається передача знань інженеру по знаннях з використанням діалогу, лекцій, дискусій, спостережень. Зрештою, експерт і інженер по знаннях отримують остаточний неформальний опис задачі.

Для експерта джерелом знань є попередній досвід по рішенню задачі, книги, відомі приклади. Для інженера по знаннях це досвід по рішенню аналогічних задач, методи представлення знань і маніпулювання ними.

4. Визначення ресурсів і цілей складання плану розробок

Термін розробки і впровадження ЕС, як правило складає не менше року.

На процес розробки суттєво впливає обсяг фінансування. Якщо воно недостанє, тоді краще адаптувати існуючу систему, а не створювати нову.

Концептуалізація (структурування) знань

Концептуалізація – це розробка неформального опису знань про ПО у вигляді графів, таблиць, діаграм або тексту, що відбивають основні концепції та взаємозв’язки між поняттями ПО.

Проводиться змістовний аналіз ПО, будується структура отриманих знань і визначається:

- Термінологія
- Список основних понять та їх атрибутів
- Відношення між поняттями
- Стратегії пошуку
- Обмеження стратегій

Формалізація знань

Створюється формалізоване представлення концепцій ПО мовою представлення знань:

- Продукційні моделі
- Семантичні сітки
- Фреймові моделі
- Логічні моделі
- Об’єктно-орієнтовані мови

Створення прототипу

Створення одного або кількох прототипів за допомогою:

- Програмування традиційними мовами (C++, Паскаль)
- Програмування спеціалізованими мовами (LISP, FRL, SmallTalk)
- Використання інструментальних засобів розробки ЕС (G2)
- Використання пустих експертних оболонок

Прототип містить всі блоки ЕС і потрібен для перевірки правильності кодування даних, правил і стратегій пошуку. Процес вдосконалення прототипу є ітеративним. Він продовжується від кількох місяців до кількох років в залежності від складності ПО.

При розробці прототипу враховують:

- Функціонування системи при значному розширенні бази знань
- Можливості системи для рішення ширшого кола задач
- Зауваження користувачів про функціонування ЕС
- Розробка системи вводу/виводу, що здійснює аналіз або синтез природної мови

Під час розробки прототипу відбувається модифікація системи: пере формулювання понять, переконструювання представлення знань, вдосконалення прототипу.

Тестування

Інженер по знаннях підбирає приклади, щоб перевірити всі можливості розробленої ЕС:

- Зручність інтерфейсу вводу-виводу
- Ефективність стратегії пошуку
- Якість висновку
- Коректність БЗ

Від тестових прикладів залежить подальша робота ЕС. Приклади мають бути неоднорідними і охоплювати всю ПО. При підготовці тестових прикладів, їх варто класифікувати по під проблемах ПО, виділяти стандартні випадки, визначати межі складних ситуацій

Зручність інтерфейсу вводу-виводу

Вводом-виводом вважають дані, що набуваються під час діалогу з інженером по знаннях та висновки, що видаються системою.

Вхідні питання системи можуть бути незручними для користувача з наступних причин:

- Неправильні питання
- Питання, що є важкими для розуміння, багатозначними і не відповідними до знань користувача
- Незручна для користувача мова

Для вихідних повідомлень це:

- Занадто мало або багато слів
- Невдала організація та впорядкованість висновків
- Незрозумілий рівень абстракцій з незрозумілою лексикою

Джерелом похибок у правилах виведення може бути:

- Неврахування взаємозв'язку сформованих правил
- Суперечливість і неповнота правил

До помилок в роботі ЕС приводять і застосовані стратегії керування. Наприклад, вибрана стратегія аналізує факти у порядку, що різниться від «природного» для експерта. Це може впливати не лише на ефективність системи, а й привести до іншого кінцевого результату.

Зміна стратегії потрібна і у разі неефективної роботи ЕС під впливом надмірно складних виводів і пояснень.

Оцінювання ЕС відбувається за різними критеріями:

- **Колективу розробників.** Еективність реалізації, не протиріччя БЗ, повнота охоплення ПО, толерантність програми до змін в представленні знань
- **Експертів.** Оцінювання висновків та пояснень
- **Користувачів.** Зручність інтерфейсу, зрозумілість роботи ЕС.

Дослідна експлуатація

Перевіряється придатність ЕС для кінцевого користувача, зокрема, користь і зручність роботи.

Під **користю** розуміють спроможність під час діалогу визначати потреби користувача, виявляти і усувати причини невдач у роботі, вирішувати поставлені задачі.

Зручність – це природність взаємодії з ЕС, гнучкість відповідно до різної кваліфікації користувачів, толерантність системи до помилок, спроможність не виходити з ладу при помилкових діях користувача.

Відбувається втілення ЕС в робоче середовище і навчання персоналу по експлуатації та обслуговуванню. Забезпечується зв'язок ЕС з існуючими БД та іншими системами.

Класифікація експертних систем (як системи, що засновані на знаннях)

Клас ЕС об'єднує кілька тисяч різних програмних комплексів, що можна класифікувати за різними критеріями:

- За типом вирішуваної задачі
- За зв'язками з реальним часом
- За типом використаної ЕОМ
- За ступенем інтеграції з іншими програмами

Класифікація за типом вирішуваної задачі

Задачі, що вирішуються на ЕС можна поділити:

- **Задачі аналізу**, де множина рішень обмежена і втілена у систему.
- **Задачі синтезу**, де множина рішень не обмежена і виводиться з рішень під задач чи компонентів.

Задачі аналізу

Інтерпретація даних – процес визначення змісту даних для отримання конкретних результатів. Передбачає багатоваріантний аналіз даних..

Наприклад, ЕС АВТАНТЕСТ – визначення основних властивостей особистості по результатах психодіагностичного тестування

Діагностика – процес класифікації об'єктів та пошук несправностей або відхилень від норми в системі.

Наприклад, ANGY - діагностика і терапія звуження коронарних судин. GRIB – діагностика помилок в апаратурі і математичному забезпеченні ЕОМ.

Моніторинг – неперервна інтерпретація даних у реальному часі. Якщо трапляється відхилення певних параметрів за межі норми, тоді спрацьовує повідомлення.

Наприклад, REACTOR – контроль за роботою електростанцій, FALCON – контроль аварійних здавачів на хімічному заводі.

Підтримка прийняття рішень – сукупність процедур, що виробляє необхідні рекомендації для прийняття рішення. Допомога фахівцю для вибору альтернативи серед множини виборів.

Наприклад, CRYSIS – вибір стратегії виходу фірми з кризового стану, CHOICE – допомога у виборі страхової компанії або інвестора.

Задачі синтезу

Проектування – підготовка необхідної документації для створення об’єктів з визначеними властивостями. В задачах проектування відбувається два процеси: процес виведення рішення і процес роз’яснення.

Наприклад, CADHELP – проектування інтегрованих інтегрованих схем.

Планування – знаходження плану об’єктів, що виконують певну функцію. ЕС використовують моделі поведінки реальних об’єктів, щоб логічного вивести наслідки планованої діяльності.

Наприклад, STRIPS – планування поведінки робота, ISIS – планування промислових замовлень, MOLGEN – планування експерименту.

Керування – функція організованої системи, що підтримує певний режим діяльності. ЕС керує поведінкою складних систем у відповідності із заданими специфікаціями.

Наприклад, GAS –керування газовою котельною, PROJECT ASSISTANT – керування системою календарного планування.

Комбіновані задачі

Прогнозування – передбачення наслідків деяких подій на основі аналізу існуючих даних. Система використовує параметричну динамічну модель, де значення параметрів підлаштовують під задану ситуацію.

Наприклад, WILLARD – передбачення погоди, PLANT – оцінювання майбутнього врожаю, ECON – економічне прогнозування.

Навчання – використання ЕС для навчання певній дисципліні. Система діагностує помилки і підказують правильні рішення. Відбувається акумуляція знань про рівень учня, щоб планувати наступне спілкування.

Наприклад, УЧИТЕЛЬ ЛИСПа – навчання мові програмування LISP, PROUST – навчання мові Паскаль.

Класифікація за зв'язками з реальним часом

Статичні ЕС – розробляються в предметних областях, де база знань і дані не змінюються у часі і є стабільними. Наприклад, діагностика несправностей в автомобілі.

Квазідинамічні ЕС – інтерпретують ситуацію, яка змінюється в деякому фіксованому інтервалі часу. Наприклад, Мікробіологічні ЕС, де виміри відбуваються раз на 4-5 годин, аналізується динаміка отриманих показників по відношенню до попереднього вимірювання.

Динамічні ЕС – працюють разом з здавачами об'єктів в режимі реального часу з безперервною інтерпретацією даних, що надходять у систему. Наприклад, керування гнучкими виробничими комплексами, моніторинг в реанімаційних палатах.

Класифікація за ступенем інтеграції з іншими програмами

Автономні ЕС працюють в режимі консультації з користувачем для вирішення задач, де не потрібні традиційні методи обробки даних (розрахунки, моделювання).

Гібридні ЕС – програмний комплекс, що містить стандартні пакети прикладних програм (мат статистика, лінійне програмування, СУБД) і засоби маніпулювання значеннями. Це може бути інтелектуальна надбудова над пакетами прикладних програм або інтегроване середовище для вирішення складної задачі з елементами експертних знань.

Розробка таких систем є набагато складнішою, ніж розробка автономної ЕС, оскільки поєднання різних методологій прикладних програм породжує цілий комплекс теоретичних та практичних труднощів.

Основні функції експертних систем

Оскільки теорія експертних систем виросла з більш загальної концепції штучного інтелекту, то немає нічого дивного в тому, що проблематика цих областей має багато спільного. На деяких з таких зв'язків акцентується увага в подальших розділах при огляді літератури. Ви також зустрінете в них заслання на подальші глави цієї книги, в яких та або інша тема розгледить детально.

Придбання знань

Бучанан у такий спосіб сформулював функцію придбання знань [Buchanan et al, 1983]: "[Придбання знань це] передача потенційного досвіду розв'язку проблеми від деякого джерела знань і перетворення його у вигляд, який дозволяє використовувати ці знання в програмі".

Передача знань виконується в процесі досить тривалих і великих співбесід між фахівцем із проектування експертної системи (будемо надалі називати його інженером по знаннях) і експертом у певній предметній області, здатним досить чітко сформулювати наявний у нього досвід. За існуючими оцінками, таким методом можна сформувати від двох до п'яти "елементів знання" (наприклад, правил впливу) у день. Звичайно, це дуже низька швидкість, а тому багато дослідників розглядають функцію придбання знань у якості одного з головних "вузьких місць" технології експертних систем.

Причин такої низької продуктивності предосить. Нижче перелічені тільки деякі з них. Фахівці у вузькій області, як правило, користуються власним жаргоном, який важко перевести на звичайний "людський" мову. Але зміст жаргонного "слівця" аж ніяк не очевидний, а тому потрібно досить багато додаткових питань для уточнення його логічного або математичного значення. Наприклад, фахівці з військової стратегії говорять про "агресивну демонстрацію" іноземної військової моці, але при цьому не можуть пояснити, чому така "агресивна" демонстрація відрізняється від демонстрації, що не несе погрози.

Факти й принципи, що лежать в основі багатьох специфічних галузей знання експерта, не можуть бути чітко сформульовані в термінах математичної теорії або детермінованої моделі, властивості якої добре зрозумілі. Так, експертові у фінансовій галузі може бути відомо, що певні події можуть стати причиною зростання або зниження котирувань на фондовій біржі, але він нічого вам не скаже точно про механізми, які приводять до такого ефекту, або про кількісну оцінку впливу цих факторів. Статистичні моделі можуть допомогти зробити загальний довгочасний прогноз, але, як правило, такі методи не працюють відносно курсів конкретних акцій на коротких часових інтервалах.

Для того щоб розв'язати проблему в певній галузі, експерту недостатньо просто мати суму знань про факти й принципи в цій області.

Наприклад, досвідчений фахівець знає, якого роду інформацією потрібно розташовувати для формулювання того або іншого судження, наскільки надійні різні джерела інформації і як можна розчленувати складну проблему на більш прості, які можна вирішувати більш-менш незалежно. Виявити в процесі співбесіди такого роду знання, засновані на особистому досвіді, що й погано піддаються формалізації, значно складніше, чим отримати простий перелік якихось фактів або загальних принципів.

Експертний аналіз навіть у дуже вузькій галузі, виконуваний людиною, дуже часто потрібно помістити в досить великий контекст, який включає й багато речей, що видадуться експерту саме собою, що розуміють, але для стороннього аж ніяк такими, що не являються. Візьмемо для прикладу експерта-юриста, який бере участь у судовому процесі. Дуже важко креслити кількість і природу знань загального роду, які виявляються залучені в розслідування того або іншої справи.

Подання знань

Подання знань - ще одна функція експертної системи. Теорія подання знань - це окрема область досліджень, тісно пов'язана з філософією

формалізму й когнітивною психологією. Предмет дослідження в цій області - методи асоціативного зберігання інформації, подібні тем, які існують у мозку людини. При цьому основна увага, природно, приділяється логічній, а не біологічній боку процесу, вилучаючи подробиці фізичних перетворень.

Приклад 1.1.

Синтаксис і семантика представлення знань про сімейні відносини Основна частина подання знань, на яку часто навіть не звертають особливої уваги, полягає в тому, що подання повинна якимось способом "стандартизувати" семантичну розмаїття людської мови. Ось кілька речень.

"Степан - батько Бориса".

"Степан - Бориса батько".

"Бориса батько - Степан".

"Батьком Бориса є Степан".

Усі ці фрази виражають ту саму думку (семантично ідентичні). При машинній поданні цієї думки (знання) ми намагаємося знайти більш простий метод зіставлення форми й змісту, чим у звичайній людській мові, тобто добитися того, щоб вирази з однаковим (або схожим) змістом були однаковими й за формою.

Наприклад, усі приведені вище фрази можуть бути зведені до виразу в такій формі:

батько (Степан. Борис)

батько (Степан, Василь)

В 70- х роках дослідження в області подання знань розвивалися в напрямках розкриття принципів роботи пам'яті людину, створення теорій добування відомостей з пам'яті, розпізнавання й відновлення. Деякі з досягнутих у теорії результатів привели до створення комп'ютерних програм, які моделювали різні способи зв'язування понять (концептів). З'явилися комп'ютерні застосування, які могли деяким чином відшукувати потрібні "елементи" знання на певному етапі розв'язку деякої проблеми. Згодом

психологічна вірогідність цих теорій відійшла на другий план, а основне місце, принаймні з погляду проблематики штучного інтелекту, зайняла їхня здатність служити інструментом для роботи з новими інформаційними й керувальними структурами.

Загалом , питання подання знання було й швидше за все залишиться питанням суперечливим. Філософи й психологи найчастіше бувають шоковані безцеремонністю фахівців зі штучного інтелекту, які жваво базікають про людське знання на жаргоні, що уявляє дику суміш термінології, узятої з логіки, логістики, філософії, психології й інформатики. З іншого боку, комп'ютерний формалізм виявився новаторським засобом постановки, а іноді й пошуку відповідей на важкі запитання, над якими сторіччями билися метафізики.

В області експертних систем подання знань цікавить нас, в основному, як засіб відшукання методів формального опису великих масивів корисної інформації з метою їх наступної обробки за допомогою символічних обчислень. Формальний опис означає впорядкування в рамках якої-небудь мови, що володіє достатньо чітко формалізованим синтаксисом побудови виразів і такого ж рівня семантикою, що погоджує значення виразу з його формою.

Символічні обчислення позначають виконання нечислових операцій, у яких можуть бути сконструйовані символи й символьні структури для подання різних концептів і відносин між ними.

В області штучного інтелекту ведеться інтенсивна робота зі створення мов подання (representation languages). Під цим терміном розуміються комп'ютерні мови, орієнтовані на організацію описів об'єктів і ідей, на противагу статичним послідовностям інструкцій або зберіганню простих елементів даних. Основними критеріями доступу до подання знань є логічна адекватність, евристична потужність і природність, органічність нотації. .Ці терміни, швидше за все, потребують пояснень.

Логічна адекватність означає, що подання повинна мати здатність розпізнавати всі відмінності, які ви закладаєте у вихідну сутність. Наприклад, неможливо уявити ідею, що кожні ліки має який-небудь побічний небажаний ефект, якщо тільки не можна буде провести відмінність між призначенням конкретного лікарського препарату і його побічним ефектом (наприклад, аспірин збільшує виразкову хворобу). У більш загальному вигляді вираз, що передає цей ефект, звучить так:

"кожні ліки має небажаний побічний ефект, специфічний для цього препарату".

Евристична потужність означає, що поряд з наявністю виразної мови подання повинна існувати деякий засіб використання вистав, сконструйованих і інтерпретованих таким чином, щоб з їхньою допомогою можна було розв'язати проблему. Часто виявляється, що мова, що володіє більшою виразною здатністю в термінах кількості семантичних відмінностей, виявляється й більше складним у керуванні описом взаємозв'язків у процесі розв'язку проблеми. Здатність до виразу в багатьох зі знайдених формалізмів може виявитися досить обмеженою в порівнянні з англійською мовою або навіть стандартною логікою. Часто рівень евристичної потужності розглядається по результату, тобто по тому, наскільки легко виявляється витягти потрібне знання стосовно до конкретної ситуації. Знати, які знання найбільше підходять для розв'язку конкретної проблеми, - це одне з якостей, яке відрізняє дійсно фахівця, експерта в певній галузі, від новачка або просто начитаний людини.

Природність нотації слід розглядати як якусь чесноту системи, оскільки більшість застосувань, побудованих на базі експертних систем, потребує накопичення великого обсягу знань, а розв'язати таке завдання досить важко, якщо угоди в мові подання занадто складні. Будь-який фахівець скаже вам, що при інших рівних характеристиках краще та система, з якої простіше працювати. Вирази, якими формально описуються знання, повинні бути по

можливості простими для написання, а їх зміст повинен бути зрозумілий навіть тому, хто не знає, як же комп'ютер інтерпретує ці вирази. Прикладом може слугувати декларативний програмний код, який сам по собі дає досить чітка подання про процес його виконання навіть тому, хто не має подання про деталі реалізації комп'ютером окремих інструкцій.

За минулі роки було запропоновано чимало угод, придатних для кодування знань на мовному рівні. Серед них відзначимо правила, що породжують (production rules), структуровані об'єкти (structured objects) і логічні програми (logic programs). У більшості експертних систем використовується один або трохи з перерахованих формалізмів, а доводи на користь і проти кожного з них дотепер являють собою тему для жвавих дискусій серед теоретиків.

Керування процесом пошуку розв'язку

При проектуванні експертної системи серйозна увага повинна бути приділене й тому, як здійснюється доступ до знань і як вони використовуються при пошуку розв'язку. Знання про те, які знання потрібні в тієї або іншій конкретній ситуації, і вміння ними розпорядитися - важлива частина процесу функціювання експертної системи. Такі знання дістали найменування мета-знань - тобто знань про знання. Вирішення нетривіальних проблем вимагає й певного рівня планування й керування при виборі, яке питання потрібно задати, який тест виконати, і т. ін.

Використання різних стратегій перебору наявних знань, як правило, виявляє досить істотний вплив на характеристики ефективності програми. Ці стратегії визначають, яким способом програма відшукує розв'язок проблеми в деякому просторі альтернатив. Як правило, не буває так, щоб дані, якими розташовує програма роботи з базою знань, дозволяли точно "вийти" на ту область у цьому просторі, де має сенс шукати відповідь.

Більшість формалізмів вистави знань може бути використане в різних режимах керування, і розробники експертних систем продовжують

експериментувати в цій області. У наступних главах будуть описані системи, які спеціально підібрані таким чином, щоб проілюструвати відмінності в існуючих підходах до розв'язку проблеми керування. У кожній із представлених систем є що-небудь корисне для студентів, що спеціалізуються в галузі розробки й дослідження експертних систем.

Приклад. 1.2. Обслуговування автомобіля

Уявіть собі, що ваш автомобіль із труднощами заводиться, а в дорозі явно відчувається зниження потужності. Самі по собі ці симптоми недостатні для того, щоб ухвалити рішення, де ж шукати джерело несправності - у паливній або електросистемі автомобіля.

Пізнання в обладнанні автомобіля підказують - потрібно ще поекспериментувати, перш ніж звати на допомогу механіка. Можливо, погана паливна суміш, тому придивитеся до вихлопу й нагару на свічах. Можливо, збоїть розподільник - подивитеся, не ушкоджена чи його кришка. Ці досить специфічні евристики не гарантують, що віднайдеться дійсна причина, але раптом вам посміхнеться фортуна, і ви знайдете несправність без втомлюючої процедури послідовної перевірки всіх систем. Швидше за все, ваших знань досить для того, щоб виконати загальну перевірку, перш ніж займатися доскональним вивченням окремих вузлів.

Наприклад, подивитися, чи досить потужна іскра у свічі (якщо це так, то підозри з електросистеми можна зняти), перш ніж перевіряти акумулятор. При відсутності спеціальних евристик, чому більш методично ви будете діяти, тим більше шансів швидко знайти причину несправності. Загальне евристичне правило говорить:

"Спочатку перевір увесь вузол, а вже потім приступай до перевірки його компонентів".

Це правило можна вважати частиною режиму керування - систематичної стратегії застосування наявних знань. Інше евристичне правило можна сформулювати, наприклад, так:

"Спочатку міняй більш дешеві деталі, а вже потім берися за більш дорогі".

У деяких випадках ці дві евристики можуть суперечити один одному, так що потрібно заздалегідь вибрати, яка з них має пріоритет у випадку, якщо обидві включені в той самий режим керування.

Роз'яснення ухваленого рішення

Питання про те, як допомогти користувачу зрозуміти структуру й функції деякого складного компонента програми, зв'язаний із порівняно новою галуззю взаємодії людину й машини, яка з'явилася на перетинанні таких областей, як штучний інтелект, промислова технологія, фізіологія й ергономіка. На сьогодні внесок у цю галузь дослідників, що займаються експертними системами, полягає в розробці методів подання інформації про поведінку програми в процесі формування ланцюжка логічних висновків при пошуку розв'язку.

Подання інформації про поведінку експертної системи важливо з багатьох причин.

Користувачі, що працюють із системою, потребують підтвердження того, що в кожному конкретному випадку висновок, до якого прийшла програма, в основному коректно.

Інженери, що мають справу з формуванням бази знань, повинні переконатися, що сформульовані ними знання застосовані правильно, у тому числі й у випадку, коли існує прототип.

Експертам у предметній області бажане простежити хід міркувань і спосіб використання тих відомостей, які з їхніх слів були введені в базу знань. Це дозволить судити, наскільки коректно вони застосовуються в даній ситуації.

Програмістам, які супроводжують, налагоджують і модернізують систему, потрібно мати у своєму розпорядженні інструмент, що дозволяє заглянути в "її нутро" на рівні більш високому, чим виклик окремих мовних процедур.

Менеджер системи, що використовує експертну технологію, який зрештою відповідає за наслідки розв'язку, прийнятого програмою, також потребує підтвердження, що ці розв'язки досить обґрунтовані.

Здатність системи пояснити методику ухвалення рішення іноді називають прозорістю системи. Під цим розуміється, наскільки просто персоналу з'ясувати, що робить програма й чому. Цю характеристику системи слід розглядати в сукупності з режимом керування, про який ішла мова в попередньому розділі, оскільки послідовність етапів ухвалення рішення тісно пов'язана із заданою стратегією поведінки.

Відсутність достатньої прозорості поведінки системи не дозволить експертові вплинути на її продуктивність або дати пораду, як можна її підвищити. Простежування й оцінка поведінки системи - завдання досить складне й для її розв'язку необхідні спільні зусилля експерта й фахівця з комп'ютерних наук.

Питання для самоперевірки

1. Чим експертні системи відрізняються від звичайних програмних застосувань і типових програм штучного інтелекту? чи може програма, що не використовує методи штучного інтелекту, мати такі ж властивості?

2. У чому різниця між експертною системою й системою, заснованої на знаннях?

3. Чи є експертною системою програма прогнозування погоди в Одеській області, яка виводить повідомлення такого роду: "Завтра погода не буде відрізнятися від сьогоднішньої"? Припустимо, що вона представляє сьогоднішню погоду в символьному виді, легко модифікується й здатна до розширення, прекрасно працює й може пояснити, чому вона прийшла до певного висновку, вивівши приблизно таке повідомлення: "Добові зміни кліматичних умов у цю пору року малоймовірні".

4. Чи є експертною системою програма, яка формує прогноз погоди на певну дату (скажемо, 16 червня), узявши середні температуру повітря,

кількість опадів, що випали, і кількість сонячних годин 16 червня за всі роки, починаючи з 1900?

5. Чи є система пошуку в мережі World Wide Web експертної? Якщо ні, то яких властивостей їй не вистачає для того, щоб кваліфікувати її як експертну систему пошуку потрібної Web-сторінки?

6. Чому завдання придбання знань є вузьким місцем у проектуванні експертних систем? Які розв'язки пропонуються для усунення такої ситуації?

7. Поясните зауваження про логічну й евристичну адекватність, яке ставиться до мови подання знань.

2.1.1 Лабораторна робота №11. Робота з фактами в середовищі CLIPS

МЕТА: навчити працювати з фактами в середовищі CLIPS.

ЗАВДАННЯ: ознайомитися з поняттям «факт»; ознайомитися синтаксисом введення фактів в середовищі CLIPS; ознайомитися з шаблонами; розглянути приклади.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Факти — одна з основних форм представлення інформації в CLIPS. Факти є фундаментальним поняттям теорії експертних систем і призначені для використання в правилах системи. Кожен факт представляє фрагмент даних, поміщених в поточний список фактів системи (робочу пам'ять).

Факти використовуються для того, щоб внести в систему заздалегідь відомі знання або додати інформацію, отриману в процесі діалогу з користувачем абов результаті обчислень.

Факти бувають простими і складеними. Складові факти є аналогами структур в класичних мовах програмування і визначаються за допомогою спеціальних конструкторів.

В системі CLIPS фактом є список неподільних значень, укладених в дужки. Наприклад:

(Sun is shining)

(The weather is sweet)(Temperature 36) (Model "Opel Astra")

Примітка. Крім явно заданих фактів, CLIPS також автоматично додає зумовлений факт **(initial-fact)** щоразу при виконанні команди (reset). Факт (initial-fact) може використовуватися для визначення моменту запуску механізму логічного висновку, а також неявно присутня в правилах, для яких незадана передумова

*Функція **assert** — додавання факту в список.*

Функція **assert** приймає в якості параметрів послідовність фактів, які підлягають додаванню в список фактів. Синтаксичну структуру виклику функції assert можна представити таким чином:

(assert <факт> <факт> <факт>)

Для того щоб мати можливість спостерігати процес додавання видалення або зміни фактів необхідно викликати пункт меню Execution — Watch і встановити прапорець навпроти пункту «Facts» (рис.2.3).

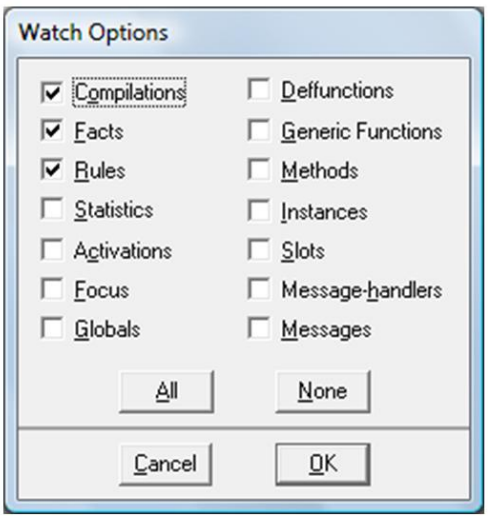

Рисунок 2.3 - Установка параметрів трасування

Розглянемо приклад використання функції **assert**. У діалоговому вікні CLIPS(рис. 2.4) введемо такий вираз:

(Assert (weather is fine))

У тому випадку, команда введена правильно, в список фактів додатися факт(weather is fine) і з'явиться результат виконання:

==> f-*1 (weather is fine)*

<Fact-1>

що означає, що факт був доданий в систему і отримав номер 1.

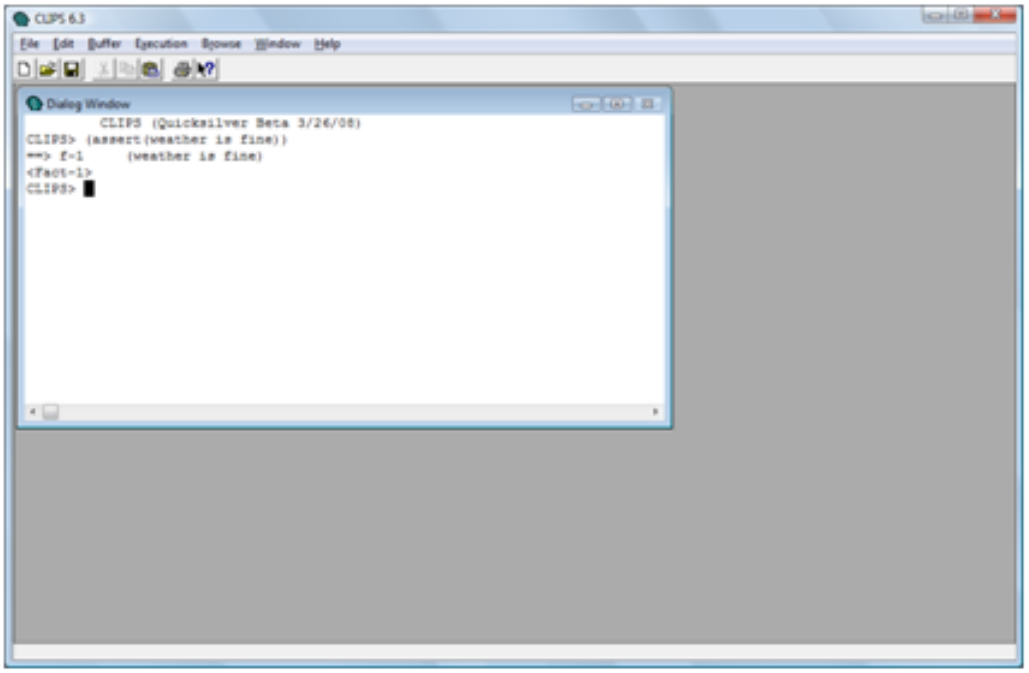

Рисунок 2.4 - Додавання факту

При успішному виконанні функція **assert** повертає адресу останнього доданого факту. Якщо під час додавання деякого факту сталася помилка, команда припиняє свою роботу і повертає значення FALSE.

Перевірити поточний стан списку фактів можна або за допомогою введення в діалоговому вікні команди (facts), або викликавши пункт меню Window —1Fact (рис.2.5)

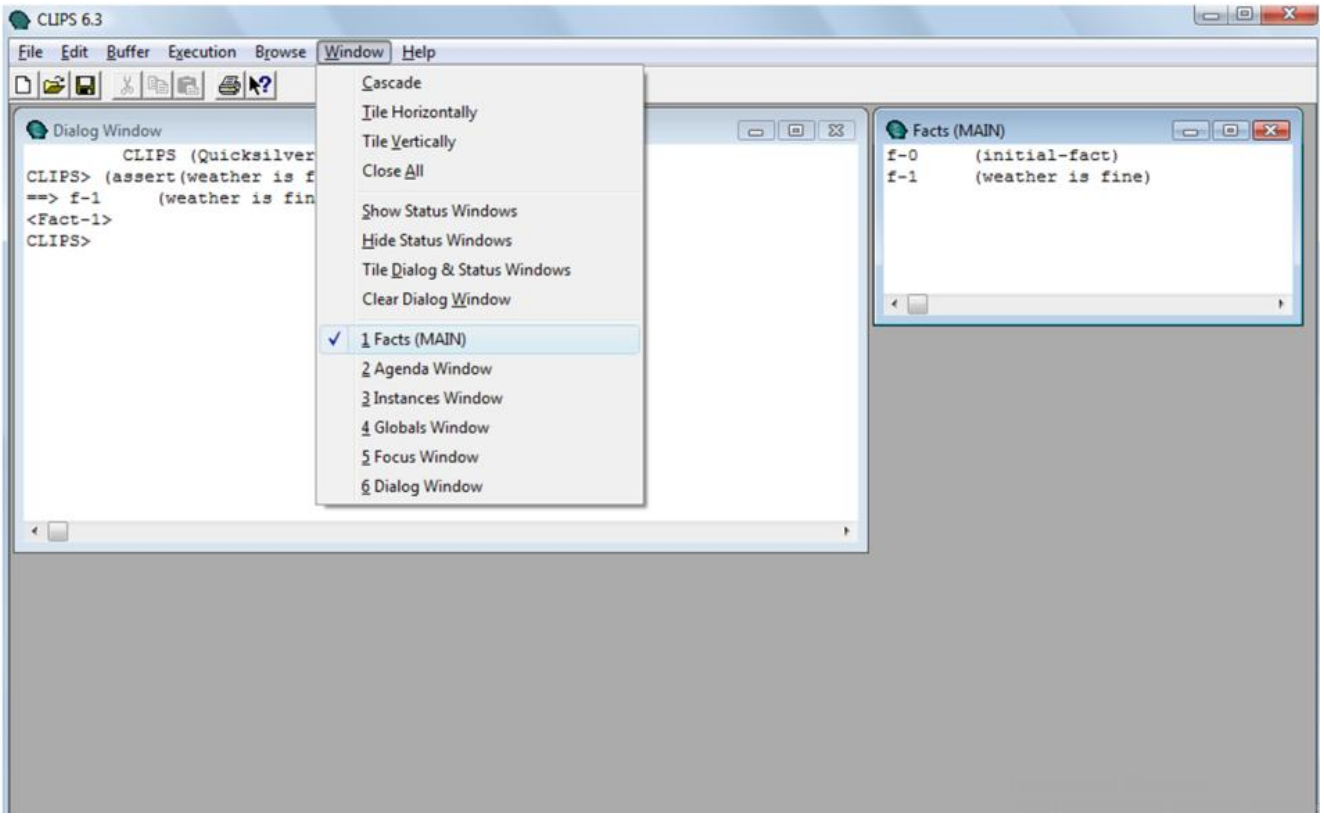

Рисунок 2.5 - Виклик вікна зі списком фактів

Для очищення пам'яті середовища використовується команда *(clear)*.

*Функція **retract** — видалення фактів*

Для видалення фактів з поточного списку фактів в системі CLIPS передбачена функція **retract**. Кожним викликом цієї функції можна видалити довільне число фактів. У разі якщо був включений режим перегляду зміни списку фактів, то відповідне інформаційне повідомлення буде відображатися у вікні CLIPS при видаленні кожного факту.

Синтаксичну структуру виклику функції **retract** можна представити таким чином:

(retract <визначення-факту> <визначення-факту> ...)

або

*(retract * *)*

Аргумент <визначення-факту> може бути або змінної, пов'язаної з адресою факту (адреса факту повертається командою **assert**), або індексом факту без префікса (наприклад, 3 для факту з індексом f-3), або вираз, що обчислюється цей індекс (наприклад, (+ 1 2) для факту з індексом f-3). Якщо в якості аргументу функції **retract** використовувався символ * (зірочка), то з поточного списку будуть видалені всі факти.

Функція **retract** не має значення, що повертається.

Розглянемо роботу функції retract на прикладі. У діалоговому вікні CLIPS введемо такі команди:

(assert (a) (b) (c) (d) (e) (f))(Retract 0 1 2 4)

Результатом виконання наведеної вище послідовності команд буде: Додавання в систему шести фактів.

Видалення з системи фактів з номерами 0, 2 і 4. (рис.2.6)

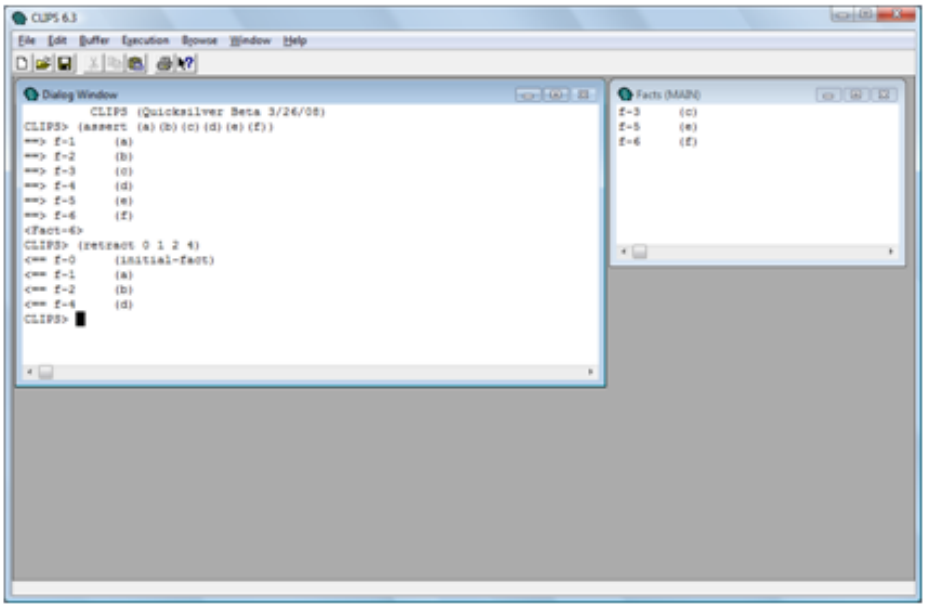

Рисунок 2.6 - Додавання та видалення фактів

За умов успішного виконання зазначених команд у списку залишаться факти (c),(e) та (f).

Невпорядковані факти (шаблони)

Шаблон факту складається з імені факту і визначення полів для зберігання даних, які називаються слотами. Шаблони корисно використовувати для опису будь-якої сутності, що має набір атрибутів. Наприклад, якщо перед нами стоїть мета внести систему інформацію про автомобіль, то його слотами можуть в простому випадку бути колір і марка. Для визначення абстрактної структури шаблону служить спеціальний конструктор *deftemplate*. Синтаксична схема використання конструктора в простому випадку може бути представлена в наступному вигляді:

(deftemplate <Імя_шаблона> ["Необов'язковий коментар"]

<Определеніе_слота>

...

<Определеніе_слота>

)

Слот може бути простим або складеним. У простому слоті може бути збережено одне значення примітивного типу CLIPS, в складеному слоті може зберігатися список з декількох примітивних типів. Ключовими словами для визначення призначення слота є slot для простого слота і multilslot — для складеного. Отже, <визначення слота> в простому випадку складається з:

- ключового слова slot або multislot, що визначає тип слота;
- імені слота, яке є значенням типу symbol.
- необов'язкового обмеження на тип значення, що зберігається в слоті.

Розглянемо приклад шаблону для фактів, що описують людини. У діалоговому вікно CLIPS введемо конструктор:

(deftemplate human "This is template for describing a human"(slot sex)(slot age)

(multislot name)

)

У цьому прикладі ми створили абстрактний шаблон з ім'ям *human*, в якому є 2 простих слота для зберігання статі і віку людини і один складовою слот для зберігання імені та прізвища.

У разі успішного створення шаблону, оболонка повернеться в режим очікування введення без будь-яких повідомлень. В іншому випадку ми отримаємо повідомлення про помилку.

Для перегляду списку шаблонів в системі на поточний момент використовується візуальним інструментом "Deftemplate Manager", який знаходиться в меню "Browse". Існує можливість вивести вміст шаблону в діалогове вікно, натиснувши кнопку Pprint у вікні "Deftemlate Manager" (рис.2.7).

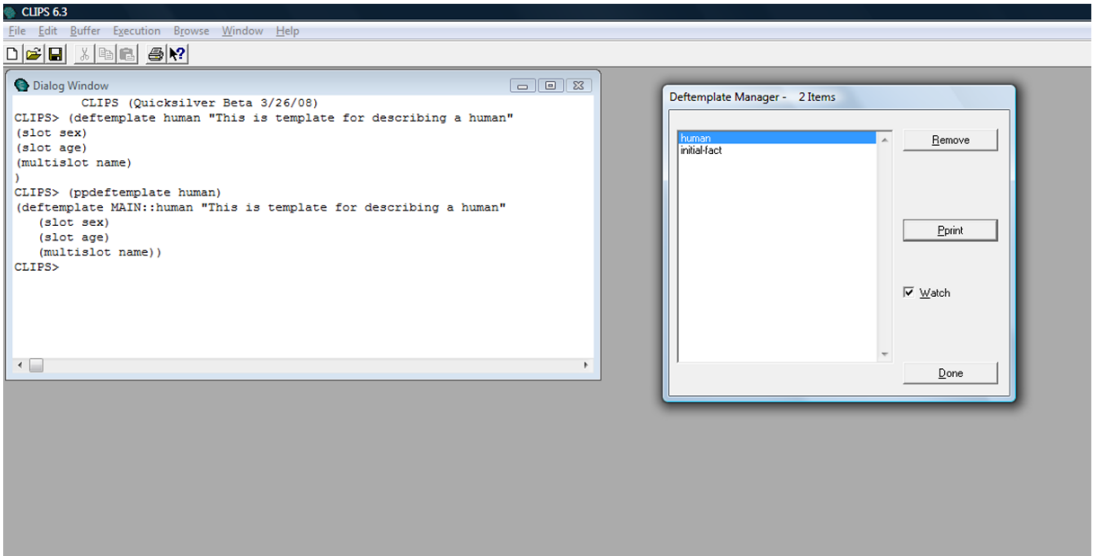

Рисунок 2.7 - Результат створення шаблону і вікно менеджера шаблонів

Після визначення шаблону, ми можемо вносити інформацію про людей у вигляді фактів. Невпорядкований факт визначається шляхом вказівки імені його шаблону з подальшим перерахуванням імен і значень слотів. Наприклад,неврегульованим фактом є наступна конструкція:

(Human (sex male) (age 25) (name Petrov Sergey)).

Додамо в систему факти про кількох людей. Для цього використовуємо вжевідому функцію **assert**.

(assert (human

(sex male)(age 25)

(name Petrov Sergey)

); Дужка закриває факт

); дужка, що закриває виклик функції assert

(assert (human

(sex female)(age 23)

(name Ivanova Dasha)

)

); дужка, що закриває виклик функції assert

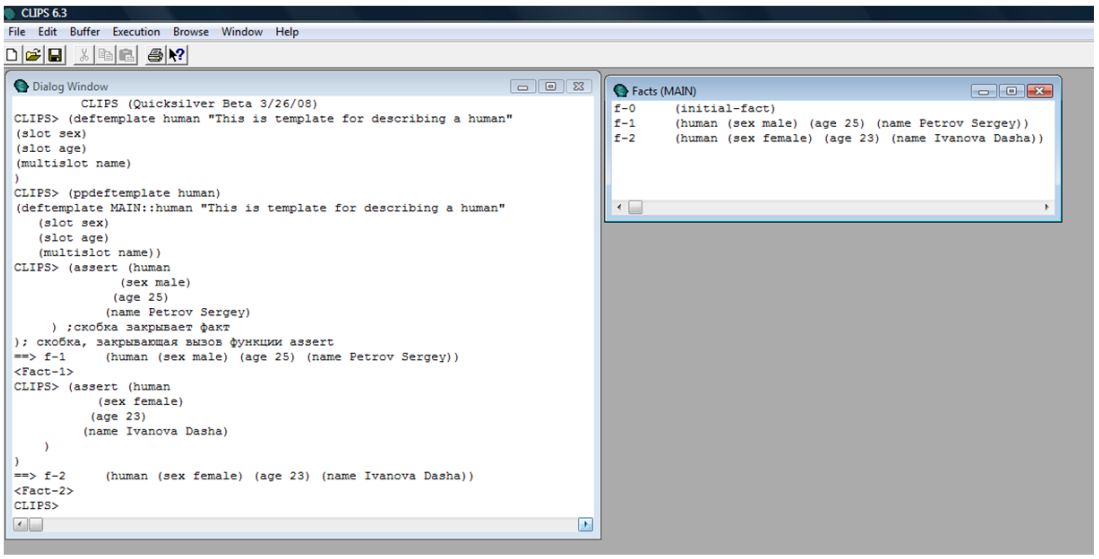

Рисунок 2.8 - Додавання фактів за шаблоном

Команди modify, reset, clear

Для зміни шаблонного факту служить команда ***modify***. Як параметр команда приймає ухвалу факту (змінна-адресу або індекс факту), а також нові значення певних слотів. Наприклад, виконаємо послідовно команди:

(clear)

(deftemplate weather(slot temperature) (slot windspeed)

)

(assert

(weather (temperature 10) (windspeed 0))

)

(modify 1 (temperature 20))

Наведені вище команди визначають шаблон weather, додають в систему факт про температуру і швидкості вітру, а потім командою modify виробляються зміни значення слота temperature. Прослідкуйте за виконанням команди modify, включивши трасування фактів (рис.2.4). За суттю, команда видаляє існуючий факті та додає новий (рис.2.9).

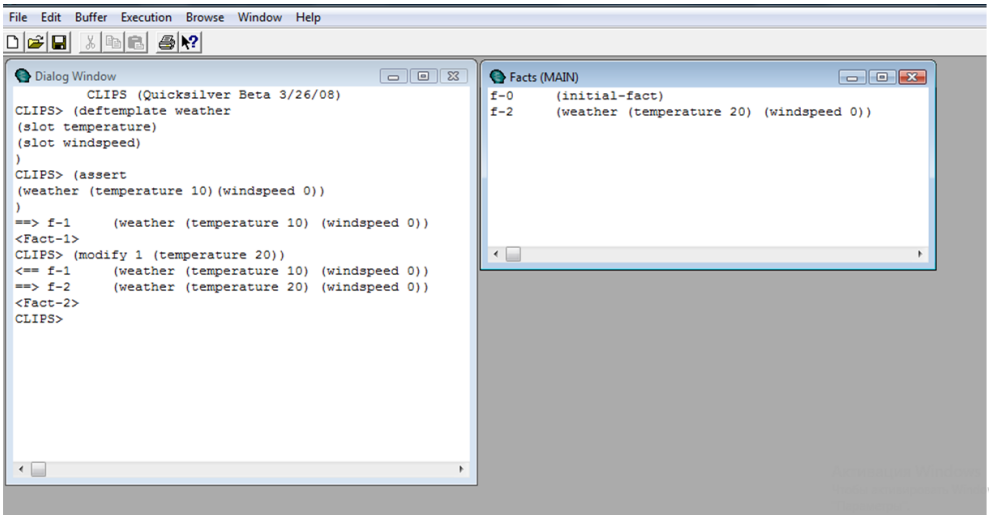

Рисунок 2.9 - Результат роботи функції modify

<u>Примітка.</u> При виконанні команди modify індекс факту змінюється!

Команда **(reset)** очищає список фактів, а потім додає в нього факти, оголошені конструкторами deffacts, включаючи факт *(initial-fact)*. Зазвичай використовується в поєднанні з командою **(run)** для перезапуску написаної програми.

На відміну від команди **(reset)**, команда **(clear)** виконує глибоке очищення виконуваної середовища. Очищаються не тільки факти, але також всі визначені списки, правила змінні, шаблони. Команда **(clear)** не додає в пам'ять системи ніяких фактів.

Питання для самоконтролю:

1. Що називають «фактом»?
2. Яке призначення функції (assert)?
3. Яке призначення функції (retract)?
4. Яке призначення функції (modify)?
5. Яке призначення команда (run)?
6. Яке призначення функції (reset)?
7. Яке призначення функції (clear)?
8. У чому відмінність функцій (clear) і (reset)?

Завдання для самостійного виконання:

1. Визначте і внесіть в систему шаблон, що описує мобільний телефон (модель,колір, тип корпусу, власник).
2. Додайте в систему кілька моделей мобільних телефонів.

Форма звіту: вивести на екран факти про мобільні телефони у вікні FACTS.

2.1.2 Лабораторна робота №12. Робота з правилами в середовищі CLIPS

МЕТА: Навчити студентів використовувати правила в середовищі CLIPS.

ЗАДАЧІ: ознайомити з поняттям «правило»; ознайомитися синтаксисом введення правил в середовищі CLIPS; ознайомитися з використанням змінних вправилах; розглянути приклади.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Правила в CLIPS служать для представлення евристик або так званих "емпіричних правил", які визначають набір дій, виконуваних при виникненні деякої ситуації. Правила складаються з передумов і дії.

Передумови правила являють собою набір умов (або умовних елементів), які повинні задовольнятися, для того щоб правило виконалося.

Передумови правил задовольняються в залежності від наявності або відсутності деяких заданих фактів у списку фактів (про який було розказано в попередньому розділі) або деяких створених об'єктів, що є екземплярами класів, визначених користувачем (про які буде показано нижче).

Один з найбільш поширених типів умовних виразів у CLIPS - зразки (*patterns*). Зразки складаються з набору обмежень, які використовуються для визначення того, чи задовольняє деякий факт або об'єкт умовного елементу. Іншими словами, зразок задає деяку маску для фактів або об'єктів. Процес зіставлення зразків фактами або об'єктів називається процесом зіставлення зразків (pattern- matching).

CLIPS надає механізм, званий механізмом логічного висновку (*inference engine*), який автоматично зіставляє зразки з поточним списком фактів і певними об'єктами в пошуках правил, які застосовні в даний момент.

Використання конструктора ***defrule***

Для оголошення і додавання нових правил в базу використовується конструктор ***defrule***.

Синтаксична схема конструктора може бути представлена наступним чином: *(defrule <ім'я_правила> [<необов'язкові_комментарі>] [<необов'язкове_визначчення_властивості_правила>]<нередумови>; ліва частина правила =>; спец. символ<наслідок>; права частина правила)*

Ім'я правила повинно бути значенням типу *symbol* і не може бути зарезервованим словом мови CLIPS. У разі повторного оголошення правила з однаковим ім'ям, старе правило буде видалено, навіть якщо нове правило неможливо буде додати внаслідок синтаксичної помилки або з будь-яких іншихпричин.

Коментар є необов'язковим — зазвичай у ньому описують призначення правила. Коментар повинен бути значенням типу *string*, значення коментаря зберігається разом з правилом.

Визначення властивості правила складається з ключового слова *declare*, і наступного за ним вказівки властивості. У правила може бути дві властивості

— *salience* і *auto-focus*.

<визначення-властивості-правила> =(declare <властивість-правила>)

<властивість-правила> = (salience <целочисленное вираз>)

або

(Auto-focus <логічний вираз>)

Властивість *salience* дозволяє користувачеві призначати певний пріоритет для своїх правил. І оголошений пріоритет повинен бути вираженням, які мають цілі значення з діапазону від -10 000 до +10 000. Вираз, що представляє пріоритет правила, може використовувати глобальні змінні і функції. Проте краще не використовувати в цьому виразі функцій, що мають побічну дію. У разі якщо пріоритет правила явно не заданий, йому присвоюється значення за замовчуванням — 0. Чим більше число, що визначає пріоритет, тим вище пріоритет у правила.

Значення пріоритету може бути обчислено в одному з трьох випадків:

1. при додаванні нового правила;
2. при активації правила;

3. на кожному кроці основного циклу виконання правил.

Два останніх варіанти називаються динамічним пріоритетом (*dynamic salience*). За замовчуванням значення пріоритету обчислюється тільки під час додавання правила. Для зміни цієї установки можна використовувати діалогове вікно пункту меню Execution — Options. У діалоговому вікні вкажіть необхідний режим обчислення пріоритету за допомогою списку Salience Evaluation (рис.2.10).

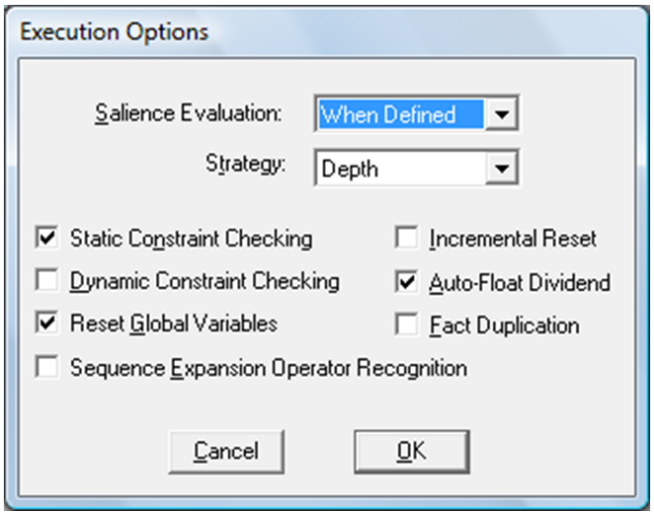

Рисунок 2.10 - Вікно налаштування параметрів механізму логічного висновку

Передумова чи ліва частина правила задається набором умовних елементів, який зазвичай складається з умов, застосованих до деяких зразкам. Заданий набір зразків використовується системою для зіставлення з наявними фактами та об'єктами. Всі умови в лівій частині правила об'єднуються за допомогою неявного логічного оператора *and*. Якщо в лівій частині правила не вказано ні один умовний елемент, CLIPS автоматично підставляє умова-зразок *initial-fact*. Таким чином, правило активізується всякий раз при появі в базі знань факту *initial-fact*.

Наслідок або *права частина правила* містить список дій, виконуваних при активізації правила механізмом логічного висновку. Дії правила виконуються послідовно, але тоді і тільки тоді, коли всі умовні елементи в лівій частині цього правила задовольняються.

Для поділу правої і лівої частини правил використовується символ =>.

Приклад.

Необхідно набрати наступний лістинг у вікні введення:

(clear)

(defrule MyRule "This is my first rule"(initial-fact); передумова

=>

(printout t crlf)

(printout t "HELLO!" Crlf)(printout t crlf)

)

Потім виділити введений текст і натиснути Ctrl-M (рис.2.11). Виділені команди будуть виконані в діалоговому вікні. Там же будуть виведені повідомлення про помилки.

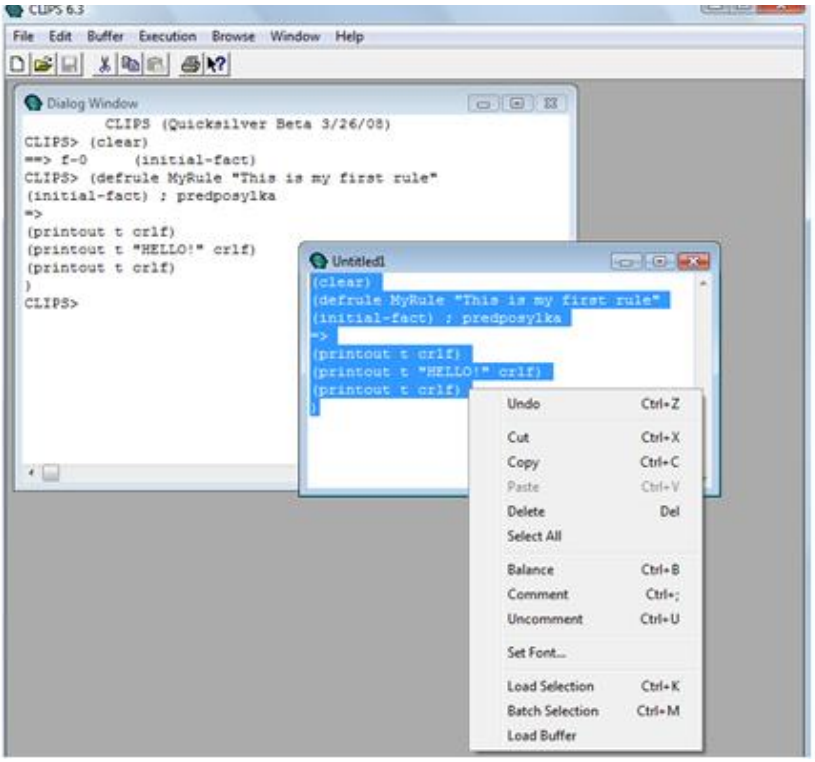

Рисунок 2.11 - Відправлення на виконання виділеної частини програми

Наявність правила в системі можна проконтролювати шляхом виклику пунктуменю Browse — Defrule Manager (рис.2.12).

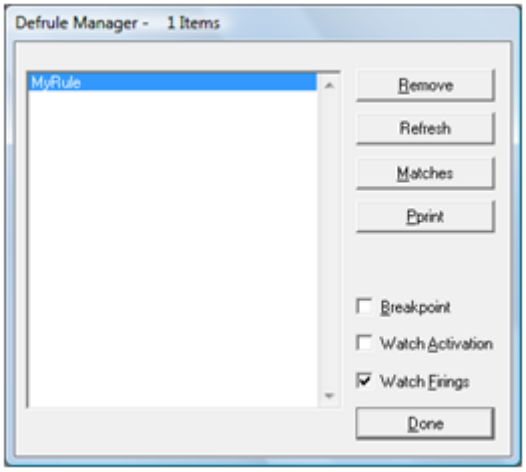

Рисунок 2.12 - Менеджер правил, внесених в систему

Якщо додавання правила пройшло вдало і в менеджері фактів відображається рядок MyRule (рис.2.12), можна запустити механізм логічного виведення. Для цього потрібно виконати в командному діалоговому вікні послідовно дві команди:

(reset)(run)

Команда ***(reset)*** очищає список фактів, а потім додає в нього факти, оголошеніконструкторами *deffacts*, включаючи зумовлений факт *(initial-fact)*.

Команда **(run)** запускає процес логічного висновку. По суті, починається перевірка умов для правил, які були введені в оболонку CLIPS раніше. Перевірка передумови для правила MyRule буде вдалою — передумова буде задоволена, оскільки в системі існує факт *(initial-fact)*, що відповідає зразку у правилі.

Для створення правила, активація якого відбудеться при появі в системі певнихфактів, достатньо просто перерахувати ці факти в передумові правила.

Так як передумова задоволена, буде виконано наслідок правила, тобто виконанапослідовність команд, записана після оператора =>, і в діалоговому вікні відобразитися повідомлення «HELLO!» (рис.2.13).

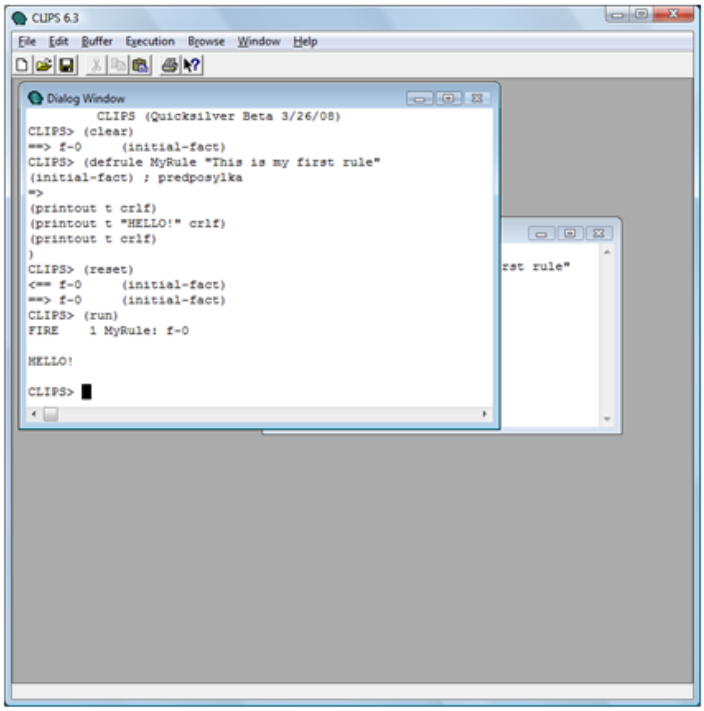

Рисунок 2.13 - Результат роботи правила MyRule

Приклад 2.

Ввести наступний лістинг у вікно введення:*(clear)*

(defrule FinalRule; poslednee pravilo

(All Rules Activated); nalichie fakta dla aktivacii

=>

(printout t "All rules has been activated. THE END" crlf)

)

(defrule SecondRule; vtoroe pravilo

(First Rule Activated); nalichie fakta dla aktivacii

=>

(printout t "II rule activated. Adding fact ..." crlf)(assert

(Second Rule Activated)

)

); dobavlaetsa fakt aktivacii vtorogo pravila(defrule ThirdRule; tretje pravilo

(First Rule Activated)

(Second Rule Activated); dla aktivacii trebuetsa nalachie dvuh faktov

=>

(printout t "III rule activated." Crlf)

(assert; dobavlen fakt aktivacii vseh 3 pravil(All Rules Activated)

)

)

(defrule FirstRule; pervoe pravilo=>; Aktivacia prohodit pri nalichii inintal-fact (printout t "I rule activated. Adding fact ..." crlf)(assert

(First Rule Activated)

)

)

Потім виділити код і передати його на виконання середовищі, натиснувши Ctrl+ M (рис.2.14).

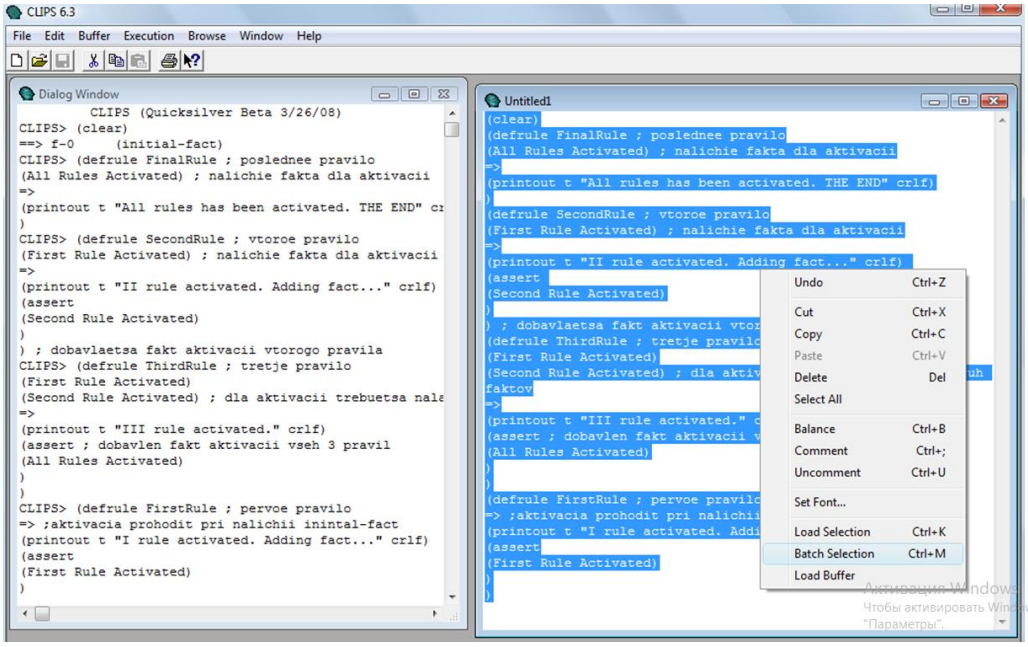

Рисунок 2.14 - Додавання правил

У менеджері правил необхідно перевірити додавання чотирьох правил в систему: First Rule, Second Rule, Third Rule, Final Rule (рис.2.15).

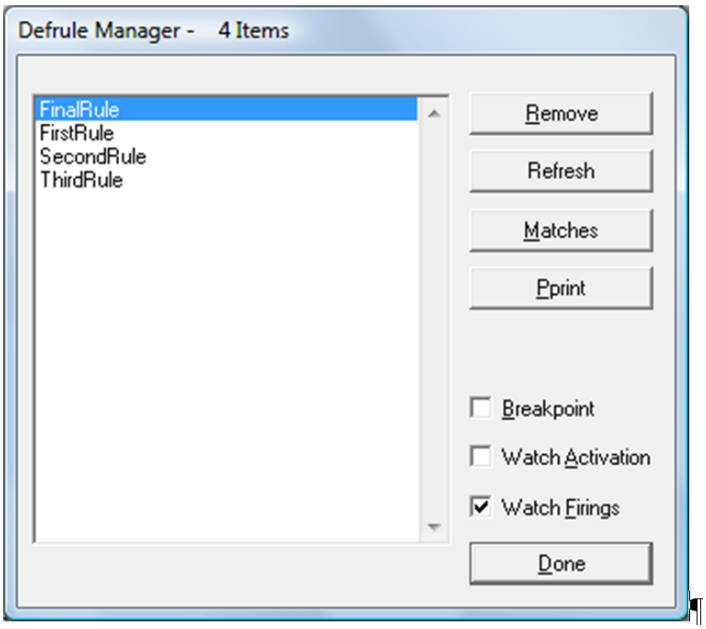

Рисунок 2.15 - Відображення доданих правил в менеджері правил

Правила, задані в даній програмі, повинні активуватися за наступною схемою:

перше правило — під час запуску програми;

друге — при активації першого;

третє — при успішній активації першого і другого правила; фінальне — за успішною активацією всіх трьох правил.

Необхідно виконати запуск програми, давши команди ***(reset)*** і ***(run)*** і переконатися, що правила виконуються в необхідній послідовності (рис.2.16). У коді програми визначення правил записано в розкид — першим визначено фінальне правило, потім другий і т.д. Послідовність визначення правил може бути важлива лише в разі настання однакових умов виконання для правил з однаковим пріоритетом. У цьому випадку застосовується стратегія вирішення конфліктів. В даному випадку послідовність визначення правил абсолютно не важлива, так як всі правила мають такі умови, які забезпечують відсутність конфліктів і гарантують бажану послідовність виконання. При старті логічного висновку єдиним правилом, умова яке задовольняється, є правило FirstRule, тому для активації інших правил немає відповідних фактів. Всередині свого тіла правило FirstRule додає в систему факт *(First Rule Activated)*, що говорить про його активації. При наступній перевірці, в системі виявляється факт *(First Rule Activated)*, який підходить для активації правила SecondRule, яке в свою чергу додає в систему свій факт.

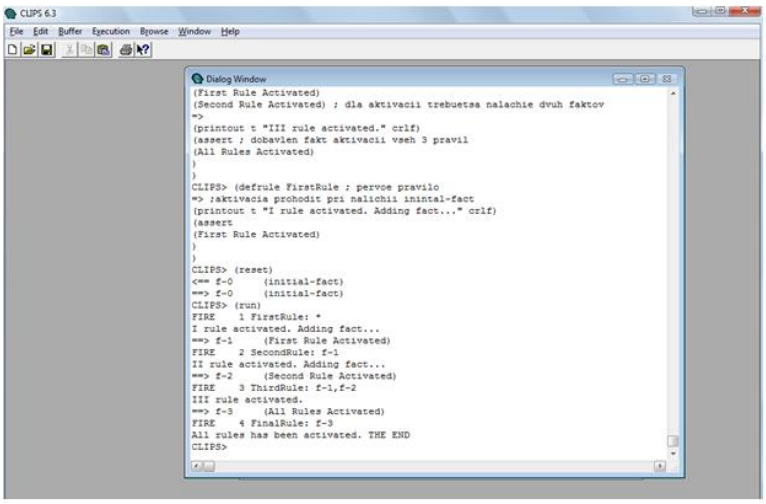

Рисунок 2.16 - Результат виконання програми про чотири правилах

Використання зразків у правилах

У загальному випадку ліва частина правила (передумова) містить список умовних елементів (conditional elements або CEs), які повинні задовольнятися для активації правила. Існує вісім типів умовних елементів, які використовуються в лівій частині правил: CEs-зразки, test CEs, and CEs, or CEs, not CEs, exists CEs, forall CEs і logical CEs.

Зразки — це найбільш часто використовуваний умовний елемент. Він містить обмеження, які служать для визначення, чи задовольняє який-небудь факт зі списку фактів заданим зразком.

Умова *test* використовується для оцінки вирази, як частини процесу зіставленняобразів.

Умова *and* застосовується для визначення групи умов, кожне з якої повинна бути задоволена.

Умова *or* — для визначення одного умови з деякої групи, яке має бути задоволене.

Умова *not* — для визначення умови, яке не повинно бути задоволене.

Умова *exists* — для перевірки наявності, принаймні, одного збігу факту (або об'єкта) з деяким заданим зразком.

Умова *logical* дозволяє виконати додавання фактів і створення об'єктів в правій частині правила, пов'язаних з фактами та об'єктами, що збіглися із заданим зразком у лівій частині правила (підтримка достовірності фактів в базі знань)

Зразки

Зразок складається зі *списку обмежень полів*, *групових символів* (wildcards) і *змінних*, які використовуються для пошуку безлічі фактів, відповідних бажаного зразком. Таким чином, зразок як би визначає маску, якій повинні

відповідати факти. Якщо в списку фактів знайдений факт, відповідний обмеженням, то умовний елемент вважається задоволеним.

Обмеження полів — це набір обмежень, які використовуються для перевірки простих полів або слотів об'єктів. Обмеження полів можуть складатися тільки з одного символьного обмеження, однак, кілька обмежень можна з'єднувати разом. Додатково до символьних обмеженням, CLIPS підтримує три інших типу обмежень: *об'єднуючі обмеження*, *предикатні обмеження* та *обмеження, які повертають значення*.

Символьні обмеження — це обмеження, що визначають точну відповідність між полями факту і зразком. Символьне обмеження повністю складається з констант, таких як речові і цілі числа, значення типу *symbol*, рядки або імена об'єктів.

Наприклад, в розглянутому вище правилі:

(defrule FinalRule

(All Rules Activated); зразок з символьним обмеженням.

=>

(printout t "All rules has been activated. THE END" crlf)

)

рядок (All Rules Activated) є умовним елементом з символьним обмеження, що накладаються на необхідний для активації правила факт жорсткі рамки відповідності заданим шаблоном. Іншими словами, факт, який може активувати правило, повинен бути точно таким же як зазначено в умовному елементі. Символьні обмеження можна накладати і на невпорядковані факти. Наприклад, для шаблону *human*, визначеного в лабораторній роботі №11, можлива така запис правила для реакції на досягнення Сергієм Петровим 25

років (припускаємо, що ця подія буде ознаменовано додаванням факту в нашу систему):

(defrule PetrovHas25(human

(sex male)(age 25)

(name Sergey Petrov)

)

=>

(printout t "Sergey Petrov is 25 years old")

)

Групові символи для простих і складених полів

У CLIPS є два різних групових символу, які використовуються для зіставлення полів у зразках. CLIPS інтерпретує ці групові символи як місце для підстановки деяких частин фактів. Груповий символ для простого поля записується за допомогою знака ? , Який відповідає *одному будь-якому значенню*, збереженому в заданому полі. Груповий символ складеного поля записується за допомогою знака $? і відповідає послідовності значень, збереженої в складеному поле. Послідовність може бути порожня. Групові символи для простих і складених полів можуть комбінуватися в будь-якій послідовності. Не можна використовувати груповий символ складеного поля для простих полів. Розглянемо *приклад*:

(defrule FirstOrSecondRule

(? Rule Activated); зразок с груповим символом.

=>

(printout t "1st or 2nd rule has been activated" crlf)

)

Зразок *(? Rule Activated)* активує своє правило при появі в системі будь факту, в якому на першому місці буде стояти будь-яке значення, а на другому і третьому відповідно значення *Rule* і *Activated*. Таким чином, у розглянутій вище програмі

«про чотири правилах» факти *(First Rule Activated)* і *(Second Rule Activated)* будуть задовольняти зразком. Причому правило активується при появі кожного з фактів, тобто у нашому прикладі — двічі. Звернемо увагу, що факт *(All Rules Activated)* не зможе активувати правило *FirstOrSecondRule*, тому його другий елемент «Rules» не збігається з обмеженням «Rule».

Розглянемо зразок зі складним полем *(data 1 $?)*. Як приклад наведемо лише кілька фактів, які підходять під обмеження:

(data 1 blue red) (data 1 blue 3 red)

(data 1 yellow red green 10)

і т.д. Але факт *(data 2 red blue)*, вже не буде підходити під задані обмеження, в силу різних другого елементу.

Якщо розглянути зразок *($? Activated)* в прикладі про три правила, то можна переконатися, що він підійде під всі факти, які додаються по ходу виконання програми. Можна записати правило:

(defrule AnyOfThreeRule

($? Activated); зразок с груповим символом.

=>

(printout t "Some rule has been activated" crlf)

)

яке буде виконуватися кожного разу після активації першого, другого або третього правила.

З складовими фактами ситуація аналогічна. Так, правило:

(defrule SomebodyIs25(car

(sex ?) (age 25) (name $?)

)

=>(printout t "Somebody is 25" crlf))

активується, якщо в системі з'явиться факт, що описує людини будь-якої статі, віком 25 років. Причому кількість елементів у складеному слоті name значення не має і може бути будь-яким.

Змінні, пов'язані з простими і складовими полями

Групові символи замінюють будь-які поля зразка, і можуть приймати які завгодно значення цих полів. Значення поля може бути пов'язані зі змінними шляхом запису імені змінної безпосередньо після знака групового символу. Ім'я змінної має бути значенням типу *symbol* і обов'язково починатися з літери. В імені змінної не дозволяється використовувати лапки, тобто рядок не може використовуватися як ім'я змінної або її частину. Змінні, яким були присвоєні значення в результаті виконання зіставлення, можна використовувати в правій частині правила. Область видимості такої змінної обмежується тілом правила.

Розглянемо приклад з уже відомим нам шаблоном з лабораторної роботи 11. Необхідно додати список зумовлених фактів:

(deffacts people

(human (sex male)(age 25) (name Sergey Petrov)) (human (sex female)(age 23) (name Ivanova Dasha))

)

Після цього додаємо правило:

(defrule ListOfPeople(human

(sex ?x)

(age ?y)

(name $?z)

)

=>

(printout t ?z " is " ?x " and " ?y crlf)

)

При запуску програми, що містить вищенаведене правило, на екран буде виведено інформацію про всі факти people, які присутні в списку, причому із зазначенням значень полів цих фактів.

Відповідь системи:

(Ivanova Dasha) is female and 23(Sergey Petrov) is male and 25

Питання для самоконтролю:

1. Що називають правилом?
2. Уявіть конструкцію *defrule* в загальному вигляді.
3. Що означає властивість *salience*?

Завдання для самостійного виконання:

Модифікуйте виконане завдання лабораторної роботи №11 так, щоб:

- отримати список всіх власників мобільних телефонів;
- отримати список власників телефонів марки Samsung.

Форма звіту: вивести на екран інформацію про власників мобільних телефонів марки Samsung.

2.1.3 Лабораторна робота №13. Методи вирішення конфліктів

МЕТА: навчити використовувати методи вирішення конфліктів при роботі з правилами.

ЗАДАЧІ:

1. ознайомити студентів зі стратегіями вирішення конфліктів;
2. розглянути роботу правил при використанні різних стратегій вирішенняконфліктів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

При реалізації прямого виводу в продукційних базах знань машина логічних висновків зіставляє ліві частини (антецеденти) правил з базою даних і поміщає правила, антецеденти яких задовольняються, в Агенді (конфліктна безліч). Агенда являє собою список всіх правил, умови яких задовольняються, але які ще не були виконані. Агенда працює аналогічно стеку - правило, яке повинне бути виконане першим є верхнім правилом в Агенді.

Коли правило стає активним (умови в його лівій частині задовольняються), воно поміщається в агенді у відповідності з наступними правилами:

Знову керуючі правила вміщуються над усіма правилами з більш низькою значимістю (salience) і нижче всіх правил з більш високою значущістю.

1. Для визначення місця серед правил рівної значимості використовується поточна стратегія вирішення конфлікту.
2. Якщо в результаті додавання або видалення факту одночасно

активізуються кілька правил і кроки 1 і 2 не дозволяють виконати упорядкування, то ці правила упорядковуються між собою довільно (але не випадково).

У CLIPS підтримується сім стратегій вирішення конфліктів: *"вглиб" (depth), "вшир" (breadth), "простоти" (simplicity), "складності" (complexity), LEX, MEA* і*випадкового вибору (random)*. За замовчуванням використовується стратегія вглиб.

Поточна стратегія може бути встановлена командою *set-strategy*, при цьому агенту буде впорядковано на основі нової стратегії. Синтаксис команди:

(set-strategy <strategy>),

де <strategy> :: = depth | breadth | simplicity | complexity | lex | mea | random

За замовчуванням використовується стратегія depth.

Стратегія "вглиб". Нові правила що активуються вміщуються в агенді над усіма правилами такої ж значущості. Наприклад, нехай факт f-1 активує правила rule-

1 і rule-2, а факт f-2 активує правила rule-3 і rule-4. Тоді якщо f-1 встановлюється раніше, ніж f-2, то rule-3 і rule-4 опиняться в агенді вище правил rule-1 і rule-2. Проте становище правила rule-1 щодо правила rule-2 і правила rule-3 щодо правила rule-4 буде довільним.

Стратегія "вшир". Знову активуються правила вміщуються нижче всіх правил з такою ж значущістю. Наприклад, нехай факт f-1 активує правила rule-1 і rule-2, а факт f-2 активує правила rule-3 і rule-4. Тоді, якщо f-1 встановлюється раніше, ніж f-2, то rule-1 і rule-2 опиняться в Агенда вище правил rule-3 і rule-4. Однак, положення правила rule-1 щодо правила rule-2 і правила rule-3 щодо правила rule-4 буде довільним.

Стратегія "простоти". Серед правил однаковою значимості, знову активуються правила вміщуються над усіма правилами з рівною або більшою специфічністю (specificity). Специфічність правила визначається числом порівнянь, які повинні бути виконані в лівій частині правила. Кожне порівняння з константою або попередньо пов'язаної змінної збільшує специфічність на одиницю. Кожен виклик функції, зроблений з лівої частини правила в умовному елементі з предикативним обмеженням (:), обмеженням повертається значенням (=) або УЕ-перевіркою (test) збільшує специфічність на одиницю. Булеві функції "і", "або", "не" не збільшують специфічність правила, але їх аргументи збільшують. Виклики функцій, що виконуються з функцій не збільшують специфічність. Наприклад, наступне правило:

(defrule example(item ?x ?y ?x)

(test (and (numberp ?x) (> ?x (+ 10 ?y)) (< ?x 100)))

=>)

має специфічність 5 (вважаються оператори (item? x? y? x),? x, numberp,>, <).

Стратегія "складності". Серед правил однаковою значимості, знову активуються правила вміщуються над усіма правилами з рівною або меншою специфічністю.

Стратегія LEX. Для визначення місця правила в агенді серед правил з однаковою значимістю в першу чергу використовується новизна зразків, які активізують дане правило. Кожен факт і екземпляр позначаються "тимчасовим тегом" для зазначення його новизни по відношенню до всіх інших фактам і екземплярам в системі. Для визначення місця розташування правила в агенді зразки (факти або екземпляри), пов'язані з активацією кожного правила упорядковано зменшенням новизни. Правило з більш пізнім

зразком поміщається вище правил з більш ранніми зразками. Щоб визначити відносний порядок розміщення двох правил, відсортовані тимчасові теги цих зразків, що активують ці правила, порівнюються попарно починаючи з самих більших значень. Порівняння продовжується до тих пір, поки не буде виявлено, що часовий тег однієї активації більше відповідного тимчасового тега інший активації. Правило з великим значенням тимчасового тега розміщується в агенді вище іншого правила.

Якщо одне правило має більше зразків, ніж інше, а всі порівнювані тимчасові теги ідентичні, то правило з великим числом тимчасових тегів поміщається вище. Якщо два правила мають рівну новизну, правило з більш високою специфічністю поміщається вище правила з більш низькою специфічністю.

Стратегія MEA. Для визначення місця правила в агенді серед правил рівної значимості в першу чергу використовується тимчасової тег зразка, пов'язаного з першою умовою в правилі. Правило, у якого тимчасової тег першого зразка (умовного елемента) більше тимчасових тегів перших зразків інших правил, поміщається в агенді вище них.

Стратегія LEX використовується для визначення місця правила, якщо тимчасові теги перших зразків рівні.

Стратегія випадкового вибору (Random Strategy). Кожній активації зіставляється випадкове число, яке використовується для визначення її місця розташування в агенді серед активацій рівної значимості. Це випадкове число зберігається, коли стратегія змінюється, так що при поверненні до випадкової стратегії відновлюється той же порядок (серед активацій, які перебували в агенді, коли стратегія була змінена).

Питання для самоконтролю:

1. Що називають агендою?

2. На підставі яких умов правила потрапляють в агенду?

3. Які стратегії дозволів конфліктів використовуються в CLIPS? В чому полягає їх особливість?

Завдання для самостійного виконання:

1. Використовуючи редактор clipsedt.exe сформувати за допомогою конструкції deffacts початковий набір з п'яти довільних фактів (далі позначаються (a), (b), (c), (d) і (e))

2. Сформувати набір правил, де (n), (m), (p), (r), (s) і (t) - деякі довільно вибрані факти (в квадратних дужках вказана значимість правила). Зберегти підготовленіконструкції у файлі <file_name> .clp.

(A) (b) => (m) [5000]

(A) (c) => (n) [6000]

(B) (c) (d) => (p) [5000] (a) (d) (c) => (r) [6000](M) (n) => (s) [6000]

(N) (p) (r) => (t) [5000]

3. Завантажити середу CLIPS (файл clipswin.exe). Активізувати вікна "Facts Window" і "Agenda Window". За допомогою команди Load Constructs меню File (або «гарячої» комбінації Ctrl-L) завантажити факти і правила з файлу

<file_name> .clp.

4. Виконати початкову установку командою (run) («гаряча» комбінація - Ctrl- U). Зафіксувати стан списку фактів і Агенда.

5. Виконати в покроковому режимі обробку правил («гаряча» комбінація - Ctrl- T), фіксуючи після кожного кроку стан Агенда і списку фактів.

6. Повторити дії п. 4 і 5 при різних стратегіях вирішення конфліктів. Для зміни стратегій використовувати пункт Options-Execution ..

Форма звіту: Зафіксувати і пояснити результати, отримані в результаті використання різних стратегій вирішення конфліктів.

2.2 Основні етапи розробка експертних систем

2.2.1 Лабораторна робота №14. Робота з умовними елементами CLIPS

МЕТА: вивчити умовні елементи середовища CLIPS.

ЗАДАЧІ: ознайомити студентів з умовними конструкціями CLIPS ситуаціями їхвикористання.

ТЕОРЕТИЧНІ ВІДОМОСТІ

1. Умовний елемент or

До цих пір всі розглянуті правила містили між шаблонами заданий умовний елемент **and**. Це означає, що запуск правила відбуваєится, тільки якщо всі шаблони приймають істинне значення. Крім того, в мові CLIPS передбачена можливість задавати в лівій частині правил і явний умовний елемент **and**, і явний умовний елемент **or**.

Приклад 1

У разі затоплення (flood) АБО спалаху (fire) електрична установка повинна бути вимкнена (shut down)

```
(defrule shutdown(or (fire)
(flood))
=>
(printout t "SHUT DOWN!" crlf)
)
```

2. Умовний елемент and

За своїм призначенням умовний елемент **and** є протилежним умовного елементу **or**. В останньому випадку для активізації правила досить використовувати будь-який з кількох умовних елементів, а в першому

випадку (коли застосовується умовний елемент **and**) потрібно, щоб були виконані успішно перевірки за допомогою всіх умовних елементів. Система CLIPS автоматично включає ліву частину правила в неявний умовний елемент **and**.

Приклад 2

У разі затоплення (flood) І спалаху (fire) електрична установка повинна бути вимкнена (shut down)

(defrule shutdown(and (fire) (flood))

=>

(printout t "SHUT DOWN!" crlf)

)

Умовний елемент not

Даний умовний елемент використовується для визначення умови, яке не повинно бути задоволене. Іноді виникає така необхідність, щоб активізацію правила можна було здійснити виходячи з відсутності якогось конкретного факту в списку фактів. Система CLIPS дозволяє задавати умова відсутності деякого факту в лівій частині правила за допомогою умовного елемента **not**.

Приклад 3

Якщо вода тепла і температура повітря висока, а день вихідний, то цілком припустимо відпочити на пляжі.

(defrule high-flow-rate(temp high)

(water warm)

(not (working day))

=>

(printout t "go to the seaside" crlf))

3. Умовний елемент exists.

Даний елемент використовується для перевірки наявності хоча б одного збігу факту (або об'єкта) з деяким заданим зразком.

Приклад 4

При наявності будь-якої загрози, вивести повідомлення про неї.

(defemplate emergency (slot type)

)

(defrule alert(exists (emergency))

=>

(printout t "Operator Alert" crlf))

4. Умовний елемент forall

Умовний елемент **forall** дозволяє визначити, що деякий заданий умова виконується для всіх заданих умовних елементів.

Приклад 5.

Правило all-students-passed визначає, чи пройшли всі студенти читання, письмоі арифметику, використовуючи умову **forall**:

(defrule all-students-passed(forall (student ?name) (reading ?name)

(writing ?name) (arithmetic ?name))

=>

(printout t "All students passed." crlf))

При цьому дане правило задовольняється, поки немає жодного студента. При додаванні факту (student Bob) правило перестає задовольнятися, т. К.

Немає фактів, що підтверджують, що Bob пройшов всі необхідні пред¬мети. Правило не почне задовольнятися й після додавання фактів (reading Bob) і (writing Bob). А ось після додавання факту (arithmetic Bob) правило буде активовано і зможе вивести на екран відповідний запис. Якщо додати факт (student John), правило знову не задовольняється, так як один з студентів (John) не вивчив всі необхідні предмети. Використовуючи умовний елемент **exists**, можна змінити це правило так, щоб воно не виконувалося у разі відсутності студентів.

5. Умовний елемент logical

Умовний елемент **logical** дозволяє вказати, що існування деякого факту залежить від існування іншого факту групи фактів. Умовний елемент **logical** являє собою засіб підтримки істинності, передбачений в мові CLIPS.

Приклад 6.

Розглянемо наступне правило, яке показує, що пожежники повинні використовувати кисневі маски, якщо при пожежі виділяється отруйний дим:

(defrule noxious-fumes-presents(logical (fire)

(noxious-fumes-present))

=>

(assert (use-oxygen-masks)))

В результаті даної дії додається факт *use-oxygen-masks* є «логічно залежним» від фактів *fire* і *use-oxygen-masks*. Видалення одного з цих фактів призводить до видалення факту *use-oxygen-masks*. (У діалоговому вікні CLIPS ввести команду *(retract 1)*.

Питання для самоконтролю:

1. Яка існує різниця між умовними елементами *or* і *and*?
2. У яких випадках використовується умовний елемент *exists*?

3. У яких випадках використовується умовний елемент *forall*?

4. У яких випадках використовується умовний елемент *not*?

5. У яких випадках використовується умовний елемент *logical*?

Завдання для самостійного виконання:

Скласти програму в CLIPS, яка при попаданні в систему фактів про групи крові донора (dtype) і пацієнта (ptype) видає інформацію про можливість переливання.

Правила переливання:

1. Кров групи О (dtypeO) може бути перелита тільки пацієнтові з групою кровіО (ptypeO).

2. Кров групи А (dtypeA) може бути перелита пацієнту з групою крові А (ptypeA) або групою крові О (ptypeO).

3. Кров групи В (dtypeB) може бути перелита пацієнту з групою крові В (ptypeB) або О (ptypeO).

4. Кров групи АВ (dtypeAB) може бути перелита пацієнту з групою крові АВ(ptypeAB), А (ptypeA), В (ptypeB) і О (ptypeO).

Форма звіту: Представити набір правил в середовищі CLIPS у вікні Facts. Виконати перевірку працездатності правил шляхом додавання в систему фактів про групи крові пацієнта і донора.

2.2.2 Лабораторна робота №15. Формування вхідних даних для розробки експертної системи

МЕТА: навчити формувати вхідні дані для створення ЕС.

ЗАДАЧІ: ознайомити студентів з процедурою формування вхідних даних, бази знань експертної системи, ознайомити з формуванням зумовлених фактів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Вхідні дані — знання, які задаються до початку роботи експертної системи. У середовищі CLIPS вхідні дані представляють собою набір визначених фактіві правил.

Для створення зумовлених фактів використовується конструктор *deffacts*

Приклад. (deffacts startup

(refrigerator light on) (refrigerator door open)

)

Перевірити роботу конструктора deffacts можна скориставшись діалогом **Watch Options**. Для цього виберіть пункт **Watch** меню **Execution** або використайте комбінацію клавіш <Ctrl>+<W>. У діалоговому вікні **Watch Options** увімкніть режим перегляду зміни списку фактів, поставивши галочку в полі **Facts**. Під час запуску і після виконання команди *(clear)* CLIPS автоматично конструює наступні зумовлені шаблони і факти:

(deftemplate initial-fact)(deffacts initial-fact

(initial-fact))

Зумовлений факт initial-fact шаблону initial-fact надає зручний спосіб для запуску програм мовою CLIPS — правила, що не мають умовних елементів, автоматично перетворюються в правила з умовою, що перевіряє наявність факту initial-fact. Факт initial-fact можна обробляти так само, як і всі інші факти CLIPS, додавання користувачем або програмою за допомогою команди *(assert).*

Формування передумовлених фактів і правил вимагає від розробника детального аналізу предметної області та пов'язане з роботою з експертом у даній галузі.

Наприклад, необхідно створити експертну систему для допомоги автолюбителю в діагностиці несправності двигуна автомобіля.

Опитування експерта в області технічного обслуговування автомобілів дозволяє отримати наступні факти-стану двигуна:

1. двигун працює нормально,
2. двигун працює незадовільно,
3. двигун не заводиться

Також в результаті консультації з експертом можна отримати наступні причини, що пояснюють можливі несправності двигуна, тобто правила:

1. Якщо двигун працює нормально, то це означає, що він нормально обертається, система запалювання і акумулятор знаходяться в нормі і ніякого ремонту не потрібно.
2. Якщо двигун запускається, але працює ненормально, то це говорить, принаймні, про те, що акумулятор в порядку.
3. Якщо двигун не запускається, то потрібно дізнатися, чи намагається він обертатися. Якщо двигун обертається, але при цьому не заводиться, то це може говорити про наявність поганий іскри в системі запалювання. Якщо двигун навіть не намагається заводитися, то це говорить про те, що іскри немає в принципі.
4. Якщо двигун не заводиться, але обертається, потрібно перевірити наявність палива. Якщо палива немає — то, скоріше за все, для ремонту машини потрібнопросто заправитися.
5. Якщо двигун не заводиться, потрібно також перевірити, чи заряджений акумулятор, якщо ні, то його слід зарядити.
6. Якщо двигун не заводиться, і існує ймовірність поганий іскри в системі запалювання, то необхідно перевірити контакти. Контакти можуть бути в одному з трьох станів - чисті, обпалені і брудні, у разі обпалених контактів їх необхідно замінити, у випадку якщо контакти брудні, їх досить просто почистити.

7. Якщо двигун не заводиться, іскри немає і акумулятор заряджений, то потрібно перевірити котушку запалювання на електричну провідність. У разі якщо струм не проходить через котушку, то її необхідно замінити. Якщо котушка запалювання в порядку, значить необхідно замінити розподільні дроти.

8. Якщо двигун запускається, але при цьому поводиться інертно, не відразу реагує на подачу палива, то необхідно прочистити паливну систему.

9. Якщо двигун запускається, але відбуваються перебої з запалюванням, то це говорить про наявність поганий іскри в системі запалювання, для усунення даної несправності необхідно відрегулювати зазори між контактами.

10. Якщо двигун запускається і стукає, то необхідно відрегулювати запалювання.

11. Якщо двигун запускається, але не розвиває нормальної потужності, то це може говорити про обпалених або забруднених контактах

12. Крім цього також можуть бути отримані рекомендації-факти з ремонту двигуна:

1. заправити автомобіль
2. зарядити акумулятор
3. замінити свічки
4. очистити свічки
5. замінити котушку запалювання
6. очистити паливну лінію
7. відрегулювати зазор свічки
8. ремонт не потрібен
9. зверніться до професійного автомеханіка

Отримані від експерта знання є вхідними даними і вносяться в базу знань експертної системи за допомогою конструкцій *deffact* і *defrule*.

Питання для самоконтролю:

1. Що називають вхідними даними?
2. Який існує зв'язок між вхідними даними і базою знань ЕС?
3. Які конструкції використовуються в CLIPS для запису вхідних даних?

Завдання для самостійного виконання:

Сформувати набір вхідних даних для вирішення завдання, запропонованої віндивідуальному завданні:

№ з/п студента за списком групи	**Задача для розв'язання**
1	Підбір мобільного телефону
2	Підбір подарунка до дня народження
3	Підбір напрямку тренувальної програми
4	Дії під час надзвичайної ситуації у навчальному закладі
5	Діагностика несправності комп'ютера
6	Визначення оптимального раціону харчування
7	Підбір краватки
8	Діагностика ангіни у пацієнта
9	Підбір місця для проведення відпустки
10	Підбір меню для святкової вечері

Форма звіту: вхідні дані представити у вигляді набору зумовлених фактів іправил в середовищі CLIPS.

2.2.3 Лабораторна робота №16. Реалізація елементів експертної системи

МЕТА: навчити студентів створенню елементів ЕС.

ЗАДАЧІ: ознайомити студентів з процедурою формування елементів ЕС; ознайомити з процесом створення функцій, ознайомитися з конструкцією deffunction.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Раніше вивчені конструкції *defrule, assert, deftemplate* є базовим набором конструкцій для побудови експертної системи. Однак даний набір є неповними. У процесі формування фактів, які формуватимуть ядро ЕС може брати участь як програміст (створення зумовлених фактів і шаблонів) так і безпосередньо користувач.

Для формування набору фактів, одержуваних від користувача необхідно реалізувати механізм діалогу між користувачем і ЕС. Створення такого діалогу можливо за допомогою функцій користувача.

Для визначення функцій користувача служить конструктор *deffunction*, синтаксис включає в себе 5 елементів:

1. ім'я функції;
2. необов'язкові коментарі;
3. список з нуля або більше параметрів;
4. необов'язковий символ групових параметрів для вказівки того, що функціяможе мати змінне число аргументів.

Послідовність дій або виразів, які будуть виконані (обчислені) по порядку під час виконання тіла функції.

(deffunction <имя-функции> [<необяз-комментарий>]<обязательные-параметры> [<групповой-параметр>]

<действия>

)

Приклад. Розглянемо просту функцію, що виводить на екран свій єдиний аргумент:

(deffunction PrintOneArgument (?a)(printout t ?a crlf)

)

Тут *PrintOneArgument* — ім'я функції, *(? a)* — один приймається аргумент, *(printout t? a crlf)* — тіло функції, що складається з одного виклику функції printout.

Виклик цієї функції з тіла програми може виглядати так:

(PrintOneArgument "Hello world")

Функція може приймати також і груповий параметр — набір деяких значень. Наприклад:

(deffunction CountElementsInGroup ($?x)

(printout t "Argument x consist of" (length ?x) "elements" crlf)

)

Виклик такої функції може виглядати так: *(CountElementsInGroup (a,b,c,d))* Відповідь системи:

Argument x consist of 4 elements

Функція повертає значення, що дорівнює значенню, яке повернуло останню дію або обчислене вираз під час виконання функції. Якщо останню дію не повернуло ніякого результату, то виконувана функція також не поверне результату (як у наведеному вище прикладі). Якщо функція не виконує ніяких дій, то повернене значення дорівнює FALSE. У разі

виникнення помилки при виконанні чергового дії виконання функції буде перервано і повернутим значенням також буде FALSE.

Розглянемо приклад функції, яка задає користувачеві питання і повертає відповідь, введений з клавіатури:

(deffunction ask (?question)(printout t ?question)

(bind ?answer (read))

?answer

)

Тут ім'я функції — *ask*, що приймається параметр *?question*. При виконанні функції значення змінної *?question* виводиться в якості питання користувачеві, потім виконується функція *bind* зв'язує змінну *?answer* з інформацією введеної з клавіатури. Запит введення з клавіатури виконується функцією *(read)*.

Приклад виклику:

(ask "How old are you?")

Для більш зручного аналізу користувальницьких відповідей може знадобитися, щоб функція дозволяла користувачу вводити не довільний відповідь, а один з наперед заданих варіантів. Завдання можна вирішити за допомогою групового символу. Розглянемо наступну функцію, яка використовує в своєму коді функцію *ask*, розглянуту вище.

(deffunction ask-allowed (?question $?allowed)(bind ?answer (ask ?question))

(while (not (member ?answer $?allowed))do

(printout t "Reenter, please" crlf)(bind ?answer (ask ?question))

)

?answer

)

Тут груповий параметр *$?allowed* служить для завдання набору допустимих відповідей. Першим рядком тіла виконується зв'язування значення, що повертається функцією *ask*, зі змінною *?answer*. Після чого використовується циклічна структура

(while <условие> do

<оператор>

...

<оператор>

)

В цій структурі питання задається до тих пір, поки користувач не введе дозволений відповідь. Перевірка відповідності виконується функцією *(member ?X $?y)*, яка повертає TRUE в тому випадку, якщо всередині складеного поля *$?y* знайде елемент ідентичний значенню змінної *?x*. Для спрощення роботи кінцевого користувача з ЕС допустимим є формування запитань, на які потрібно відповідь «ТАК» або «НІ»

Приклад.

(deffunction ask-yes-no (?question)

(bind ?response (ask-allowed ?question yes no))(eq ?response yes)

)

Ця функція *ask-yes-no* має відомим нам параметром *?question* а також пропонує варіант для вибору відповіді (yes no). Розглядаючи в прикладі функція пов'язаназ раніше розглянутої функцією *ask-allowed*.

Приклад.

(defrule testrule(initial-fact)

=>

(if (ask-yes-no "Are you a boy?")then

(assert(you are a boy))else

(assert(you are a girl))

)

)

У наведеному прикладі користувачеві задається питання "Are you a boy?" В разі позитивної відповіді (*yes*), в систему буде додано факт *(you are a boy)*. У разі негативної відповіді (*no*), в систему буде додано факт *(you are a girl)* (рис.2.17).

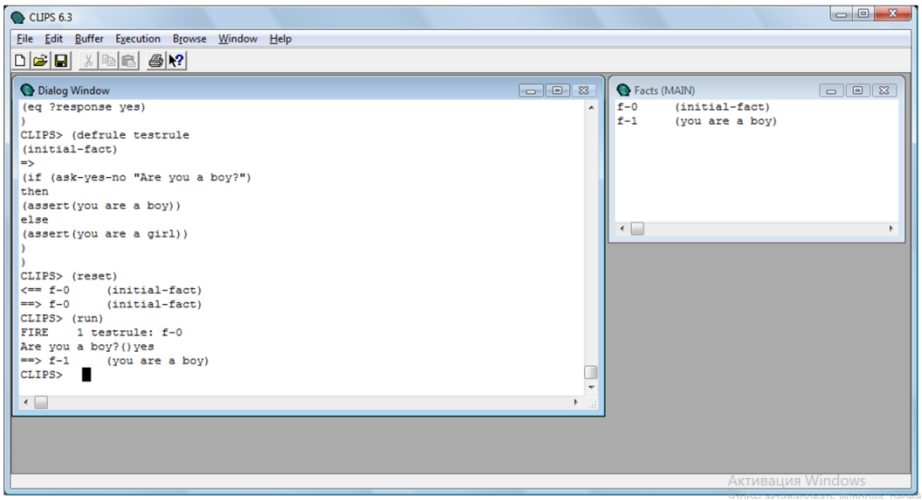

Рисунок 2.17 - Результат роботи функції «ask-yes-no»

Розглянемо просту експертну систему, яка в найпростіших випадках допоможе локалізувати проблему з двигуном автомобіля. Припустимо, що в результаті бесіди з експертом ми отримали наступні знання:

– У найпростішому випадку проблеми з двигуном можуть ділитися на дві

категорії *«Працює нестабільно»* і *«Не заводиться»*.

– Якщо двигун не заводиться, то він може або зовсім не розкручуватися, аборозкручуватися від акумулятора, але не заводитися.

– Якщо двигун не розкручується, то проблема може бути в акумуляторної батареї або в стартері двигуна.

– Якщо ж двигун не заводиться, то проблема може полягати у відсутності палива або несправності системи запалювання.

– Якщо двигун працює нестабільно, виною може слабка іскра або несправністьсвічок, а також несправність в системі подачі палива.

Запрограмуємо знання використовуючи продукційну модель, яку реалізуємо на мові CLIPS:

```
(clear)
(deffunction ask (?question $?allowed)(printout t ?question ?allowed)
(bind ?answer (read))
?answer
)
(deffunction ask-allowed (?question $?allowed)(bind ?answer (ask ?question))
(while (not (member ?answer $?allowed) )do
(printout t "Reenter, please" crlf)(bind ?answer (ask ?question))
)
?answer
)
(deffunction ask-yes-no (?question)
(bind ?response (ask-allowed ?question yes no))(eq ?response yes)
```

)

(defrule EngineState(not (work ?))

=>

(if (eq (ask-allowed "What is the problem: 1-engine does not work, 2-engine worksunstable" 1 2) 1)

then(assert(work doesnot))else

(assert(work unstable))

)

)

(defrule KindOfFail(work doesnot)

=>

(if (eq (ask-allowed "1 - Engine does not rotate, 2 - engine rotates, but not start" 1 2)1)

then

(assert(engine does-not-rotate))else

(assert(engine rotates))

)

)

(defrule CheckBatt (engine does-not-rotate)

=>

(assert(suggest "Check your battery or engine starter"))

)

(defrule EnoughFuel(engine rotates)

=>

(if (ask-yes-no "Is it enough fuel?")then

(assert(need to check ignition))else

(assert(suggest "Add Fuel"))

)

)

(defrule IgnitionCheck(not (ignition ?))

(need to check ignition)

=>

(if (ask-yes-no "Check ignition system. Is there strong spark?")then

(assert(ignition ok))else

(assert(ignition failed))

)

)

(defrule SuggestOfIgnition(ignition failed)

(engine rotates)

=>

(assert(suggest "Your engine does not start because of the faulty of ignition system"))

)

(defrule UnstableIgnition(work unstable) (not (ignition ?))

=>

(assert(need to check ignition))

)

(defrule SuggestFuel(work unstable) (ignition ok)

=>

(assert (suggest "Your engine work unstable due to unstable fuel supply"))

)

(defrule PrintSuggest(suggest ?x)

=>

(printout t ?x crlf)

)

(defrule NoSuggest (declare (salience -10))(not (suggest ?))

=>

(printout t "Sorry, there is no suggest." crlf)

)

Питання для самоконтролю:

1. Які синтаксичні вимоги до користувацької функції існують для постановки питання користувачеві?
2. Яке призначення у функції bind?
3. Яке призначення у функції ask?

Завдання для самостійного виконання:

Сформувати експертну систему для вирішення завдання, запропонованої в індивідуальному завданні. За основу взяти набір встановлених фактів і правил, сформованих за результатами лабораторної роботи №15.

Форма звіту: сформувати експертну систему в середовищі CLIPS.

2.3 Експертні навчальні системи

2.3.1 Лабораторна робота №17. Використання експертної системи для навчання

МЕТА: навчити студентів використовувати і проектувати ЕС для вирішення завдань, що виникають в процесі навчання.

ЗАДАЧІ

1. ознайомити студентів з завданнями, які вимагають експертного рішення;
2. ознайомити студентів з процедурою опису глобальних змінних *defglobal*.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Оголошення глобальної змінної

У CLIPS глобальні змінні оголошуються і отримують своє початкове значення за допомогою конструктора *defglobal*. Його синтаксис має наступний вигляд: *(defglobal <опис змінної> *)*

У свою чергу, параметр <опис змінної> має вигляд:

<опис змінної> :: =

<глобальна змінна> = <вираз>

Параметр <вираз> містить значення змінної, або вираз в круглих дужках, визначальне значення змінної. Параметр <глобальна змінна> містить ім'я глобальної змінної у форматі *symbol*: <глобальна змінна> ::=? **

Ім'я глобальної змінної завжди починається з символів? * І закінчується символом *. Так вона вказується при оголошенні. Також вона вказується при зверненні до неї. Це потрібно, щоб система не плутала змінні з командами і виразами. В одному конструкторі *defglobal* може бути оголошено кілька змінних одночасно.

Розглянемо приклад (рис.2.18):

Рисунок 2.18 - Використання конструктора defglobal для опису змінних

Одним із завдань, що вимагає експертної оцінки, в процесі навчання є процес діагностики формування в учнів необхідних компетенцій.

У сучасну шкільну програму включено вивчення інформатики в початковій школі (2-4 класи).

При переході з молодшої школи в старшу, для подальшого успішного вивчення інформатики необхідним є проведення діагностики сформованості в учнів інформаційно-комунікативної компетенції. Для вирішення даного завдання доцільно використовувати експертну систему.

Використання анкети:

У процесі проходження тесту перевіряються знання та вміння учнів, що входять до складу інформаційно-комунікативної компетенції, а саме:

– Уміння використовувати різні джерела інформації;

− Знання про те, які джерела інформації існують;

− Уміння використовувати комп'ютерні технології;

− Вміння знаходити альтернативні джерела інформації.

Анкета являє собою відкритий тест. Правильна відповідь учня збільшує його оцінку на один бал.

При підведенні підсумків тесту використовується наступна таблиця

Таблиця 2.1 - Шкала оцінювання

Кількість балів	**Рівень сформованості компетенції**
<= 5	Низький рівень
6 — 11	Нижче середнього
12 — 17	Середній рівень
18 — 22	Вище середнього
>= 23	Високій рівень

АНКЕТА

I. Знання про те, які джерела інформації існують

Мета: виявити знання учнів про джерела інформації і вміння класифікувати їхза групами.

1. Назви джерела текстової інформації:

а) розмова з одним, поради батьків, музика, рекомендації вчителя, спів птахів,сигнали машин

б) доповіді, креслення, схеми, фотографії, малюнкив) книги, енциклопедії, підручники

2. Назви джерела графічної інформації:

а) розмова з одним, поради батьків, музика, рекомендації вчителя, спів птахів,сигнали машин

б) доповіді, креслення, схеми, фотографії, малюнкив) книги, енциклопедії, підручники

3. Назви джерела звукової інформації:

а) розмова з одним, поради батьків, музика, рекомендації вчителя, спів птахів,сигнали машин

б) доповіді, креслення, схеми, фотографії, малюнкив) книги, енциклопедії, підручники

II. Вміння використовувати різні джерела інформації

Мета: перевірити вміння використовувати різні джерела інформації.

Для підготовки повідомлення за темою «Тварини України» учень використовував різні джерела інформації. Вибери, яку інформацію він отримавз даних джерел

1. З підручника учень отримав інформацію про:

а) природні умови України і зіставлення їх з життям тваринного світу;

б) цікаві факти про пристосування до життя тварин України і відображення цихфактів у оповіданні;

в) відеоматеріал для включення в доповідь;

г) схеми ланцюгів харчування тварин України.

2. З схем учень отримав інформацію про:

а) природні умови України і зіставлення їх з життям тваринного світу;

б) цікаві факти про пристосування до життя тварин України і відображення цихфактів у оповіданні

в) відеоматеріал для включення в доповідь

г) Схеми ланцюгів харчування тварин України

3. З Інтернету учень отримав інформацію про:

а) природні умови України і зіставлення їх з життям тваринного світу;

б) цікаві факти щодо пристосування до життя тварин України і відображення цихфактів у оповіданні;

в) відеоматеріал для включення в доповідь;

г) схеми ланцюгів харчування тварин України.

4. З схем учень отримав інформацію про:

а) природні умови України і зіставлення їх з життям тваринного світу.

б) цікаві факти щодо пристосування до життя тварин України і відображення цих фактів у оповіданні;

в) відеоматеріал для включення в доповідь;

г) схеми ланцюгів харчування тварин України.

III. Вміння використовувати комп'ютерні технології

Мета: перевірити знання комп'ютерних технологій.

Для підготовки наочного, демонстраційного повідомлення по темі уроку навколишнього світу тобі необхідно використовувати текстовий редактор?

Так (ні)

1. Для підготовки наочного, демонстраційного повідомлення по темі уроку навколишнього світу необхідно тобі використовувати графічний редактор?

Так (ні)

2. Для підготовки наочного, демонстраційного повідомлення по темі урокунавколишнього світу необхідно тобі використовувати адресну книгу?

Так (ні)

3. Для підготовки наочного, демонстраційного повідомлення по темі урокунавколишнього світу необхідно тобі використовувати калькулятор?

Так (ні)

IV. Вміння використовувати комп'ютерні технології

Мета: перевірити вміння використовувати комп'ютерні технології.

1. Програма «Калькулятор» використовується для: а) ведення чорнових записів

a) редагування тексту;
b) редагування графічних файлів;
c) виконання операцій з числами.

2. Програма «WordPAd» використовується для:

a) ведення чорнових записів;
b) редагування тексту;
c) редагування графічних файлів;
d) виконання операцій з числами.

3. Програма «Paint» використовується для:

a) ведення чорнових записів;
b) редагування тексту;
c) редагування графічних файлів;
d) виконання операцій з числами.

V. Уміння знайти потрібний джерело інформації не тільки в навчальнихзавданнях, але і в реальній життєвій ситуації

Мета: виявити вміння знаходити потрібний джерело інформації в реальній життєвій ситуації.

Вам необхідно відправитися на виставку екзотичних тварин, привезених в місто. Для того, щоб отримати інформацію про виставку Вам необхідно:

а) Прочитати книгу (так / ні);

б) Дізнатися адресу виставки з газети (так / ні);

в) Зателефонувати і дізнатися про час проведення виставки (так / ні);

г) Запитати сусіда у дворі (так / ні);

д) Прослухати інформацію по радіо (так / ні);

е) Знайти інформацію в підручнику (так / ні);

ж) Прочитати афішу гастролей виставки (так / ні);

з) Почути рекламу гастролей у маршрутці. (так / ні).

VI. Вміння знаходити альтернативну і додаткову інформацію

Мета: перевірити вміння учнів знаходити альтернативну і додаткову інформацію.

Вибери зайву пропозицію з тексту:

Піднявся вітер. Сонце померкло і перетворилося в мутний, ледве помітний гурток; стало темно. Пісок летів в очі, рот, вуха. Верблюди лежали, витягнувшиші, закривши ніздрі і очі. Коли вітер заспокоївся, ми побачили, що навколо насстояли високі сосни, а між ними виднілися веселі берізки.

VII. Вміння працювати із засобами мережі Інтернет

Мета: перевірити вміння працювати із засобами Інтернету

1. Сервіс WWW дозволяє:

а) спілкуватися з друзями;

б) переглядати інформацію;

в) брати участь у спілкуванні та обговоренні різних тем.

2. Сервіс «Електронна пошта» дозволяє:

а) спілкуватися з друзями;

б) переглядати інформацію;

в) брати участь у спілкуванні та обговоренні різних тем.

3. Сервіс ФОРУМИ дозволяє:

а) спілкуватися з друзями

б) переглядати інформацію

в) брати участь у спілкуванні та обговоренні різних тем.

Завдання для самостійного виконання:

Сформувати експертну систему що дозволяє провести діагностику за запропонованою вище анкетою.

Форма звіту: сформувати експертну систему в середовищі CLIPS.

2.3.2 Лабораторна робота №18. Розробка бази знань для навчальної експертної системи

МЕТА: навчити студентів отримувати експертні знання та формувати БЗ для вирішення завдань, що виникають в процесі навчання.

ЗАДАЧІ

1. ознайомити студентів з основами інженерії знань, методами вилучення та здобуття знань;

2. ознайомити студентів з особливостями формування БЗ щодо методичних особливостей навчання в ЕНС.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Особливості отримання знань від експертів. Основні поняття інженерії знань

Основні труднощі в розробці експертних систем пов'язані із проблемою добування й структурування знань. Саме ці питання досліджує наука за назвою — *інженерія знань* (knowledge engineering).

Інженерія знань — напрямок досліджень і розробок в області інтелектуальних систем, що ставить метою розробку моделей, методів і систем для одержання, структурування й формалізації знань фахівців з метою проектування баз знань.

Основні напрямки досліджень інженерії знань представлені на рис. 2.19.

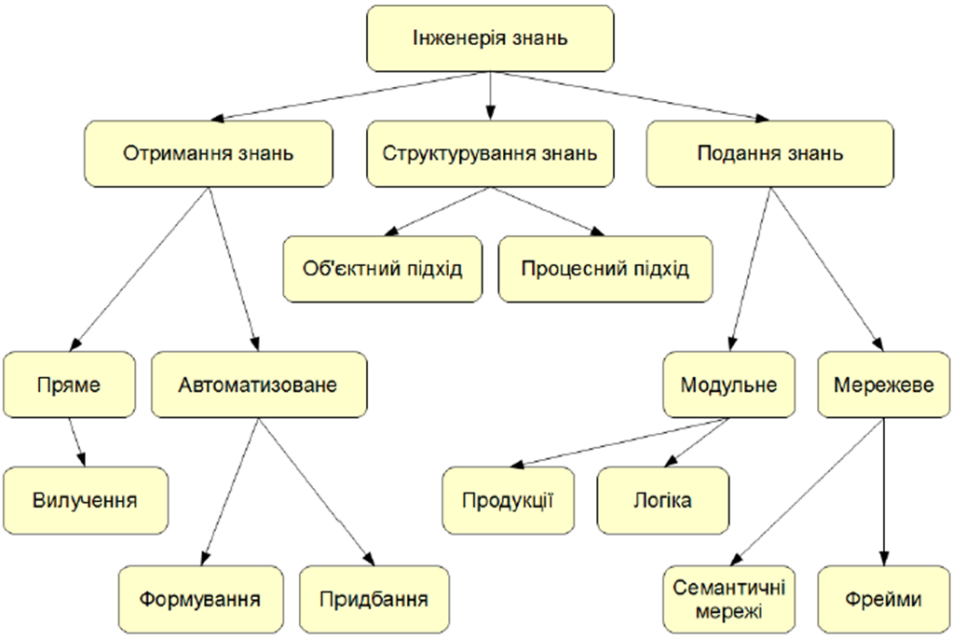

Рисунок 2.19 - Структура інженерії знань

Поле знань

Центральним поняттям на стадіях одержання й структурування є так називане *поле знань. Поле знань* — це умовний неформальний опис основних понять і взаємозв'язків між поняттями предметної області, виявлених із системи знань експерта, у вигляді графа, діаграми, таблиці або тексту.

Поле знань *z* формується на третій стадії розробки ЕС — стадії структурування. Поле знань, як перший крок від структурування до формалізації, представляє модель знань про предметну область у тім виді, у якому її зумів виразити аналітик на деякій "своїй" мові.

Узагальнено синтаксичну структуру поля знань можна представити як

$$Pz=(I ,ПРО,M),$$

де *I* — структура вхідних даних, підлягаючих обробці й інтерпретації в експертній системі;

ПРО — структура вихідних даних, тобто результату роботи системи;

M— операціональна модель предметної області, на підставі якої відбувається модифікація *I* в *O.*

Включення компонентів *I* й *O* в *Pz* обумовлене тим, що складові й структура цих інтерфейс них компонентів неявно присутні в моделі репрезентації в пам'яті експерта. Операціональна модель *M* може бути представлена як сукупність концептуальної структури *Sk , що* відбиває понятійну структуру предметної області, і функціональної структури *S f , що* моделює схему міркувань експерта *M*=(*Sk , S f*). *Sk* виступає як статична, незмінна складова *Pz* у той час як *S f* представляє динамічну, змінювану складову.

Формування *Sk* засноване на виявленні понятійної структури предметної області. Далі описаний досить універсальний алгоритм проведення концептуального аналізу на основі модифікації парадигми структурного

аналізу і побудови ієрархії понять (так називана "піраміда знань"). Приклад *Sk* й *S* представлений на рис.2.20 й рис.2.21.

В останні роки концептуальну структуру називають *онтологією предметної області*, вона включає впорядковані поняття предметної області (ПО) А и моделює основні функціональні зв'язки RA або відносини між поняттями, що утворять *Sk* . Крім онтології розуміння задачі відбиває модель або стратегія ухвалення рішення *S f* в обраній ПО. Таким чином, *S f* утворить стратегічну складову М, часто вона має форму простої таблиці рішень, як на рис. 2.20.

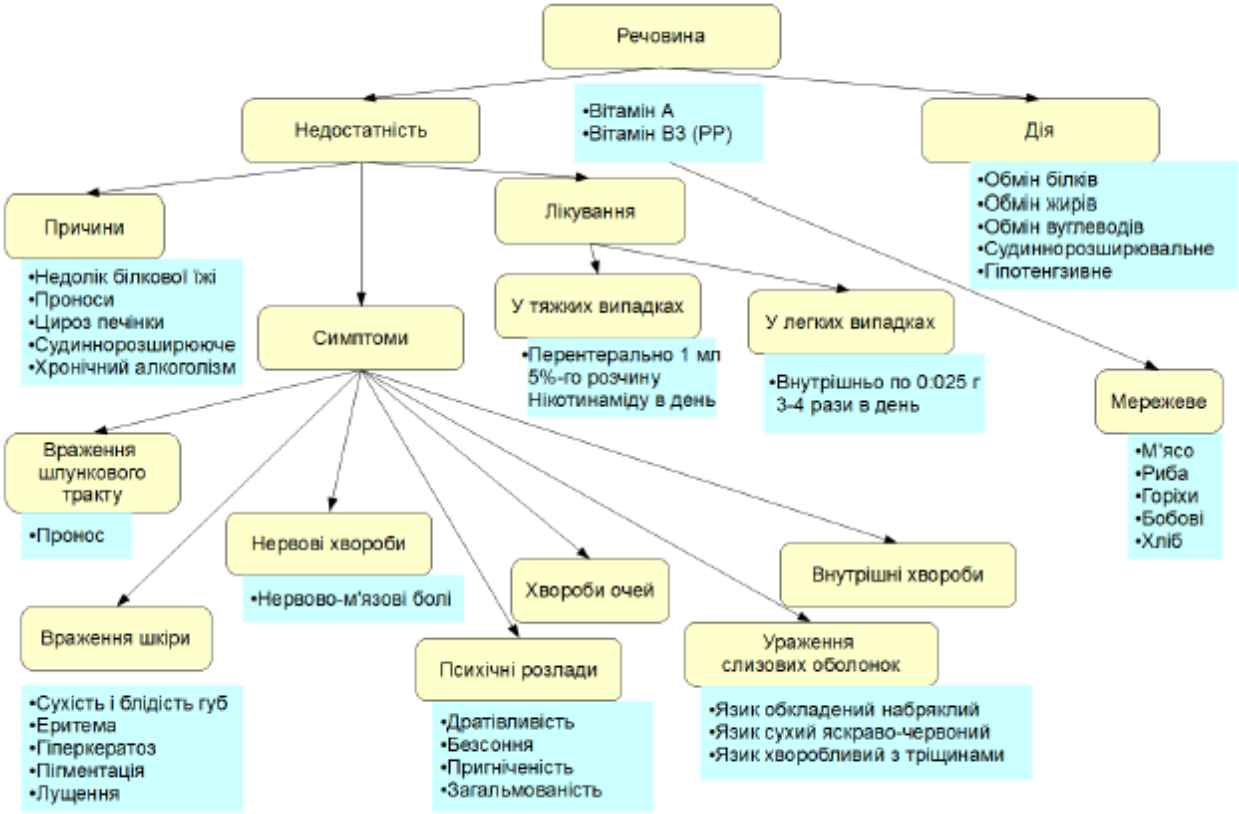

Рисунок 2.20 - Концептуальна складова поля знань

Схему, що відображає відносини між реальною дійсністю й полем знань, можна представити так, як показано на рис. 2.21.

Ознака 1	Ознака 2	Ознака 3	…Діагноз	Лікування
Ураження шкіри	Ураження очей	Нервові розлади	Брак речовини	Продукти
…			…	
Сухість губ або лущення	Підупадання зору		Вітамін А	Вершкове масло або морква
			…	
		Безсоння	Вітамін В3	М'ясо або риба
			…	
		Дратливість	Вітамін В3	М'ясо або риба
			…	

Рисунок 2.21 – Функціональна складова поля знань

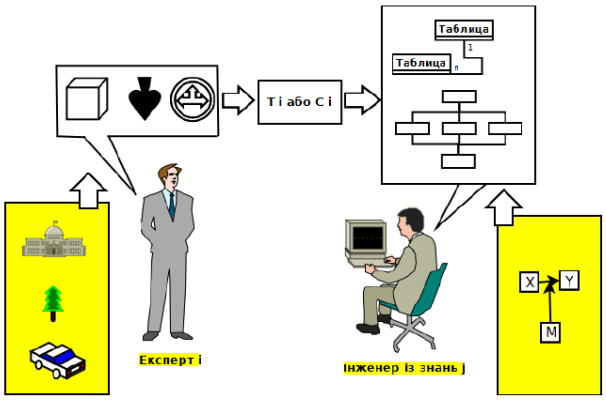

Рисунок 2.22 - "Зіпсований телефон" при формуванні поля знань

Поле *Pzg ij* — це результат, отриманий "після 4-й трансляції" (якщо говорити мовою інформатики). 1-я трансляція (*Ii*) — це сприйняття й інтерпретація дійсності *Про* предметну область *g i-м* експертом. У результаті *Ii* у пам'яті експерта утвориться модель *Mgi* як семантична репрезентація дійсності і його особистого досвіду по роботі з нею. 2-я трансляція *(Vi,}* — це вербалізація досвіду *i-го* експерта, коли він намагається пояснити свої міркування *Si* і передати свої знання *Zi*, інженерові по знаннях. У результаті *Vi*, утвориться або текст T*i*, або мовне повідомлення *Ci* 3-я трансляція *(Ij)* — це сприйняття й інтерпретація повідомлень *Ti* або *Ci j-м* інженером по знаннях. У результаті в пам'яті інженера по знаннях утвориться модель миру *Mgj*. 4-я трансляція *(Kj)* — це кодування й вербалізація моделі *Mgj* у формі поля знань *Pzg ij*.

Найбільше ця схема нагадує дитячу гру в "зіпсований телефон"; перед інженером по знаннях стоїть найскладна задача — домогтися максимальної відповідності *Mgi* й *Pzg ij* На жаль, *Pzg* не є відбиттям дійсності *Og,* тому що знання — річ сугубо авторизована, суб'єктивна. Так варто було б на кожної ЕС ставити чіткий ярлик *i — j,* тобто "база знань експерта *i* у розумінні інженера по знаннях *j".* Варто замінити, наприклад, інженера по знаннях Петрова на Сидорова, і вийде зовсім інша картина.

Приведемо приклад впливу суб'єктивних поглядів експерта на *Mgi* й *Vi.* Реальність *(Og)*: два чоловіки прибігають на вокзал за 2 хвилини до відходу поїзда. У каси - черга. В автоматичних касах вільно, але ні в того, ні в іншого немає дріб'язку. Наступний поїзд через 40 хвилин. Обоє спізнюються на важливу зустріч.

Інтерпретація 1-го експерта (I1): не можна приходити на вокзал менш чим за 10 хвилин.

Інтерпретація 2-го експерта (I2): треба завжди мати дріб'язок у кишені.

Вербалізація 1-го експерта (V1): спізнився до потрібного поїзда, тому що не розрахував час.

Вербалізація 2-го експерта (V2): спізнився, тому що на вокзалі плутанина, у касах черга.

Наступні трансляції ще більше будуть спотворювати й видозмінювати модель, але тепер уже з урахуванням суб'єктивного сприйняття інженерів по знаннях.

Таким чином, якщо вважати поле знань змістовною (семантичною) моделлю предметної області, те ця модель двічі суб'єктивна. І якщо модель *Mgi* (див. рис.4) — це усічене відображення *Og,* те саме *PZ* — лише відблиск *Mgi* через призму *Vi* й *Mgj.*

"Піраміда" знань

Ієрархічність понятійної структури свідомості підкреслюється в роботах багатьох психологів. Поле знань можна стратифікувати, тобто розглядати на різних рівнях абстракції понять. В "піраміді знань" кожен наступний рівень служить для сходження на новий щабель узагальнення й поглиблення знань у предметній області. Таким чином, можлива наявність декількох рівнів понятійної структури *Sk*. Представляється доцільним зв'язати це із глибиною професійного досвіду (наприклад, як у системі АВТАНТЕСТ) або з рівнем ієрархії в структурних сходах організації (рис. 2.22).

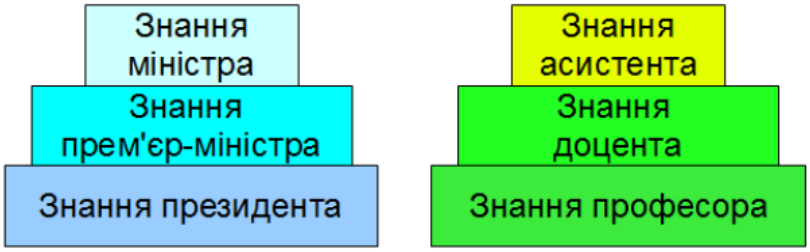

Рисунок 2.22 - Піраміди знань

Природно, що й стратегії прийняття рішень, тобто функціональні структури *Sf* на різних рівнях, будуть істотно відрізнятися.

Стратегії одержання знань

При формуванні поля знань ключовим питанням є сам процес одержання знань, коли відбувається перенос компетентності експертів на інженерів по знаннях. Для назви цього процесу в літературі по ЕС одержали поширення кілька термінів: придбання, видобуток, добування, одержання, виявлення, формування знань. В англомовній спеціальній літературі в основному використаються два терміни: acquisition (придбання) і elicitation (виявлення, добування, установлення).

Термін *"придбання" трактується* або дуже широко — тоді він включає весь процес передачі знань від експерта до бази знань ЕС, або вже як спосіб

автоматизованої побудови бази знань за допомогою діалогу експерта й спеціальної програми (при цьому структура поля знань заздалегідь заставляється в програму). В обох випадках термін "придбання" не стосується самого таїнства екстрагування структури знань із потоку інформації про предметну область. Цей процес описується поняттям *"добування "*.

Добування знань (knowledge elicitation) — це процес взаємодії аналітика із джерелом знань, у результаті якого стає явним процес міркувань фахівця при ухваленні рішення й структура його подань про предметну область.

За всіх часів більшість розробників ЕС відзначало, що процес *добування знань* залишається самим "вузьким" місцем при побудові промислових ЕС. При цьому їм доводиться практично самостійно розробляти методи добування, зіштовхуючись із наступними труднощами:

1.організаційні непогодженості;

2.невдалий метод добування, що не збігає зі структурою знань у даній області;

3.неадекватна модель (мова) для подання знань.

Можна додати до цього:

• невміння налагодити контакт із експертом;

• термінологічний різнобій;

• відсутність цілісної системи знань у результаті добування тільки "фрагментів";

• спрощення "картини миру" експерта й ін.

Процес добування знань - це тривала й трудомістка процедура, у якій інженерові по знаннях, збройному спеціальними знаннями по когнітивній

психології, системному аналізу, математичній логіці й ін., необхідно відтворити модель предметної області, якою користуються експерти для ухвалення рішення. Часто починаючі розробникі ЕС, бажаючи спростити цю процедуру, намагаються підмінити інженера по знаннях самим експертом. З багатьох причин це небажано.

По-перше, більша частина знань експерта – це результат численних нашарувань, щаблів досвіду. І часто знаючи, що з А треба В, експерт не віддає собі звіту, що ланцюжок його міркувань був набагато довше, наприклад А -> D -> З -> В, або А-> Q -> R -> В.

По-друге, як було відомо ще Платону, мислення діалогично. І тому діалог інженера по знаннях й експерта - найбільш природна форма вивчення лабіринтів пам'яті експерта, у яких зберігаються знання, які частково мають невербальний характер, тобто виражені не у формі слів, а у формі наочних образів, наприклад. І саме в процесі пояснення інженерові по знаннях експерт на ці розмиті асоціативні образи надягає чіткі словесні ярлики, тобто вербалізує знання.

По-третє, експертові сутужніше створити модель предметної області внаслідок глибини й обсягу інформації, який він володіє. Ще в ситуаційному керуванні було виявлено, що об'єкти реального миру зв'язані більш ніж 200 типами відносин (тимчасового, просторового, причинно-наслідкові, типу "частина-ціле" й ін.). Ці відносини й зв'язки предметної області утворюють складну систему, з якої виділити "кістяк" або головну структуру іноді доступніше аналітикові, що володіє до того ж системною методологією.

Термін "придбання" у межах нашого курсу залишений за автоматизованими системами прямого спілкування з експертом. Вони дійсно безпосередньо здобувають уже готові фрагменти знань у відповідності зі структурами, закладеними розробниками систем.

Більшість цих інструментальних засобів спеціально орієнтовано на конкретні ЕС з жорстко позначеною предметною областю й моделлю подання знань, тобто не є універсальними.

Придбання знань (knowledge acquisition) — процес заповнення бази знань експертом з використанням спеціалізованих програмних засобів.

Наприклад, система TEIRESIAS, що стала прародителькою всіх інструментів для придбання знань, призначена для поповнення бази знань системи MYCIN або її дочірніх галузей, побудованих на "оболонці" EMYCIN в галузі медичної діагностики з використанням продуційної моделі подання знань.

Термін *формування знань* (machine learning) традиційно закріпився за надзвичайно перспективною областю інженерії знань, що займається розробкою моделей, методів й алгоритмів навчання. Вона містить індуктивні моделі формування знань й автоматичного породження гіпотез, наприклад метод на основі навчальних вибірок, навчання за аналогією й іншими методами. Ці моделі дозволяють виявити причинно-наслідкові емпіричні залежності в базах даних з неповною інформацією, що містять структуровані числові й символьні об'єкти (часто в умовах неповноти інформації).

Формування знань (machine learning) — процес аналізу даних і виявлення схованих закономірностей з використанням спеціального математичного апарата й програмних засобів.

Традиційно до задач формування знань або машинного навчання ставляться задачі прогнозування, ідентифікації (синтезу) функцій, розшифровки мов, індуктивного висновку й синтезу з додатковою інформацією. У широкому сенсі до навчання за прикладами можна віднести й методи навчання розпізнаванню образів. Для того щоб ці методи стали

елементами технології інтелектуальних систем, необхідно вирішити ряд задач :

— забезпечити механізм сполучення незалежно створених баз даних, що мають різні схеми, з базами знань інтелектуальних систем;

— установити відповідність між набором полів бази даних і множиною елементів декларативного компонента бази знань;

— виконати перетворення результату роботи алгоритму навчання в спосіб подання, підтримуваний програмними засобами інтелектуальної системи.

Крім перерахованих існують також й інші стратегії одержання знань, наприклад, у випадку навчання на прикладах (case-based reasoning), коли джерело знань - це множина прикладів предметної області. Навчання на основі прикладів (прецедентів) включає настроювання алгоритму розпізнавання на задачу за допомогою пред'явлення прикладів, класифікація яких відома.

Навчання на прикладах тісно пов'язане з машинним навчанням. Розходження полягає в тім, що результат навчання в розглянутому тут випадку повинен бути інтерпретований у деякій моделі, у якій, можливо, уже втримуються факти й закономірності предметної області, і перетворений у спосіб подання, що допускає використання результату навчання в базі знань, для моделювання міркувань, для роботи механізму пояснення й т.д., тобто робить результат навчання елементом відповідної технології.

Наприклад, у системі INDUCE породжується несуперечливий опис деякого класу об'єктів по множинах прикладів і контрприкладів даного класу. Як мова подання використається мова змінно-значної логіки першого порядку (варіант мови багатозначної логіки першого порядку).

Останнім часом широке поширення одержали терміни data mining й knowledge discovery, що означають, по суті, той же процес формування знань

і пошук закономірностей, здійснюваний на більших вибірках даних, що звичайно перебувають у *сховищах даних* (data warehouse).

Таким чином, можна виділити три основних стратегії проведення стадії одержання знань при розробці ЕС (рис. 2.23):

— з використанням ПК при наявності підходящого програмного інструментарію — *придбання знань;*

— з використанням програм навчання при наявності репрезентативної (тобто досить представницької) вибірки прикладів прийняття рішень у предметній області й відповідних пакетах прикладних програм —- *формування знань;*

— без використання обчислювальної техніки шляхом безпосереднього контакту інженера по знаннях і джерела знань (будь те експерт, спеціальна література або інші джерела) — *добування знань.*

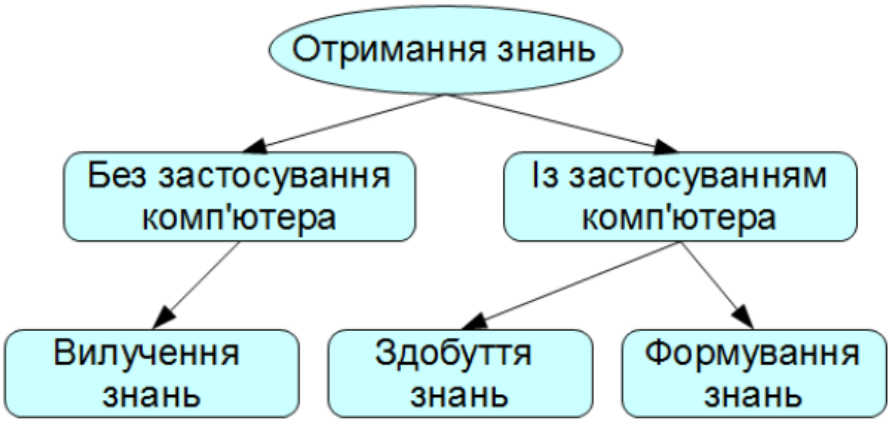

Рисунок 2.23 - Стратегії одержання знань

Питання для самоперевірки

1. Якими проблемами займається інженерія знань?
2. Що являє собою поле знань?
3. Які існують вимоги до мови представлення знань?
4. Які мови представлення знань Вам відомі?
5. Що таке мова семіотичного моделювання?

6. Який вигляд має узагальнена синтаксична структура поля знань?
7. Надайте опис трансляцій між реальністю та полем знань.
8. Які методи вилучення знань Ви знаєте?
9. В чому полягає особливість комунікативних методів вилучення знань?
10. За якими ознаками класифікуються предметні області в інженерії знань?
11. В чому полягає відмінність анкетування від отримання інтерв'ю?
12. Які підходи використовують для структурування знань?
13. В чому полягає відмінність об'єктного підходу до структурування знань?
14. У чому полягає особливість структурного підходу до структурування знань?

Завдання для самостійної роботи

1. Складіть план інтерв'ю з експертом з приводу лікування ангіни.
2. Складіть анкету для отримання знань для вирішення проблеми годування домашніх тварин.
3. Яким чином підготуватися до прослуховування лекції, як засобу отримання знань для створення БЗ?
4. Складіть план проведення круглого столу з питання «Впровадження Болонського процесу»
5. Складіть план проведення рольової гри «Призначення стипендії».
6. Складіть план діалогу з експертом для вилучення знань з приводу організації модульного процесу.
7. Як підготуватися до спостережень за роботою експерта з приводу підбору краватки до костюма?
8. Наведіть приклади формулювання відкритих питань.
9. Наведіть приклади формулювань закритих питань.
10. Наведіть приклади формулювань особистих питань.

11. Наведіть приклади формулювань безособових питань.

12. Наведіть приклади формулювань питань з використанням наочного матеріалу.

13. Наведіть приклади формулювання прямих запитань.

14. Наведіть приклади формулювання непрямих питань.

15. Складіть анкету для отримання знань для вирішення проблеми класифікації тварин Складіть план інтерв'ю з експертом з приводу лікування застуди.

16. Яким чином підготуватися до прослуховування лекції для вилучення знань з приводу вирощування кімнатних квітів?

17. Складіть план проведення круглого столу з питання «Альтернативні джерела енергії».

18. Складіть план проведення рольової гри «Визначення об'єктів капітального будівництва».

19. Складіть план діалогу з експертом для вилучення знань з приводу вибору подарунка для дитини.

20. Як підготуватися до спостережень за роботою експерта з приводу діагностики збою комп'ютера.

21. Наведіть приклади формулювання відкритих питань.

22. Наведіть приклади формулювань закритих питань.

23. Наведіть приклади формулювань питань з використанням наочного матеріалу.

24. Наведіть приклади формулювання непрямих питань.

Теми для самостійного опрацювання

1. Визначення архітектури та особливості експертних систем різного призначення; дослідження особливостей створення експертних систем навчального призначення; особливостей формування баз знань; добір прикладів баз знань з методичної складової.

2. Основні етапи розробки експертних систем. Визначення основних етапів розробки експертних систем за різними технологіями, їх порівняння та аналіз.

3. Експертні навчальні системи. Розробка основних компонентів експертної системи навчального характеру на основі застосування бази знань, що реалізовано в середовищі CLIPS.

Література

Основна література

1. Мазурок Т.Л., Черних В.В., «Експертні системи»: навчальний посібник для бакалаврів з галузі знань 0403 «Системні науки та кібернетика» за напрямом підготовки 6.040302 «Інформатика*». Одеса: ПНПУ ім. К.Д. Ушинського, 2015. 100 с.
2. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. СПб.: Питер, 2001. 384 с.
3. Пасічник В.В. Організація баз даних та знань: підручник для студентів ВНЗ. Київ: BHV, 2006. 384 с.
4. Руденко О.Г. Штучні нейронні мережі: навчальний посібник. Харків: Компанія СМІТ, 2006. 404 с.
5. Крапивный Ю.Н. Методические указания по изучению языков Prolog и Lisp для решения задач искусственного интеллекта. Одесса: ОНУ им. И.И. Мечникова, 2005. 66 с.

Допоміжна

1. Іванченко Г.Ф. Системи штучного інтелекту: навч. посіб. Київ: КНЕУ, 2011. 382 с.
2. Братко И. Программирование на языке ПРОЛОГ для искусственного интеллекта, Москва: Мир, 1990. 560 с.
3. Бондарев В.Н., Аде Ф.Г. Искусственный интеллект. Севастополь: Изд-во СевНТУ, 2002. 615 с.
4. Частиков А.П., Гаврилова Т.А., Белов Д.Л. Разработка экспертных систем. Среда CLIPS. Санкт-Петербург: БХВ-Петербург, 2003. 608 с.
5. Хант Э. Искусственный интеллект. Москва: Мир, 1978. 558 с.

6. Рассел С., Норвиг П. Искусственный интеллект. Современный поход. Москва: Изд.дом «Вильямс», 2007. 1408 с.

7. Пупков К.А., Коньков В.Г. Интеллектуальные системы. Москва: МГТУ им. Н.Э. Баумана, 2003. 348 с.

8. Башмаков А.И., Башмаков И.А. Интеллектуальные информационные технологи. Москва: Изд-во МГТУ им. Н.Э. Баумана, 2005. 304 с.

9. Гладун В.П. Партнёрство с компьютером. Київ: Port-Royal, 2000. 128 с.

10. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечёткие системы. Москва: Горячая линия - Телеком, 2006. 452 с.

11. Леоненков А.В. Нечёткое моделирование в среде Matlab и fuzzyTECH. Санкт-Петербург: БХВ - Петербург, 2003. 736 с.

12. Андрейчиков А.В., Андрейчикова О.Н. Интеллектуальные информационные системы. Москва: Финансы и статистика, 2004. 424 с.

Інформаційні ресурси

1. Електронний каталог бібліотеки ПНПУ ім. К.Д. Ушинського. URL: https://unilib.library.pdpu.edu.ua/search.php (дата звернення 27.03.2021).

2. Портал штучного інтелекту. URL: http://www.aiportal.ua/articles/expert-systems/1/ (дата звернення 27.03.2021).

3. Приклад експертної системи з вибору автомобіля. URL: http://digidrive.ua/ (дата звернення 27.03.2021).

4. Самовчитель з експертних систем. URL: http://sapr-mgsu.ua /biblio/ex-syst/ (дата звернення 27.03.2021).