МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

ІПІРОТІ Віктор Олександрович

«Система підтримки прийняття рішень онколога на основі аналізу імуногістохімічних зображень / Oncologist decision support system based on analysis of immunohistochemical images»

спеціальність: 123 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерна інженерія
Кваліфікаційна робота

Виконав студент групи КІзм-21
ІПІРОТІ Віктор Олександрович
_____

Науковий керівник:
д.т.н., Березький О.М.
_____

Кваліфікаційну роботу допущено
до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____О. М. Березький

Тернопіль – 2021

ЗМІСТ

ВСТУП

Актуальність теми. Останні десятиріччя розвитку медицини характеризуються тим, що завдяки сучасним методам експерименту багатостороння інформація, яку отримують медики при вивченні життєвих процесів, збільшується за об’ємом [1, 2]. Тому прагнення до її автоматичного аналізу та обробки є абсолютно виправданим. При побудові системи діагностики в медицині фахівці стикаються з введенням в машину багатьох параметрів. Все це значною мірою полегшує працю дослідника, об’єктивізує одержувані дані і багато того, про що робилися висновки на основі лише досвіду та інтуїції, набуває закінчену кількісну оцінку.

Актуальною задачею є автоматичний аналіз в мікроскопічних дослідженнях. Робота за мікроскопом дослідників клітин і тканин поступово доповнилася машинним аналізом, який приносить швидкодію об'єктивізації, а найголовніше – постановку і вирішення абсолютно нових завдань, пов'язаних з можливістю дослідження багатокомпонентних систем.

Зростаючий обсяг цитологічних і гістологічних досліджень у науковій роботі та медичній практиці позначив дві важливі і поки не вирішені до кінця проблеми: необхідність постійного збільшення персоналу і витрат на величезну кількість цитологічних, особливо гематологічних досліджень, а також необхідність об'єктивізації морфологічних досліджень, зокрема однакової оцінки різних ступенів дисплазії, малігнізації тканин, вираженості структурної перебудови та інших патологічних змін, з якими пов'язаний характер лікувальних втручань. Велику перспективу для вирішення вказаних проблем відкриває розвиток кількісних аспектів нормальної і патологічної морфології, а також автоматизація досліджень.

Мета і завдання дослідження. Метою роботи є формалізація бази знань аналізу імуногістохімічних зображень на основі фреймової моделі та її програмна реалізація.

Об’єкт дослідження – процес представлення знань.

Предмет дослідження – моделі представлення знань.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

– проаналізувати різні типи знань;

– здійснити аналіз моделей представлення знань;

– проаналізувати імуногістохімічні зображення;

– провести аналіз алгоритмів побудови фреймової моделі представлення знань;

– здійснити програмну реалізацію фреймової моделі.

Методи дослідження базуються на використанні теорії інженерії знань, методів штучного інтелекту.

Наукова новизна одержаних результатів. Формалізовано базу знань аналізу імуногістохімічних зображень на основі фреймової моделі.

Практичне значення отриманих результатів. На основі формалізації бази знань здійснено програмну реалізацію фреймової моделі.

Публікації результатів досліджень. За результатами досліджень опубліковані двоє тез доповідей [3, 4] V-ої науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп’ютерні системи та мережі», яка відбулася 2 грудня 2021 р. в м. Тернопіль в Західноукраїнському національному університеті:

Кваліфікаційна робота складається із трьох розділів, висновків, списку використаної літератури та додатків [9].

У першому розділі проаналізовано імуногістохімічні зображення, різні типи знань, моделі представлення знань.

У другому розділі приведено структуру даних фрейму, властивості фреймів, системи фреймів, універсальні мови подання знань фреймами, приклади систем, визначено конфігурацію системи.

Третій розділ присвячений програмній реалізації модуля фреймової моделі представлення, приведено структуру програмної системи, тестування та верифікацію програмної системи.

У додатках приведені лістинги розроблених програм, світлокопії публікацій та довідка про використання результатів кваліфікаційної роботи.

1 АНАЛІЗ ІМУНОГІСТОХІМІЧНИХ ЗОБРАЖЕНЬ ТА МОДЕЛІ ПРЕДСТАВЛЕННЯ ЗНАНЬ

1.1 Різні типи знань

У сфері ШІ метою досліджень є створення таких систем, які, з одного боку, можуть використати багато знань, переданих їм фахівцями, а з другого – здатні розпочинати діалог та пояснювати свої власні висновки. Це вимагає наявності великої за обсягом і добре структурованої бази знань, суворого розмежування між різними рівнями знань, наявності множини представлень для правил, схем предикатів чи прототипів і чіткого процесу обміну інформацією між різними джерелами.

Подання знань є однією з найбільш важливих проблем, з якою доводиться зіштовхуватися при обробці знань або при побудові систем, заснованих на знаннях. Від подання знань в кінцевому підсумку залежать характеристики системи. При правильному виборі способу представлення знань можна уникнути непотрібного ускладнення системи і вирішити безліч питань. Однак вибір оптимального способу подання знань багато в чому залежить від характеру і складності розв'язуваних завдань. Наприклад, різними є подання знань для випадку, коли об'єктом є діагностика, від подання знань для випадку, коли об'єктом є проектування. Вибір оптимального представлення знань однаково важливий як для малих, так і великих задач.

Для вирішення різноманітних задач людина розуміє мову і зображення і використовує знання в конкретній предметній області. Для того щоб аналогічну роботу зробив комп'ютер ці знання необхідно представити зрозумілій для нього формі і скласти потрібну програму. В минулому такі задачі розв’язувалися на мовах типу Фортран, Паскаль та ін. Ці знання поміщалися в прикладну програму, з якою і складали єдине ціле. Проте не було зрозуміло, яким чином

використовуються знання, яку роль вони виконують, утруднювалася розробка і модифікація програм.

Така проблема зникла у системах, заснованих на концепціях штучного інтелекту та інженерії знань, які інакше називалися системами, заснованими на знаннях. Знання в них представлені в конкретній формі, а наявна база знань дозволяє їх легко визначати, модифікувати і поповнювати. Вирішуються задачі автономним механізмом логічних висновків, які зроблені на підставі знань, що зберігаються в базі знань. Визначення моделі подання знань накладає обмеження на вибір відповідного механізму логічних висновків. При проектуванні систем, заснованих на знаннях необхідно аналізувати вид знань.

З точки зору ШІ та інженерії знань визначення знань необхідно пов'язати з логічним висновком. Пропонується [6] дати визначення знань: «знання – це формалізована інформація, на яку посилаються або використовують в процесі логічного висновку». Процес рішення задач за допомогою найпростішої моделі показаний на рисунку 1.1. У даному випадку знання – це інформація, на яку посилаються, коли роблять різні висновки на підставі наявних даних за допомогою логічних висновків. Якщо ця робота виконується програмним шляхом, то знання – це обов'язково інформація, представлена в певній формі.

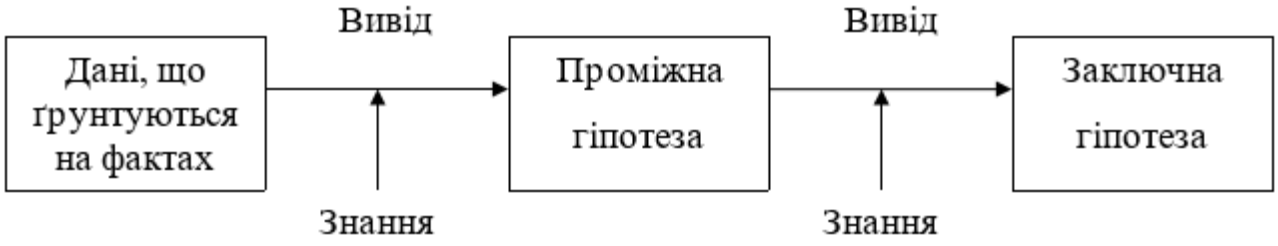


Рисунок 1.1 – Зв'язок між знаннями і виводом при вирішенні інтелектуальної проблеми

Знання з погляду вирішення проблем у певній галузі діляться на дві широкі категорії – факти та евристики. Першими є обставини добре відомі в цій галузі. Ці знання іноді називають текстовими, що передбачає їхнє достатнє висвітлення у спеціальній літературі чи посібниках. Друга категорія знань

заснована на власному досвіді спеціаліста в цій галузі, накопиченому за багато років практики. У експертних системах ця категорія знань грає на вирішальній ролі підвищення ефективності систем.

Знання можна поділити на факти (фактичні знання) та правила (знання для прийняття рішення). Такі знання як «А є А» є фактами. Вони специфічні для баз даних та мережевих моделей. Знання форми «ЯКЩО – ТО» означає правила. Крім них існує так зване метазнання (знання вище знання), тобто. знання про засоби використання знань і знання про властивості знання. Метазнання необхідні для управління базою знань, висновку, ідентифікації, навчання тощо.

Приведемо іншу класифікацію типів знань [7]. Виділяють вісім основних типів знань за такими ознаками.

1) Базові елементи, об'єкти реального світу. Вони зв’язані з безпосереднім сприйняттям, не вимагають обмірковування, і додаються до нашої бази фактів в такому вигляді, в якому вони отримані.

2) Твердження й визначення. Вони ґрунтуються на базових елементах і наперед розглядаються як достовірні.

3) Концепції. Вони представляють собою перегрупування чи узагальнення базових об'єктів за певними прийомами (з урахуванням прикладів, контр прикладів, окремі випадки, більш спільних або аналогічних концепцій тощо).

4) Відношення. Це елементарні властивості базових елементів, і відношення між концепціями.

5) Теореми. Вони є частковим випадком продукційних правил із достатньо визначеними властивостями. Теореми не представляють ніякої користі без експертних правил їх використання. Теореми в експертних системах є головною відмінністю систем управління від класичних баз даних (СУБД), у яких теореми відсутні, або програмуються. Процедура модифікації теорем чи додавання нових є дуже трудомісткою, хоч і необхідною, оскільки потрібно

забезпечити хороше структуроване управління базою даних, і оптимізувати отримання відповідей [8].

6) Алгоритми рішень. Вони необхідні для розв’язку задач. В усіх випадках вони зв'язані зі знаннями особливого типу, оскільки обумовлена ними послідовність дій виявляється оформлена у суворо необхідному порядку на відміну від інших типів знань, де елементи інформації можуть розташовуватися без зв'язку один з одним. Очевидно, що дуже важко працювати з довгими процедурами, які складаються з частин різних дій.

7) Стратегії і евристика. Це вроджені або набуті правила поведінки, що дозволяють в даній конкретній ситуації приймати рішення про необхідні дії.

8) Метазнання. Це знання про знання. Крім цього метазнання визначають ступінь довіри до певного виду знань. Крім того, сюди ж таки відносяться питання кожного типу знань і вказівок, коли і як є підстави у використанні.

1.2 Моделі представлення знань

Для різних предметних областей існують десятки моделей (чи мов) представлення знань. Всі вони можуть бути поділені на наступні класи:

- продукційні моделі;
- семантичні мережі;
- фрейми;
- формальні логічні моделі.

Розглянемо перший клас. Продукційна модель – це модель, заснована на правилах, що дозволяє представити знання у виді пропозицій типу “Якщо (умова), то (дія)”.

Перша частина пропозиції – “умова” (антецедент). Під “умовою” розуміється якась пропозиція-зразок, по якій здійснюється пошук у базі знань.

Друга частина – “дія” (консеквент). “Дії” – це дії, виконувані при успішному результаті пошуку (вони можуть бути проміжними, або такими, що виступають далі як умови, або бути цільовими, завершати роботу системи).

Висновки на такій базі знань найчастіше приймають від даних до пошуку мети (прямо) чи від мети для її підтвердження – до даних (у зворотному порядку). Дані – це вихідні факти, що зберігаються в базі фактів, на підставі яких запускається машина чи інтерпретатор правил, що перебирає правила з продукційної бази знань.

Застосування продукційних моделей різноманітне, але найкраще їх застосовувати в промислових експертних системах. Продукційна модель приваблює розроблювачів своєю наочністю, високою модульністю, гнучкістю, легкістю внесення доповнень і змін, простотою отримання логічного висновку.

Існує велика кількість програмних засобів, що реалізують продукційний підхід (мова OPS 5; “оболонки” чи “порожні” ЕС – EXSYS Professional, Kappa, ЕКСПЕРТ; ЭКО, інструментальні системи ПІЕС [9, 10] і СПЕІС [11] і ін.), а також промислових ЕС на його основі (наприклад, ЕС, створених засобами G2 [12]) і ін.

До другого класу моделей відносять семантичні мережі. Термін семантична означає “значеннєва”, а сама семантика – це наука, що встановлює відношення між символами й об'єктами, що вони позначають, тобто наука, що визначає зміст знаків.

Семантичну мережу можна представляти у вигляді орієнтованого графа, вершини якого, як правило, позначають поняття предметної області, а дуги – відношення між ними.

Як поняття звичайно виступають абстрактні чи конкретні об'єкти, а відношеннями можуть бути зв'язки типу: “це”, “має частиною”, “належить”, “любить”. Характерною рисою семантичних мереж є обов'язкова наявність трьох типів відношень:

– клас – елемент класу (машина – Шкода);
– властивість – значення (колір – синій);

– приклад елемента класу (Шкода – Рапід).

Запропоновано кілька класифікацій семантичних мереж, зв'язаних з типами відношень між поняттями [9].

За кількістю типів відношень:

– Однорідні (мають єдиний тип відношень).

– Неоднорідні (мають різні типи відношень).

За типами відношень:

– якщо відношення зв'язують два об'єкти то – бінарні;

– якщо відношення зв'язують більше двох об'єктів то – N-арні.

В семантичних мережах часто-густо використовуються наступні відношення:

– відношення типу “частина – ціле” (“елемент – множина”, “клас – підклас” і т.п.);

– функціональні відношення (визначені безперечно дієсловами “робить”, “впливає” і т. п.);

– кількісні (менше, не менше, більше, не більше, дорівнює і т. п.);

– просторові (над, під, перед, позаду, далеко від, за, ближче і т. п.);

– тимчасові (не раніше, раніше, не пізніше, пізніше, протягом і т. п.);

– атрибутивні зв'язки (мати атрибут, мати вагу);

– логічні відношення (І, АБО, НЕ);

– лінгвістичні зв'язки та ін.

Пошук рішення в базі знань типу семантичної мережі полягає у пошуку фрагмента мережі (підмережі), що відображає поставлений запит до бази.

На рисунку 1.2 зображена семантична мережа. Як вершини отут виступають поняття “людина”, “Іванов”, “Волга”, “машина”, “вид транспорту” і “двигун”.

Запропонував семаничні мережі як моделі представлення знань американський психолог Куілліан. На думку [13] основною перевагою моделі є те, що вона більше ніж інші відповідає сучасним представленням про

організацію довгострокової пам'яті людини, а недоліком – складність організації процедури пошуку висновку.

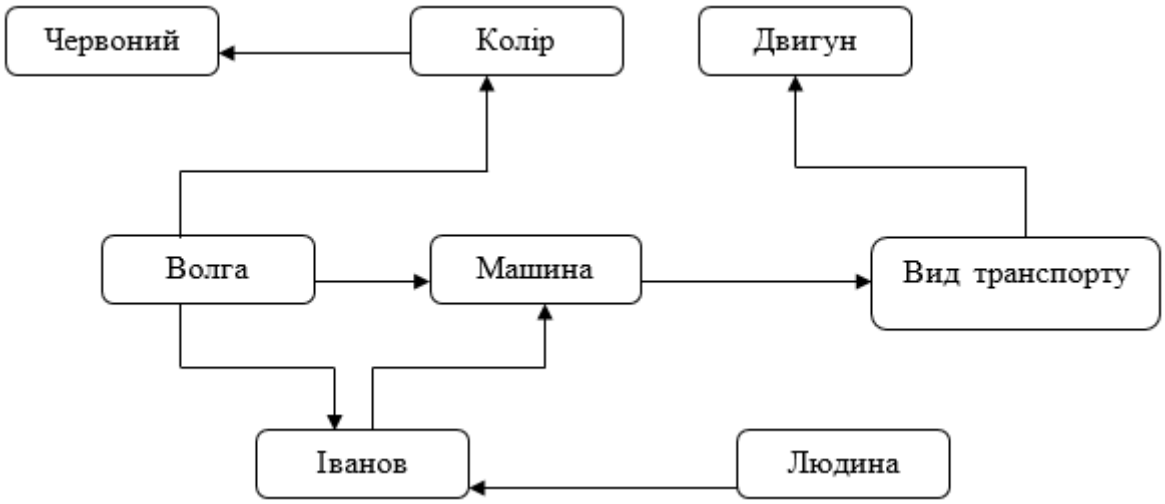


Рисунок 1.2 – Семантична мережа

Для реалізації семантичних мереж існують спеціальні мережні мови, наприклад NET [14], мова реалізації систем SIMER+MIR [15, 16] і ін. Широко відомі експертні системи, що використовують семантичні мережі як мову представлення знань – PROSPECTOR, CASNET, TORUS [17-19].

Третій клас моделей представлення знань – фрейми. Термін фрейм в буквальному перекладі з англійського означає “каркас”, “рамка”. Його у 70-их роках 20 ст. запропонував Марвін Мінський [20], який на той час був одним з піонерів ШІ. Він розробив структуру знань для сприймання просторових сцен.

Фрейм – це одиниця представлення знань, яка запам’яталася в минулому, але може бути змінена в поточній ситуації.

Фрейм – абстрактний образ для представлень деякого стереотипу сприйняття [9].

У книзі [9] для розуміння поняття абстрактного образу приведено досить зрозумілий приклад проголошення вголос слова “кімната”. Це слово породжує в слухачів образ кімнати: “житлове приміщення з чотирма стінами, підлогою, стелею, вікнами і дверима, площею 6-20 м^2”. Забрати з цього опису нічого не можна (наприклад, якщо заберемо вікна, то ми одержимо вже не кімнату, а комору). Але в цьому описі відсутня інформація про кількість вікон, розмір

дверей, висоту кімнати, колір стін, покриття підлоги й ін.

Ці незаповнені значення деяких атрибутів (“дірки”) в теорії фреймів носять назву “слоти”, а такий образ кімнати називається фреймом кімнати. Фреймом теж називається і формалізована модель для відбиття образу.

Існують системи відліку або прототипи, що зберігаються в базі знань, та фрейми екземплярів, що створюються для представлення реальних ситуацій на основі вхідних даних. Фреймова модель досить універсальна, оскільки дозволяє відобразити все різноманіття знань про світ через:

– фрейми, що використовуються для позначення об'єктів та понять (позичка, застава, рахунок-фактура);

– фремові ролі (менеджер, учитель, клієнт);

– сценарні фрейми (банкрутство, загальні збори, ювілей, огляд);

– ситуаційні фрейми (тривога, несправність, режим роботи пристрою) та ін.

Структуру фрейму можна представити у вигляді списку атрибутів:

(ІМ'Я ФРЕЙМУ:

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

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

.....

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

Той же запис можна представити у виді таблиці (таблиця 1.1), доповнивши її двома стовпцями.

Таблиця 1.1 – Структура фрейму

Ім’я фрейма			
Ім’я слота	Значення слота	Метод отримання значення	Приєднана процедура

У таблиці додаткові стовпці призначені для опису того, як інтервал отримує своє значення і як його можна приєднати до конкретного інтервалу спеціальних процедур, дозволених теоретично фреймі. Значення розташування може бути ім'ям іншого фрейму. У такий спосіб формуються мережі фреймів. Слот може отримати значення у фреймі екземпляра декількома способами:

- за замовчуванням із основи вибірки (значення за замовчуванням);
- шляхом успадкування властивостей фрейму, зазначеного в слоті АКО;
- за формулою, вказаною у слоті;
- через процедуру, що додається;
- явний діалог із користувачем;
- із бази даних.

Найважливіша властивість теорії фреймів запозичена з теорії семантичних мереж - те, що називається успадкуванням властивостей. У фреймах та семантичних мережах успадкування відбувається через A-Kind-Of=this. Слот АКО вказує на фрейм на вищому рівні ієрархії, звідки значення аналогічних слотів успадковуються, тобто переносяться.

Основна перевага фреймів як моделі представлення знань у тому, що вони відбивають концептуальну основу організації пам'яті людини [21], і навіть її гнучкість і ясність.

Спеціальні мови представлення знань у мережах фреймів FRL (Frame Representation Language) [22], KRL (Knowledge Representation Language) [23], фреймова “оболонка” Kappa [24] і інші програмні засоби дозволяють ефективно

будувати промислові ЕС. Широко відомі такі фрейм-орієнтовані експертні системи, як ANALYST, МОДИС, TRISTAN, ALTERID [11].

Програма із фреймовою моделлю представлення знань:

```
% Фрейм algoritm - визначення типового алгоритму
algoritm(a_kind_of, sorting).
algoritm(sorting_method, internal_sorting).

% Фрейм bubble_sort - представник типового алгоритму;
bubble_sort(a_kind_of, algoritm).
bubble_sort(condition, stable).

%Фрейм merge_sort - представник нетипового алгоритму;
merge_sort(a_kind_of, algoritm).
merge_sort(sorting_method, external_sorting).

% Фрейм insertion_sort - представник типового
алгоритму
insertion(instance_of, insertion_sort).
insertion(mount, 200).

% Фрейм merge - представник merge_sort алгоритму
merge(instance_of, merge_sort).
merge(mount, 500).

% Фрейм bubble - представник bubble_sort алгоритму
bubble(instance_of, bubble_sort).
bubble(mount, 100).
```

```
    % Фрейм sorting
    sorting( relative_size,
            execute(relative_size( Object, Value),
Object, Value) ).
    value( Frame, Slot, Value) :-
      Query =.. [ Frame, Slot, Value],
      call( Query), !.  %Одержання значення за допом.
безпосередньої вибірки
    value( Frame, Slot, Value) :-
      parent( Frame, ParentFrame),        % Фрейм
батьківського типу
      value( ParentFrame, Slot, Value).
    parent( Frame, ParentFrame) :-
      (Query =.. [ Frame, a_kind_of, ParentFrame]
      ;
      Query =.. [ Frame, instance_of, ParentFrame]),
      call( Query).
```

До четвертого класу представлення знань віднесемо формальні логічні моделі.

Традиційно в представленні знань виділяють формальні логічні моделі, засновані на класичному численні предикатів 1-го порядку, коли предметна чи область задача описується у виді набору аксіом. 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 використовуються різні її модифікації і розширення.

Прикладом логічних моделей подання фактів за допомогою предикатів першого порядку, що носять назву атомарної формули можуть бути:

ЛЮБОВ (Микола, Оксана): Микола любить Оксану

СТОЛИЦЯ (Відень): Відень – столиця

Як відомо, класична логіка типу логіки предикатів першого порядку є формальна система, що складається з множини термів та операцій, множини правил конструювання правильно побудованих виразів (синтаксису), системи

аксіом і множини правил виводу. Вона дає різні засоби формалізації та аналізу правильності дедуктивних міркувань. Мова класичної логіки є основою для вираження декларативних знань, де міркування визначається як операція доведення загальнозначущості (суперечливості) логічного твердження [25]. Так, квантори і логічні операції логіки предикатів першого порядку відіграють роль:

– роль ∃-квантифікації (вираження, що дещо має певну властивість, не вказуючи що саме);

– роль ∀-квантифікації (вираження, що кожен елемент якогось класу має певну властивість, без вказівки, що представляє з себе кожен такий елемент);

– роль диз'юнкції (вираження, що хоча б одне з двох тверджень істинне, не кажучи, яке саме);

– роль кон'юнкції (вираження, що кожне з двох тверджень істинне);

– роль заперечення (явно сказати, що щось хибне);

– роль рівності (ствердження або залишення невстановленим того факту, що два різних вирази означають один і той же об'єкт).

Ці парадигми корисні і часом необхідні при вирішенні багатьох проблем штучного інтелекту. Велика роль формальної логіки також в семантичному аналізі знань і обґрунтуванні висновків. Представити знання – це значить виразити в деякому формалізмі наявний у нас образ світу. Відповідність між світом і його поданням встановлюється семантичним аналізом. Такий аналіз має на меті визначити об'єкти представлення і уточнити образ світу, який визначається представленням. Отже, воно повинно дозволити здійснювати аналіз істинності висловлювань про світ. Інакше кажучи, для правильного подання треба, щоб воно могло бути предметом аналізу, що використовує інформацію з цього представлення для виявлення того, що властиво світу, а що ні. З цієї точки зору обґрунтований висновок або дедукція «підтверджуються» баченням світу, який визначається семантичним аналізом подання.

Семантичний аналіз представленого в деякому формалізмі знання повинен дозволяти визначити, що в цьому уявному світі істина, а що – брехня. Навіть якщо аналіз наділений іншими аспектами, подібна операція належить за визначенням до компетенції формальної логіки і робить особливо корисним звернення до теорії моделей.

1.3 Аналіз імуногістохімічних зображень раку молочної залози

Рак молочної залози (РМЗ) – найбільш поширена онкологічна хвороба серед жінок, яка становить близько 20% в структурі онкологічної захворюваності жіночого населення. РМЗ – єдина пухлина, від якої однаково недомагають жінки всього світу.

Рівень захворюваності на РМЗ має тенденцію до постійного збільшення в усіх країнах – членах ВООЗ (Всесвітньої організації охорони здоров'я). З урахуванням тренду поступового старіння населення планети приріст захворюваності на РМЗ становить близько півтора відсотка в рік, а це становить близько 650 тис. нових випадків. Наприклад, у 2018 р. в світі діагностували понад півтора мільйона нових випадків РМЗ, і понад п'ятсот тисяч осіб жінок померли від захворювання. Аналогічна обстановка виявляється в більшості розвинених країн світу. Найвища захворюваність РМЗ в індустріально розвинутих країнах Австралії, Північної Америки, Західної і Північної Європи. Таке поширення РМЗ в країнах, що розвиваються пояснюється, тим, що в них проходить урбанізація та прийняття західного способу життя: населення стало малорухливим, розвинулася надлишкова маса тіла, молоді люди пізно народжують, скорочується число пологів і тривалість грудного вигодовування). Обчислено, що більше половини (53%) нових випадків РМЗ і 70% смертей диспропорційно приходиться на країни розвиненої економіки, в яких проживає 82% населення земної кулі.

Зо прогнозами Міжнародної асоціації з вивчення раку, до 2030 року кількість хворих на РМЗ буде становити 2 100 000 чоловік.

В Україні теж складна ситуація. Щорічно в ній реєструється понад 16 тис. нових випадків РМЗ. Серед них 24,5% складають жінки репродуктивного віку [26]. Щорік помирає більше 7,8 тис. жінок, з них більше 20% – в репродуктивному віці. За останні 30 років рівень захворюваності на РМЗ зріс в 2,5 рази, при цьому щорічний приріст складає 7,1%. Виявлений на ранніх стадіях РМЗ (0-I-II ступеня) (згідно європейських досліджень), у приблизно 90% жінок є виліковним.

Всесвітній досвід також свідчить, що виявлення пухлини в молочній залозі до її клінічних проявів дозволяє виконувати органозберігаючі і реконструктивно-пластичні операції при РМЗ і забезпечує 20-річну виживаність у 92%-98% пацієнтів [27-34]. Трепан-біопсія є «золотим стандартом» діагностики раку грудної залози у всьому світі і дозволяє одержати стовпчик тканини, доступний гістологічному дослідженню, щоб визначити рецепторний статус пухлини і вивчити тканинні маркери. Це дозволяє онкологу підібрати максимально потрібну схему лікування у вигляді доопераційної (неадювантної) хіміотерапії і в подальшому оцінити ефективність застосованої схеми. Відповідь на доопераційну терапію як правило проявляється зменшенням розмірів пухлини та в подальшому лікуванні застосування органозберігаючих методик під час операцій.

При проведенні гістологічного дослідження злоякісної пухлини грудної залози у переважній більшості випадків (95%) виявляється протокова карцинома. Щоби оцінити її ступінь диференціації в дослідженні ми оцінюємо наявність та кількість залоз (у відсотках), ступінь ядерного поліморфізму, та кількість мітозів на 10 полів зору при збільшенні ×400. Дані показники вираховуються в балах (до 3) і це дає можливість визначити тип раку, та ступінь диференціації з інтерпретацією результатів (G1, G2, G3). Це так звана Ноттінгемська градація ступеня диференціювання пухлини. Наприклад, в тканині грудної залози виявляється протокова карцинома з формуванням

подекуди тубулярних структур різного ступеня диференціації. Строма ущільнена, волокниста. Формування тубулярних структур спостерігається на 50% площі пухлини – 2 бали. Ядра пухлинних клітин дещо збільшені, проте переважно мономорфні, подекуди з нечітко окресленими базофільними ядерцями (помірно виражений ядерний поліморфізм – 2 бали). Кількість мітозів – 5 на 10 полів зору при збільшенні ×400 (2 бали) (рисунок 1.3). За Ноттінгемською градацією загальний бал 2+2+2=6 – пухлина помірнодиференційована (G2).

Поряд із цим для лікування та прогнозу виникнення рецидивів виникає необхідність визначати рецепторний статус пухлини, проліферативний потенціал та молекулярно-генетичний тип, що можливо лише при проведенні імуногістохімічного дослідження.

При дослідженні експресії рецепторів естрогену та прогестерону визначають як кількісні так і якісні показники. Ми використовуємо методику Allred (Quick score) згідно з якою оцінюють 2 критерії: кількість позитивних клітин (PS-proportion score) та інтенсивність забарвлення (IS intensity score). Отримані бали складають в загальний бал (Total score) від 1 до 5 (враховуючи відсотки від 1 до 100).

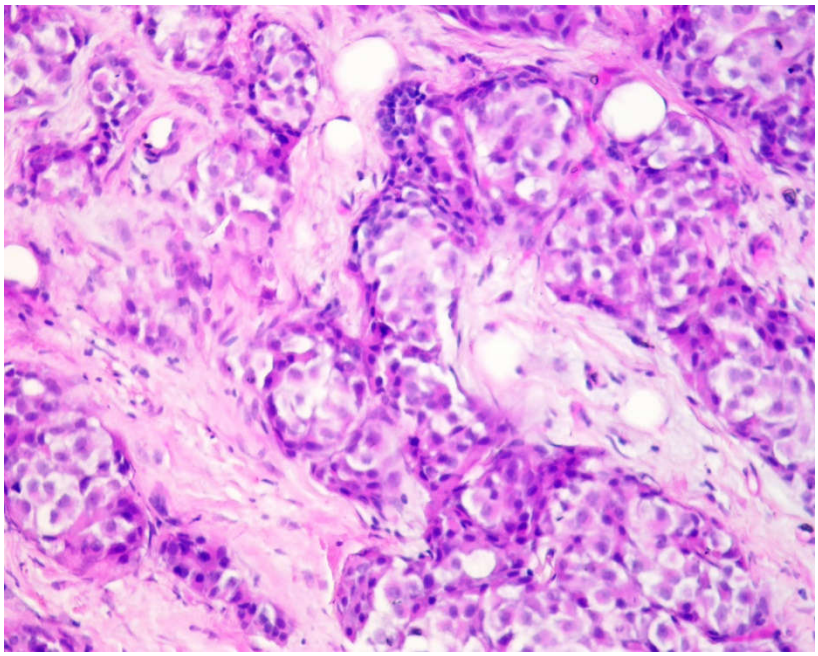


Рисунок 1.3 – Трепан біопсія грудної залози. Протокова інфільтруюча карцинома не специфікованого типу G2. Забарвлення гематоксиліном та еозином. ×200

Естроген рецептор α (DAKO, clone EP1) – позитивна реакція в 85% клітин пухлини (PS=5) значної інтенсивності (IS=3) TS=5+3=8 – позитивний результат.

Прогестерон рецептор (DAKO, clone PgR 636) – позитивна реакція у 80% клітин пухлини (PS=5) значної інтенсивності (IS=3) TS=5+3=8 – позитивний результат (рисунки 1.4, 1.5).

Таким чином пухлина є гормон позитивною з експресією рецепторів стероїдних гормонів. Ампліфікація гену HER-2/neu та експресія HER2 протеїну спостерігаються у 25-30% хворих на рак грудної залози. Такі порушення генному та експресія протеїну є несприятливими прогностичними ознаками, що вказують на підвищений ризик рецидивів та скорочення терміну життя після лікування. Такі пацієнти є резистентними до гормональної терапії відносно резистентними до специфічного лікування, проте чутливими до особливих протипухлинних препаратів.

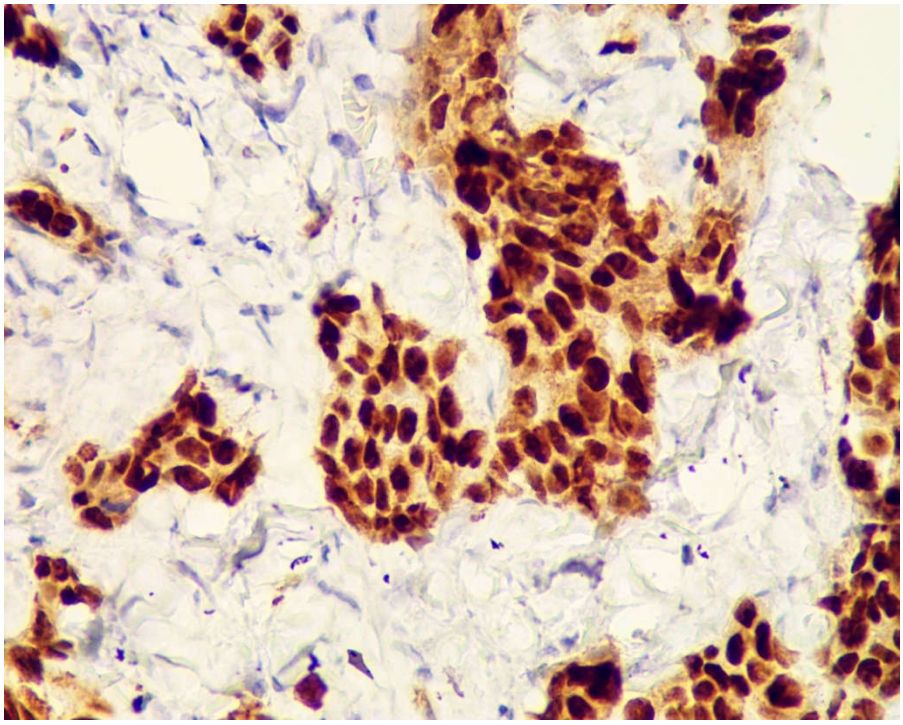


Рисунок 1.4 – Естроген рецептор α (DAKO, clone EP1) – позитивна реакція в ядрах 85% клітин пухлини (PS=5) значної інтенсивності (IS=3) TS=5+3=8 – позитивний результат. ×200

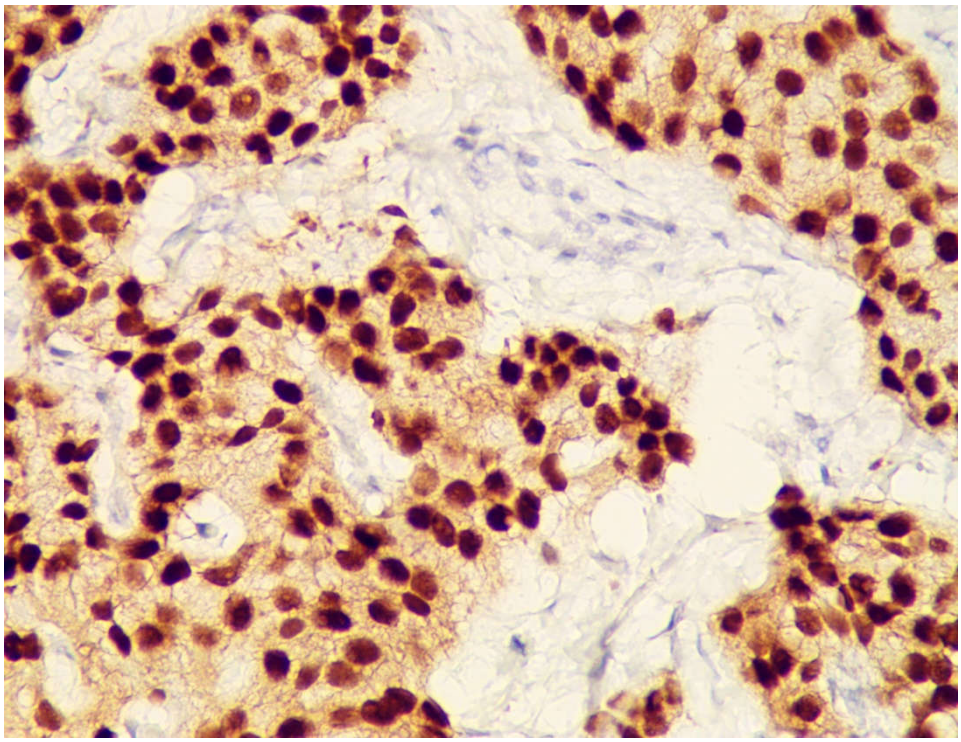


Рисунок 1.5 – Прогестерон рецептор (DAKO, clone PgR 636) – позитивна реакція у 80% клітин пухлини (PS=5) значної інтенсивності (IS=3) TS=5+3=8 – позитивний результат. ×200

Таким пацієнтам також рекомендоване лікування Транстузубамом (Herceptin). Накопичення білка HER2 можна визначити імуногістохімічними методиками. Показник оцінюється мембранним забарвленням (слабке, ледь помітне, неповне, повне) певного відсотка клітин інвазивної пухлини і інтерпретується наступним чином: 0(негативна); 1+ (негативна); 2+(сумнівна); 3+ (позитивна) згідно рекомендацій ASCO/CAP 2013.

Наприклад, Онкопротеїн c-erbB-2/ neu (HER-2/neu) (DAKO, поліклональні) – повне інтенсивне безперервне мембранне забарвлення визначається у 53 % клітин пухлини – 3+ (рисунок 1.6).

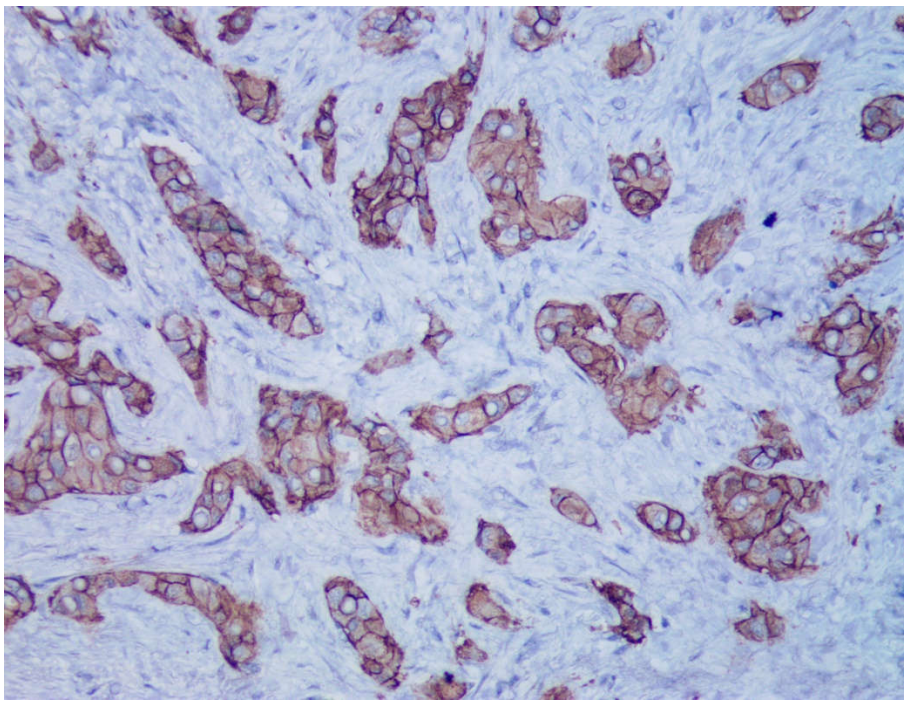


Рисунок 1.6 – Онкопротеїн c-erbB-2/ neu (HER-2/neu) (DAKO, поліклональні) – повне інтенсивне безперервне мембранне забарвлення визначається у 53 % клітин пухлини – 3+. ×200

Молекулярно-генетичний (внутрішній) підтип пухлини визначається за рівнем проліферації клітин пухлини (швидкість росту) за допомогою Ki-67. Це ядерний показник. Наприклад, Ki-67 антиген (DAKO, clone MIB-1) – позитивна реакція високої інтенсивності у 9 % клітин пухлини (рисунок 1.7).

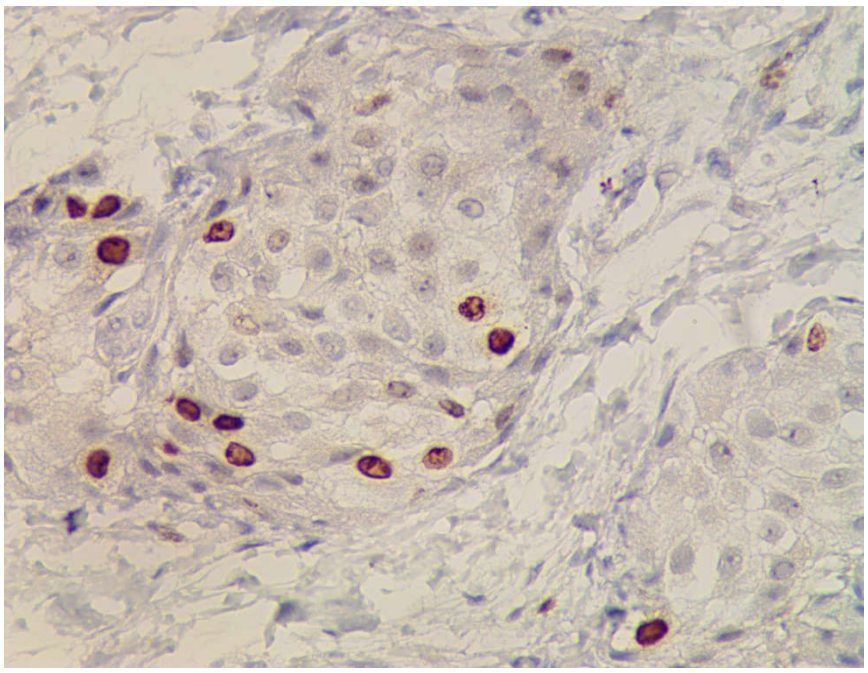


Рисунок 1.7 – Ki-67 антиген (DAKO, clone MIB-1) – позитивна ядерна реакція високої інтенсивності у 9 % клітин пухлини

В залежності від імуногістохімічної експресії рецепторів естрогену, прогестерону та онкопротеїну HER-2/neu, а також визначення рівня проліферації клітин пухлини за допомогою Ki-67 визначають молекулярно-генетичний підтип пухлини.

Виділяють 4 основних молекулярно-генетичних підтипи раку молочної залози: люмінальний А, люмінальний В, HER-2/neu-ампліфікований, базальноподібний. Всі ці підтипи відрізняються за своєю біологічною поведінкою, мають різний прогноз та потребують різного лікування.

Приклади висновків:

Приклад 1

Помірнодиференційована (G2) інвазивна протокова карцинома грудної залози (код МКХ-О: М8522/3). За результатами ІГХ-дослідження пухлина HER-2/neu позитивна, нелюмінального типу:

– естроген (ER+) – нечутлива;
– прогестерон (PR+) – нечутлива;

– HER-2/neu – позитивний результат (повне інтенсивне безперервне мембранне забарвлення 53 % клітин пухлини – 3+);

– 42% клітин позитивні (++) на маркер проліферативної активності Ki-67.

Приклад 2.

Естроген рецептор α (DAKO, clone EP1) – позитивна реакція у 82% клітин пухлини (PS=5) помірної інтенсивності (IS=2). TS=5+2=7 – позитивний результат. Прогестерон рецептор (DAKO, clone PgR 636) – специфічне забарвлення не визначається. TS=0+0=0– негативний результат (внутрішній контроль позитивний). Онкопротеїн c-erbB-2/ neu (HER-2/neu) (DAKO, поліклональні) – специфічне забарвлення не визначається. 0 балів – негативний результат. Ki-67 антиген (DAKO, clone MIB-1) – позитивна реакція значної інтенсивності у 23% клітин пухлини. Патоморфологічний висновок. Помірнодиференційована (G2) інвазивна протокова карцинома грудної залози (код МКХ-О: 8500/3). За результатами імуногістохімічного забарвлення пухлина люмінального В типу, HER-2/neu негативна:

– естроген (ER+) – чутлива (82% клітин +++);

– прогестерон (PR+) – нечутлива (внутрішній контроль позитивний);

– HER-2/neu – негативна (0 балів);

– 23% клітин позитивні (+++) на маркер проліферативної активності Ki-67.

Приклад 3.

Імуногістохімічне дослідження

Естроген рецептор α (DAKO, clone EP1) – позитивна реакція у 96% клітин пухлини (PS=5) помірної інтенсивності (IS=2). TS=5+2=7 – позитивний результат.

Прогестерон рецептор (DAKO, clone PgR 636) – позитивна реакція у 95% клітин пухлини (PS=5) помірної інтенсивності (IS=2). TS=5+2=7 – позитивний результат. Онкопротеїн c-erbB-2/ neu (HER-2/neu) (DAKO, поліклональні) –

слабке, ледь помітне неповне мембранне забарвлення у 8% клітин пухлини, 1+ бал – негативний результат.

Ki-67 антиген (DAKO, clone MIB-1) – позитивна реакція помірної інтенсивності у 11% клітин пухлини. Високодиференційована (G1) інвазивна протокова карцинома грудної залози (код МКХ-О: М8500/3).

За результатами ІГХ-дослідження пухлина люмінального А типу:

– естроген (ER+) – чутлива (96% клітин ++);
– прогестерон (PR+) – чутлива (95% клітин ++);
– HER-2/neu – негативна (1+ бал);
– 11% клітин позитивні (++) на маркер проліферативної активності Ki-67.

1.4 Висновки до розділу 1

В даному розділі проведено класифікацію типів знань, проаналізовано моделі представлення знань, проаналізовано імуногістохімічне зображення.

2 ФРЕЙМОВА МОДЕЛЬ ПРЕДСТАВЛЕННЯ ЗНАНЬ

2.1 Структура даних фрейму

Теорію фрейму було запроваджено відомим американським ученим Марвіном Мінським (1975 рік). Ця теорія належить до психологічних концепцій і способів розуміння те, як ми сприймаємо (бачимо, чуємо) явища, процеси, об'єкти тощо.

М. Мінський розглядав фрейм як структуру даних для представлення множини стереотипних ситуацій, подій та об'єктів, а також їх характеристик, знаків та властивостей. Теорія фреймів заснована на сприйнятті фактів через отриману ззовні інформацію про певне явище з даними, вже наявними, накопиченими емпіричним шляхом або отриманими в результаті розрахунків. Коли людина опиняється у новій ситуації, вона згадує у пам'яті базову структуру, звану фреймом. Фрейм – це одиниця представлення знань, які пам'ятають у минулому, і деталі яких може бути змінено залежно від ситуації. Клас деяких об'єктів (процесів) може бути визначений типовим (базовим) об'єктом, який включає найбільш суттєві характеристики об'єктів цього класу. Подібним чином, об'єкт може характеризуватися трьома складовими (об'єкт, атрибут_ j , значення_ j). Тут будь-які дві складові (атрибут, значення) являють слот. Легко бачити, що у фреймі є різні (найважливіші) слоти, які характеризують цей об'єкт «закупівлі».

Зібравши всі три складові, що торкаються даного об'єкту, отримуємо об'єктне представлення області мислень, щодо цього об'єкту. Тотальний вид цієї думки наступний:

Об'єкт(атрибут_ j значення_ j), $j = 1, ..., m$.

Тому замість побудови різних незалежних формул ми будуємо більшу структуру повної інформації про об'єкт, звану фреймом. Якщо потрібна

інформація про певний об'єкт, вона посилається на відповідний фрейм, у якому є властивості та факти про цей об'єкт. Звернемо увагу, що представлення об'єкта може бути вилучено як з логічних представлень, так і з інших представлень знань.

Інформаційні та процедурні елементи забезпечують перетворення інформації всередині фрейму та її зв'язок з іншими фреймами. Елементи фрейму називаються слотами, що означає наповнювачі.

Фрейм представлений певною структурою даних, складається з довільного числа слотів, серед яких є системні слоти і слоти, що визначаються користувачем. Кожен слот характеризується певною структурою і унікальним ім'ям усередині даного фрейму. Як системні можуть, наприклад, бути визначені наступні слоти: покажчик фрейму-батька, покажчик прямого дочірнього фрейму, користувач фрейму, дата визначення фрейму і його останньої модифікації, а також деякі інші. Системні слоти використовуються при редагуванні БД і управлінні виводом.

Фреймова система – це ієрархічна структура, вузлами якої є подібні фрейми. Розглянемо значення кожного елемента фрейму.

1) Фреймове ім'я. Це ідентифікатор, присвоєний фрейму, фрейм повинен мати ім'я, унікальне в даній системі фреймів (унікальне ім'я). Кожен фрейм, як показано на малюнку 2.1, складається з довільної кількості слотів, деякі з яких зазвичай визначаються системою для виконання певних функцій, а інші визначаються користувачем. До них відносяться слот IS-A, який відображає батьківський фрейм цього фрейму, слот покажчика дочірнього фрейму, в якому перераховані ці покажчики фреймів, слот для введення імені користувача, встановленої дати, дати модифікації, тексту коментаря та інших слотів. Кожен слот, своєю чергою, також представлений певною структурою даних.

2) Ім'я слота. Це ідентифікатор, наданий слоту; слот повинен мати унікальне ім'я у фреймі, якому належить. Зазвичай, ім'я слота не має семантичного значення і є ідентифікатором даного слота, але в деяких випадках може мати певне значення. Ці імена, крім IS-A (відношення IS-A), DDESEN-

DANTS (прямий покажчик на дочірній фрейм), FINEDBY (користувач, що визначає фрейм), DEFINEDON (дата визначення фрейму), M001P1E0CЖ / дата модифікації фрейму рама), COMMENT (коментар) і т. п. відноситься до імен, що використовуються для представлення структурованих об'єктів, таких як HASPART, RELATIONS та інші. Ці слоти називаються системними і використовуються при редагуванні бази знань і управлінні вихідними даними.

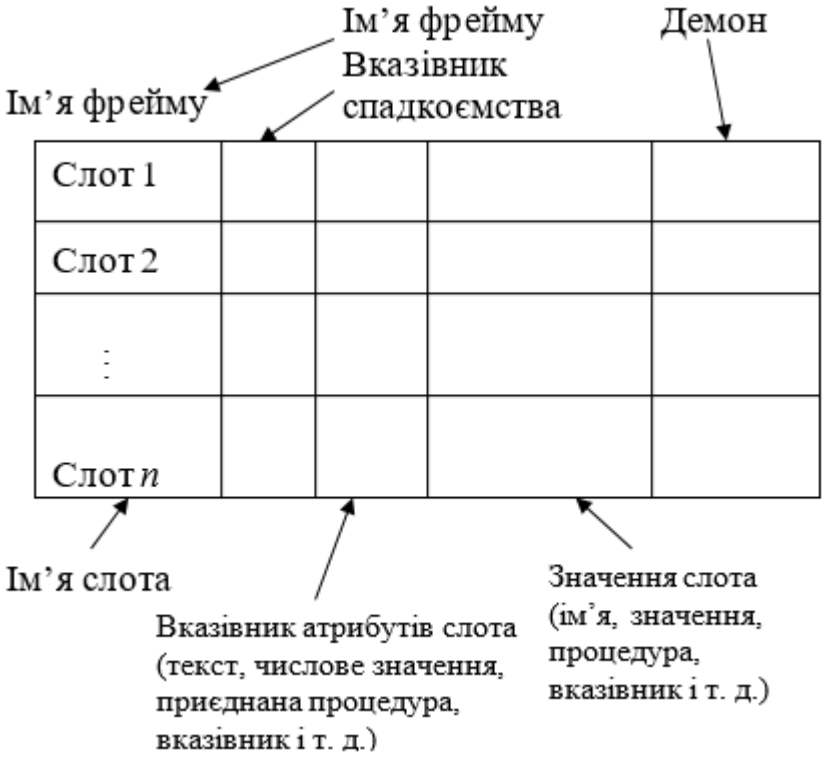


Рисунок 2.1 – Структура даних фрейму

3) Покажчики наслідування. Ці покажчики застосовуються тільки до ієрархічних систем фреймів, заснованих на абстрактно-конкретних відносинах, вони показують, яка інформація про атрибути слотів у фреймі вищого рівня успадковується слотами з такими ж іменами у фреймі рівня. Типові покажчики успадкування: Unique (U: унікальний), Same (S: ідентичний), Range (R: встановити межі), Override (O: ігнорувати) тощо. : S - всі слоти повинні мати однакові значення, R - значення слотів фрейму нижнього рівня повинні знаходитися в межах, зазначених значеннями слоти фрейму верхнього рівня, O

- якщо не вказано, значення слота фрейму верхнього рівня стає значенням слота фрейму нижнього рівня, але якщо нове значення слота фреймів нижнього рівня вказуються як слот цінності.

О служить як покажчиком U, і вказівником S. Незважаючи на те, що в більшості систем дозволено кілька параметрів для вказівки успадкування, у багатьох випадках дозволено тільки один варіант. В цьому випадку можна припустити, що використовується вказівник О за замовчуванням.

На рисунку 2.2 показані три варіанти вказівки успадкування та наведені значення, що видаються при запиті значень слотів відповідних фреймів.

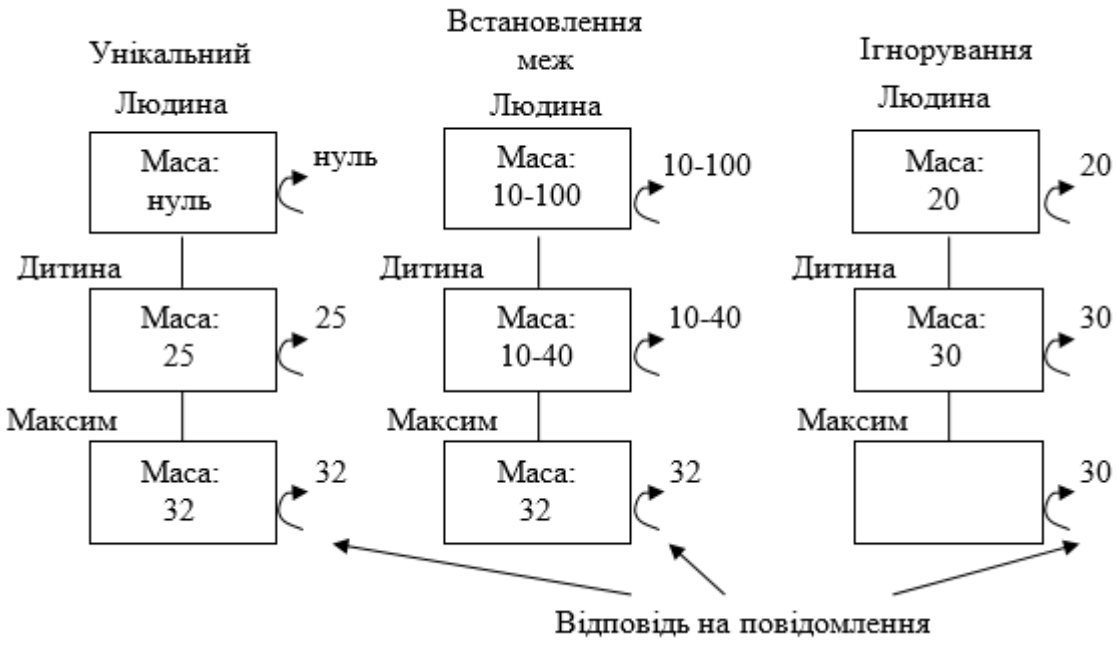


Рисунок 2.2 – Покажчики успадкування та відповідь на звернення до значення слота

4) Вказівка типу даних. Вказує, що слот має числове значення або є вказівником на інший фрейм (тобто відображає ім'я фрейму). Типи даних включають FRAME (покажчик), INTEGER (ціле число), REAL (дійсне), BOOL (логічне), LISP (приєднана процедура), TEXT (текст), LIST (список), TABLE (таблиця), EXPRESSION (вираз) та інші. .

5) Значення слота. Елемент введення значення слота. Значення слота має відповідати зазначеному типу даних слотів, а також має виконуватися умова успадкування.

6) Демон. Демон – це процедура, яка запускається автоматично при виконанні певної умови. Демони запускаються при отриманні доступу до правильного розташування. Наприклад, демон IF-NEEDED запускається, якщо при доступі до слота його значення не було визначено, IF-ADDED запускається, коли значення підставляється в слот, IF-REMOVED запускається, коли значення слота видаляється. Крім того, демон – це свого роду приєднана процедура.

7) Підключена процедура. Значення слота може бути програмою процедурного типу, званої службовою (servant) (Лісп) або методом (Smalltalk). У цьому випадку приєднана процедура запускається на повідомленні, відправленому з іншого фрейму (оскільки стан виконання в цьому випадку такий же, як в об'єктно-орієнтованій мові, мова типів фреймів також називається об'єктно-орієнтованою мовою, однак, по порядку, щоб уникнути плутанини з мовою типів Smalltalk, зазвичай називається "мовою типів фреймів"). Коли ми говоримо, що процедурні та декларативні знання об'єднані в рамках моделей представлення знань, ми розглядаємо демонів та приєднані процедури як процедурні знання. Крім того, мова представлення знань з фреймів не має певного механізму керування висновком, тому користувач повинен реалізувати цей механізм через пов'язану процедуру. Однак ця мова має дуже велику універсальність, що дозволяє, крім ієрархічного та мережевого представлення знань з використанням растрової системи, ефективно писати будь-яку програму управління виведенням одночасно з використанням приєднаної процедури. У той же час, це додаткове навантаження для користувача. Таким чином, мову представлення фреймів можна назвати мовою, орієнтованою на фахівців зі штучного інтелекту, а також мовою, орієнтованою на вирішення складних прикладних завдань. Також відомі приклади систем, які дають змогу застосовувати виробничі правила як тип даних. Це пов'язано, з

одного боку, про те, більшість систем, орієнтованих вирішення складних завдань, містять продуктивну систему як компонента, з другого боку, зниження навантаження користувача. Крім того, відомі приклади систем типу ZERO, які дозволяють використовувати функції Prolog як пов'язану процедуру.

Слот може містити не лише конкретне значення, але і ім'я процедури, що дозволяє обчислити його за заданим алгоритмом, а також одну або декілька продукцій (евристик), за допомогою яких це значення можна знайти [19].

Можна говорити про існування трьох видів слотів: [16]

1. Іменовані слоти, які можуть бути заповнені даними, наприклад, слот ОБ'ЄМ ДВИГУНА у фреймі МАШИНА. Дані, що заповнюють слот, можуть бути рядкового типу, цілими, логічними тощо. Деякі слоти можна заповнити за замовчуванням. Наприклад, у тому ж фрейм слот НАЯВНІСТЬ КОЛІС може бути заповнений за замовчуванням, оскільки це вірно в більшості випадків.

2. Слоти можуть мати типи *ISA* або *AKO*. Слот *ISA* вказує на участь даного фрейма в ієрархії фреймів і містить ім'я фрейма, що відповідає найбільшому класу; наприклад, для фрейма МАШИНА, це може бути фрейм АВТО.

Слот *AKO* вказує на родову приналежність фрейма, тобто чи має він родову чи конкретну властивість; наприклад, для фрейма АВТО це може бути властивість МАШИНА. При цьому ця властивість може успадковуватися фреймом МАШИНА за *ISA* -ієрархією.

3. Слоти можуть носити процедурний характер. Наприклад, у фреймі ТВАРИНА, значення слота КІЛЬКІСТЬ ЩОДНЯ ВИПИТОЇ ВОДИ обчислюється як функція її віку, розміру і ваги. Зрозуміло, фрейм повинен містити відповідні слоти, а саме, ВІК, РОЗМІР, ВАГА.

Тепер звернемо увагу, що один із слотів фрейма КІТ повинен мати тип *ISA* і містити інформацію про те, що кіт є ЧОТИРИЛАПА ТВАРИНА. Це, до речі, означає, що КІТ успадкує значення слотів фрейма ЧОТИРИНОГА ТВАРИНА, таких як КІЛЬКІСТЬ КІНЦІВОК, НАЯВНІСТЬ ШЕРСТІ і ін.

Приведемо визначення фрейма в нотації Бекуса-Наура:

• <фрейм>::= <ім'я фрейма> {<тіло фрейма>}

• <тіло фрейма>::= <множина слотів>

• <множина слотів>:: = <слот> | <слот>, <множина слотів>

• <слот>:: = <ім'я слота>: <значення слота>

• <значення слота>:: = <ім'я фрейма> | <ім'я процедури> | <множина>

• <множина>:: = <дискретна множина> | <щільна множина>

• <дискретна множина>:: = елемент множини>, <множина>

• <елемент множини>:: = <ідентифікатор>

• <щільна множина>:: = <інтервал> | <напівінтервал> | <відрізок>

2.2 Властивості фреймів

Розглянемо основні характеристики фреймів.

1. Базовий тип. Базові фрейми використовуються для позначення найважливіших об'єктів, що дозволяє швидко зрозуміти суть даного об'єкта та його стан, проте потрібна пам'ять, щоб запам'ятати різні положення фрейму. Отже, лише найважливіші об'єкти цієї теми зберігаються як базові фрейми, у тому числі будуються фрейми останніх класів. Фрейми для нових доменів засновані на фреймах. Однак кожен фрейм містить розташування покажчика субструктури, що дозволяє різним фреймам спільно використовувати одні й ті самі частини.

2. Процес порівняння. Процес перевірки того, що вибрано правильне зображення, називається процесом зіставлення (порівняння). Фреймова система шукає фрейм, що відповідає (релевантній) меті (даній ситуації). Інакшими словами, значення (обмеження) положення фрейму в системі фреймування відображаються значення цільових атрибутів. Процес порівняння здійснюється наступним чином:

а) Насамперед, у вигляді пропозиції та інтуїції вибирається якась базова структура з урахуванням виявлених індикаторів актуальності, тобто з допомогою субструктур ця структура підтверджує чи ні свою актуальність. У цьому випадку, виходячи з поточних цілей, визначається, яке обмеження розташування слід використовувати при відображенні. Після підтвердження процес сполучення завершується, інакше виконується крок б).

б) Якщо в цьому фреймі є слот, в якому сталася помилка, наприклад, з точки зору узгодженості з інформацією за умовчанням, потрібна інформація, щоб гарантувати, що відповідне значення для цього слоту призначено. Призначення інформації, необхідної для цього слота, має відповідати обмеженням та сподіванням цього слота.

с) Якщо попередні два кроки не дали результату, управління передається іншому відповідному фрейму в цій системі. Якщо цей фрейм не співпадає, керування передається відповідному фрейму з іншої системи фреймів. Якщо останній збіг не вдалося, то вирішення проблеми немає.

3. Ієрархічна структура. Фрейм зазвичай слідує свого роду ієрархічній структурі, особливість якої полягає в тому, що значення атрибутів фрейму вищого рівня є спільними для всіх фреймів нижнього рівня, пов'язаних з вищими рівнями (рисунок 2.3). Така структура дозволяє зручно систематизувати та реєструвати такі поняття, додавати нові поняття чи знання до відповідних поз ієрархії, спростити вираження протиріч у знаннях, переглянути знання та заповнити систему більш гнучкими рамками.

Розглянемо з прикладу структури системи моделювання процес видачі банківської позички під заставу (рисунок 2.3). Ця система містить фрейм «позичальника», яка є класом клієнтів і належить до суперкласу позичальників. Він містить показники, які стосуються клієнту-людині, такі як мета позички, інформація про заставу клієнта, зобов'язання, майно тощо.

Успадкування властивостей фреймів представлене ставленням кожного фрейму до інших фреймів. Наприклад, особистий фрейм клієнта має фрейм клієнта як суперклас і успадковує його характеристики.

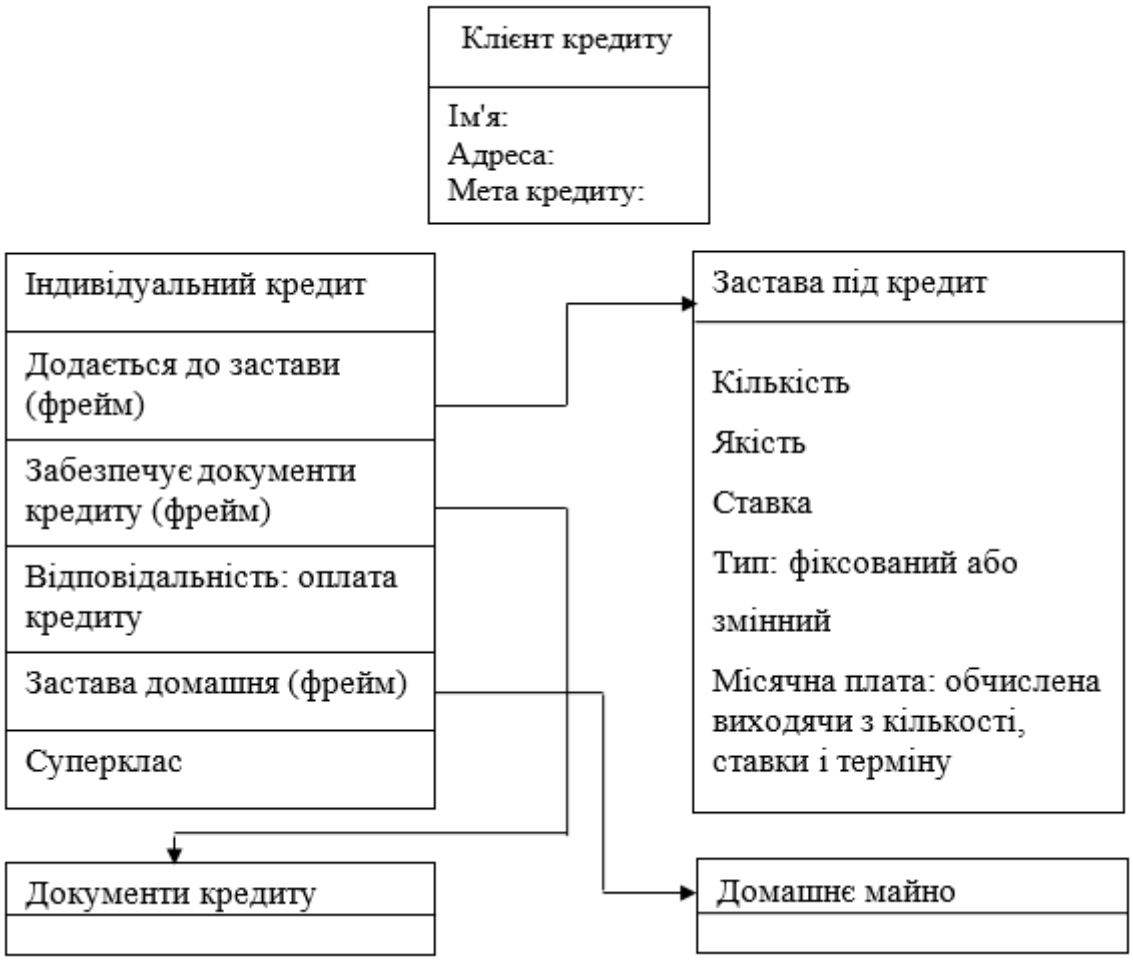


Рисунок 2.3 – Приклад фрейму моделювання процесу видачі банківського кредиту під заставу

Важливо те, що фрейми мають складну структуру даних, в якій атрибути фрейму самі є фреймами. Наприклад, «конкретний клієнт» для системи кредитування під забезпечення сам є фреймом з властивими йому атрибутами.

Процедурний характер фрейму "кредит під заставу" у тому, що величина в слоті "плата застави" може бути результатом формули, у якій використовуються слоти "кількість", "процентна ставка" і "тривалість".

На рисунку 2.4 атрибут «рух тварини» є загальним для птахів та канарки, яка знаходиться на найнижчому рівні. Якби людська пам'ять мала аналогічну структуру, то можна було б систематизувати та запам'ятовувати аналогічні концепції, уникнути непотрібних складнощів, що стосуються атрибутивної

інформації, та додавати нові концепції чи знання на відповідні позиції у цій ієрархії (навчання). Це спростило б виявлення протиріч у знаннях та управлінні послідовністю, і якби можна було зрозуміти, що цей птах навіть без використання конкретних знань (наприклад, це тригер), то гнучкість системи значно збільшилася б, наприклад, можна було б робити висновки, ґрунтуючись на знання про птахів та тварин. (Слід зазначити, що така ієрархічна структура властива не тільки представленню знань за допомогою фреймів. Наприклад, у мові представлення знань, що об'єднує фрейми та продукційні системи, іноді частина такої структури використовується як база даних продукційної системи).

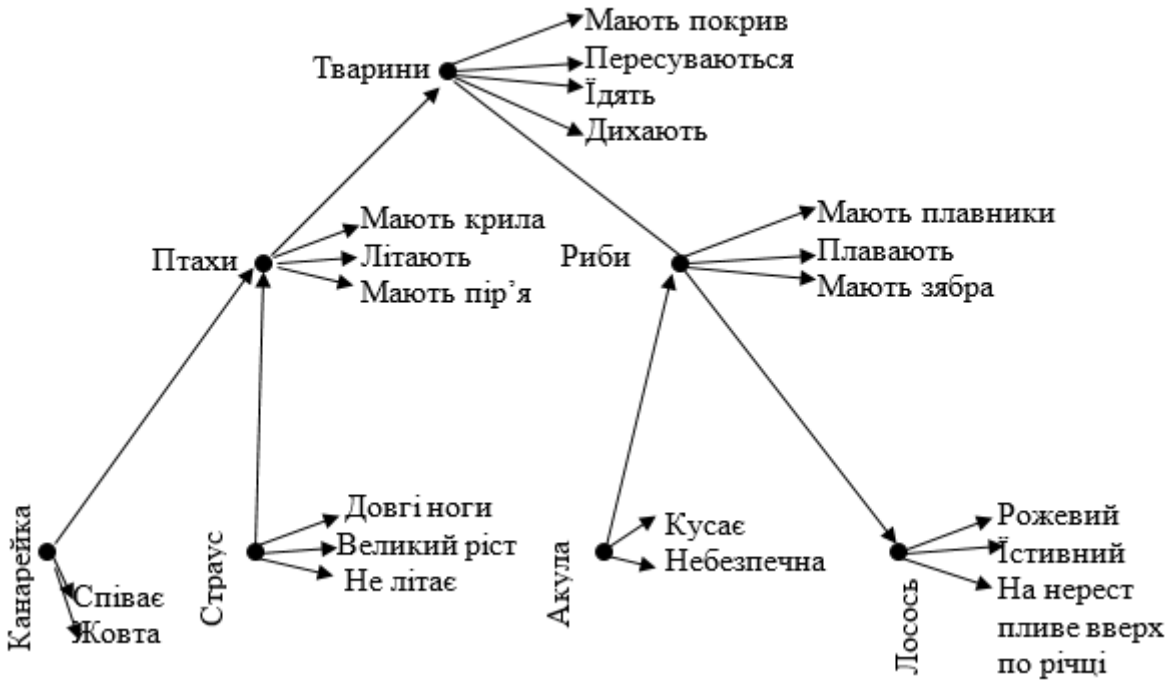
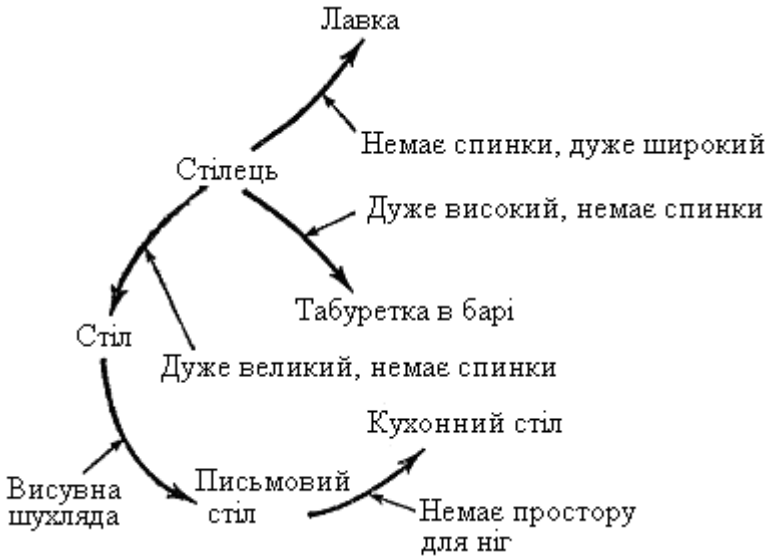


Рисунок 2.4 – Концептуальна схема класифікаційної ієрархічної структури

4. Міжфреймові мережі. Запам'ятовування концептуального об'єкта з ієрархічною класифікаційною структурою легко пояснюється фреймовою моделлю, проте якщо процес зіставлення був невдалим, виникає необхідність шукати фрейм, аналогічний попередньому. Цей пошук, що виконується з використанням покажчиків відмінностей, можливий шляхом об'єднання

фреймів, що описують об'єкти з невеликими відмінностями, з цими покажчиками та формування масиву подібних фреймів.

На рисунку 2.5 показаний приклад так званої мережі Уїнстона. Якщо при порівнянні фрейма стільця з'ясовується, що об'єкт занадто великий і він не має спинки, то фрейм столу визначається за допомогою покажчика відмінності. Якщо об'єкт виявляється занадто широким, а спинка відсутня, можна ефективно шукати фрейм лавки, використовуючи інший маркер відмінності.



Рисунку 2.5 – Мережа подібності за Уінстоном

У цьому прикладі були розглянуті покажчики розрізнення, проте семантичні мережі можуть бути побудовані з використанням всіх видів інших покажчиків, і за їх допомогою можна з великою ефективністю робити різні висновки. (У таких випадках слід розглядати систему фреймів як таку, що включає семантичну мережу.)

5. Значення за замовчуванням. Коли людина щось розглядає і думає про те, що це буде означати, або подумки уявляє щось і думає про те, що це означатиме, тоді цей процес можна представити як розподіл певних значень між термінальними слотами фрейму. При цьому, у випадку уявного представлення, межі, що торкаються розподілу цих значень є широкими.

Наприклад, читаючи фразу "Андрій взяв м'яч", уява читача малює не абстрактний м'яч, а дуже точний м'яч, наприклад, м'яч для гри в волейбол або футбол. Крім того, цей м'яч повинен мати певні атрибути, такі як розмір, колір і маса, значення яких розуміються за умовчанням. Безперечно, ці значення згадуються асоціативно з особистого досвіду читача. Ці значення за умовчанням слабко пов'язані з слотами, та поступово замінюються дійсною інформацією.

Це полягає в припущенні: «Поки не розподілені термінальні значення, рішення про занесення в довгострокову пам'ять не приймається. До того часу у фреймі зберігається значення за замовчуванням, хоча й не має до нього жодного відношення». Висновки, отримані з значень за замовчуванням, називаються висновками за замовчуванням. На підставі цих висновків можна продовжити робити висновки та заповнювати прогалини у наданій інформації. Ця функція загалом покращує можливості системи. Однак є ризик зробити невірні висновки, засновані на помилках.

Висновок за замовчуванням виробляє дуже важливу функцію при розпізнаванні образів або мови. Наприклад, якщо видно лише частину зображення, то, перевизначивши значення за замовчуванням для цієї частини, можна намалювати все зображення. Так само використання значення за умовчанням може відновити сенс контексту, з якого витягуються окремі пропозиції.

6. Відношення «абстрактне – конкретне» і «ціле – частина». Розглянута вище ієрархічна структура заснована на відношенні «абстрактне - конкретне», проте, крім цих структур, існують й інші, засновані на відношенні «ціле – частина».

Відношення «абстрактне – конкретне» характеризується тим, що у вищих рівнях, як показано рисунку 2.4, присутні абстрактні об'єкти (концепції), але в нижчих рівнях – конкретні об'єкти, а об'єкти нижчих рівнів успадковують атрибути вищого рівня. об'єкти. Ці відношення також називаються відносинами IS-A чи KIND-OF. Такі назви пояснюються формами позначень «канарейка

IS_A птах» (канарейка – птах) та «канарейка – is a KIND-OF птах» (канарейка – це різновид птаха).

Ще одні відношення «ціле – частина» пов'язані з структурованими об'єктами і показують, що об'єкт нижнього рівня є частиною об'єкта вищого рівня. Наприклад, стіна є структурною частиною аудиторії, але не аудиторією, тому об'єкт стіни не успадковує атрибут аудиторії, що є об'єктом верхнього рівня. Швидше успадкування атрибутів засноване на відношення типу «тіло – стіна – стіна аудиторії – стіна аудиторії А». Зазначимо, що два типи цих відношень розглядалися в теорії фреймів Мінського, але їх поділ, подібний до представленого тут, не проводився.

Практичне застосування в фреймих системах набуло лише відношення «абстрактне – конкретне». Однак іноді необхідно описувати та підтримувати структурований об'єкт, наприклад, у САПР, тому в таких випадках не можна обійтися без відношень типу «ціле – частина». І тут компоненти системи описуються відношеннями IS-A, а структура – відношеннями PART-OF. Однак щодо типу PART-OF не можна використовувати успадкування атрибутів, тому переваги моделі подання знань фреймами не видно. У цих випадках потрібні інші методи.

2.3 Системи фреймів

Отже, фрейми можуть посилатися один на одного через свої слоти. На них можуть бути задані відношення КЛАС-ПІДКЛАС (*ISA*) і РІД-ВИД (*AKO*). Фрейми в сенсі, визначеному вище, інколи називають фреймами-прототипами. Окрім цього, вводять поняття фрейму – екземпляра (або прикладу). Фрейм-екземпляр, це сукупність значень слотів, що задовольняють деякому фрейму.

Інакше кажучи, фрейм-екземпляр – це кортеж значень ознак, кожен з яких задовольняє деякому слоту [1].

Агрегатом $<r_1, r_2, \ldots, r_k>$ називають з'єднання кортежів $r_1, r_2, \ldots, r_k$ таке, що кожен кортеж r_i є фреймом-екземпляром, що задовольняє якому-небудь фрейму-прототипу.

Елементарним фреймом-прототипом називатимемо той фрейм, на який не посилається ніякий інший фрейм.

Елементарний фрейм є здійснимим, якщо:

а) значення змінних, підставлених в слоти фреймів, відповідають областям їх можливих значень;

б) на місця формальних параметрів підставлені фактичні параметри процедур і виконуються умови їх (процедур) активності.

Якщо два фрейми F_1 і F_2 зв'язані відношенням *ISA* або *AKO* (F_1 *ISA* F_2), то фрейм F_2 вважатиметься виконаним, якщо виконаним є F_1.

Основне обчислювальне завдання в системі фреймів

Нехай Σ – система фреймів і задано агрегат σ. Основне обчислювальне завдання в системі фреймів [35] полягає в ефективному обчисленні відношення істинності $\sigma| = \Sigma$.

Процедура обчислення відношення істинності включає наступні кроки:

Крок 1. Задоволення всіх областей значень слотів фреймів значеннями, що входять в цей агрегат.

Крок 2. Обчислення значень всіх функцій, фактичними параметрами яких є відповідні значення агрегату σ.

Крок 3. Перехід по зв'язках *AKO* і *ISA*.

Крок 4. Перехід до Кроку 1.

Процедура завершує роботу після того, як а) всі слоти всіх зв'язаних відношеннями *ISA* і *AKO* фреймів будуть задоволені і б) не залишиться невикористаних *ISA* і *AKO* зв'язків.

Відношення істинності | = вважається обчисленим із завершенням роботи процедури. Результатом є множина фреймів, виконаних на агрегаті σ.

Розглянемо конкретні структури систем і способи управління виводу (як інтелектуальної системи планування використовується програма NUDGE) на прикладі вельми простої інтелектуальної системи планування і за допомогою формалізму представлення знань фреймами.

На рисунку 2.6 показаний простий приклад ієрархічної структури інтелектуальної системи планування. Ця структура базується на відношеннях IS-A між фреймами, що описують деяку конференцію. Передбачається, що всі ці фрейми повинні містити такі пункти, як дата і місце проведення конференції, назва теми та прізвища доповідачів. Таким чином, на найвищому рівні ієрархії визначено фрейм «конференція», що містить чотири зазначених слота. У даному випадку конференції поділяються на конференції з комерційних питань та конференції з розвитку, що визначені як відповідні дочірні фрейми. Далі, загальною темою конференцій з комерційних питань є торгівля, а спільною темою конференцій по розвитку є освоєння нових видів продукції, які підставляються в якості значень у відповідні слоти. Крім того, на фреймі «конференція» по комерційних справах необхідний пункт «мета» (обсяг торгівлі), а у фреймі конференції з розвитку задається бюджет (на освоєння), тому до них додані відповідні слоти.

У даному прикладі як фрейму, дочірнього по відношенню до фрейму конференції з комерційних питань, визначено фрейм «четверта конференція з комерційним питанням», в чотири слота якого (виняток становить слот «тема») введені конкретні значення. Крім того, фрейми, що описують різні об'єкти, називаються шаблонами (template), а фрейми верхнього рівня, використовувані для представлення цих шаблонів, називаються фреймами класу.

Стосовно теми виконується наступна обробка. У момент звернення до слоту «тема» фрейму «4-а конференція з комерційних питань» запускається механізм управління спадкуванням, за допомогою якого здійснюється пошук і застосування значення цього слоту у відповідному фреймі верхнього рівня (якщо в цьому фреймі відсутнє дане значення, то пошук здійснюється в іншому фреймі верхнього рівня. У реальних системах з базами знань, представлених

фреймами, в цілях економії пам'яті та інших причин часто уникають застосування слотів з однаковими значеннями у фреймах верхнього рівня).

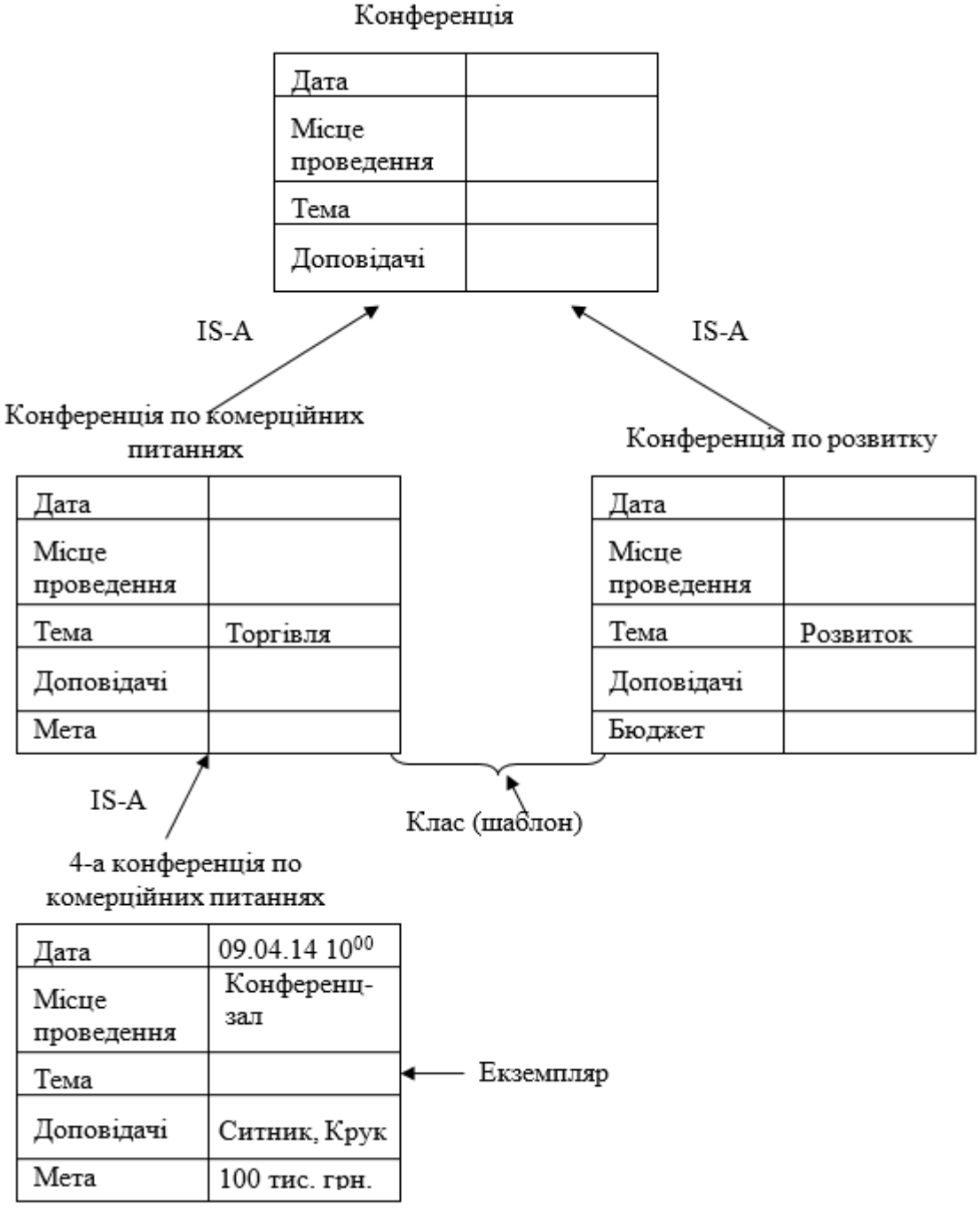


Рисунок 2.6 – Приклад фреймового представлення конференції при інтелектуальному плануванні

На рисунку 2.7 показаний приклад фрейму «4-а конференція з комерційних питань» з приєднаними демонами. Як показано на рисунку 2.8, в

кожному слоті можна вказувати три типи демонів. У загальних рисах демон функціонує наступним чином. «Місце проведення» визначено демон IF-ADDED з ім'ям «бронювання» (ім'я функції мови Лісп), який автоматично запускається при підстановці в цей слот значення «Конференц-зал». Якщо цей зал можна зайняти, то він бронюється, в іншому випадку, коли зал уже зайнятий, видається повідомлення «бронювання неможливо». Демон IF-NEEDED з ім'ям «Хто?», Приєднаний до слоту «ім'я виступаючого», у разі якщо при зверненні до даного слоту його значення було NIL (нуль), генерується запит: «Хто виступає на 4-ій конференції з комерційних питань?» Відповідь на це питання передається при підстановці вхідних даних користувача як значення слота. Таким чином, з вищевикладеного видно, що демони запускаються автоматично при зверненні до відповідного слоту.

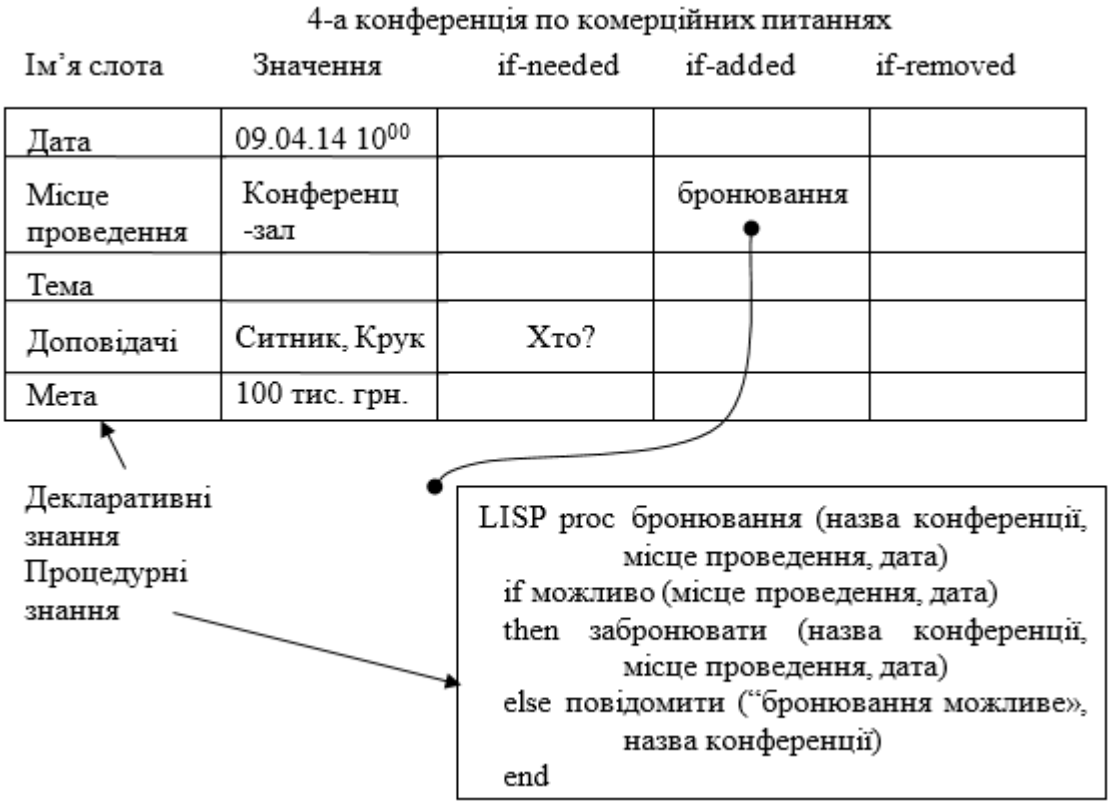


Рисунок 2.7 – Приклад приєднаної процедури типу демона, що доповнює фрейм

На рисунку 2.8 показаний приклад службової приєднаної процедури і спосіб передачі повідомлень з використанням цієї процедури. Функція (команда) MSG, наявна в приєднаній процедурі з ім'ям «вирахувати», що відноситься до слоту «обчислення» фрейму «АА», служить для передачі повідомлень. Структура пропозиції цієї функції виглядає наступним чином:

MSG (ім'я фрейму, ім'я слота, параметр, ...).

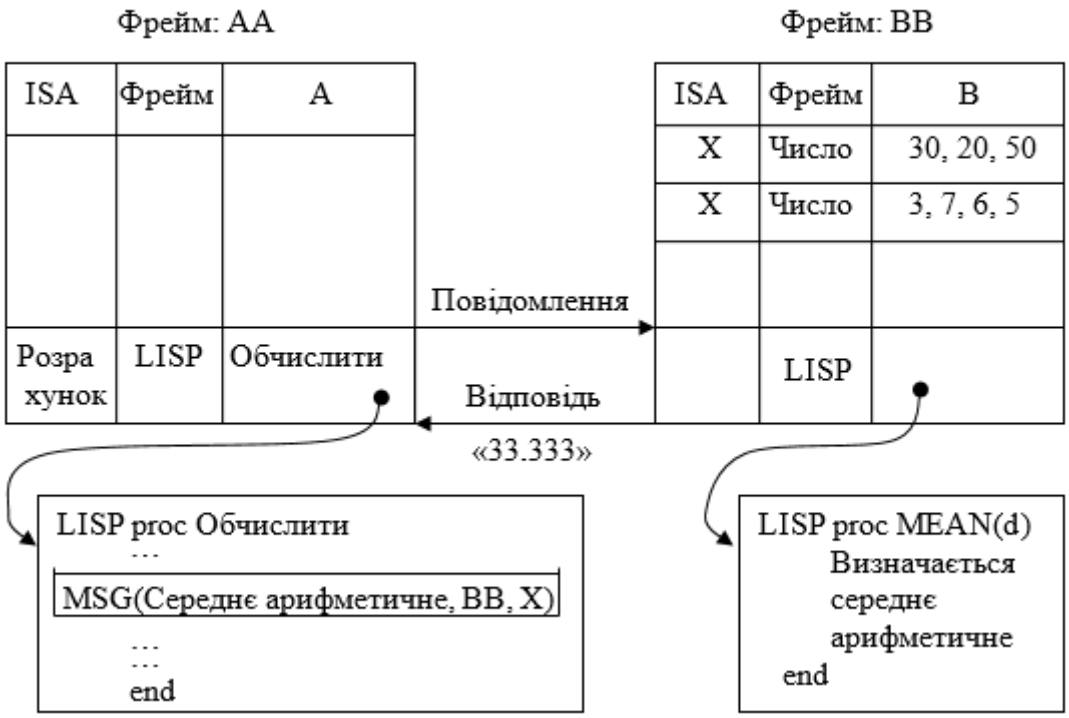


Рисунок 2.8 – Приклад приєднаної службової процедури, яка доповнює фрейм

Отже, три аргументи «середнє арифметичне», «ВВ» та «X» функції MSG є відповідно аргументами імені слота, імені фрейму і процедурної функції фрейму, передавального повідомлення. Механізм передачі прийому повідомлень за допомогою функції MSG діє таким чином. Якщо за повідомленням від іншого фрейму ініціюється приєднана процедура «вирахувати» фрейм «АА», то за допомогою функції MSG фрейму «ВВ» передається повідомлення, за яким ініціюється приєднана процедура «MEAN»

слоту «середнє арифметичне» фрейму «ВВ». За допомогою цієї процедури обчислюється величина d, тобто середнє арифметичне трьох значень 30, 20, 50 слоту «Х». Результат обчислення, тобто «33,333» передається у фрейм «АА».

Таким чином, у фреймових системах використовуються три способи управління виводу: два – за допомогою приєднаних процедур – демона та службової процедури (або методу) і один – за допомогою механізму успадкування. Останній спосіб можна назвати єдиним основним механізмом управління висновком, яким оснащуються фреймові системи. Іншими словами, за допомогою механізму керування спадкуванням, що базується на відношеннях «абстрактне – конкретне» (або відношеннях типу IS-A), здійснюється автоматичний пошук і визначення значень слотів фрейму верхнього рівня і приєднаних процедур службового типу. Крім економії обсягу пам’яті і скорочення обсягу робіт при програмуванні механізм успадкування виконує важливу роль в послідовному управлінні систем з базами знань. За допомогою об'єднання демона та службової процедури можна раціонально реалізувати будь-який механізм управління виводу. Однак для узгодженого і правильного функціонування системи в цілому, природно, необхідно її ретельне проектування. Крім того, щоб системи з базами знань, засновані на фреймових системах, ззовні виглядали інтелектуальними, при їх проектуванні необхідно передбачати застосування в складі системи приєднаних процедур. Інакше їх важко буде відрізнити від звичайних програм обробки даних. Можна сказати, що головною особливістю застосування мови представлення знань фреймами є простота написання програм для вирішення інтелектуальних проблем. Однак для людей, які не знають принципів і стратегії, що стосуються вирішення цих проблем, мова фреймів представляється лише різновидом простої об'єктно-орієнтованої мови програмування.

2.4 Універсальні мови подання знань фреймами і приклади систем

Існує безліч мов подання знань фреймами, спроектованих на основі теорії фреймів М. Мінського. Система FMS була розроблена в якості експериментального середовища для дослідників в галузі ШІ інженерії знань. В основу цієї системи була покладена система UNITS з відповідними доповненнями та змінами, що стосуються специфікації мови і розробки. Надалі вона була розширена і вдосконалена, змінилася і назва системи, яка тепер стала називатися ZERO.

2.4.1 Основні напрямки в проектуванні системи FMS

Основною причиною того, що фреймові системи вважаються найбільш сприятливим середовищем для досліджень з штучного інтелекту, очевидно, слід вважати те, що вони в цілому задовольняють численним вимогам, що стосуються представлення знань. Ці вимоги наступні:

1. Для систематизованого управління складними знаннями великого обсягу бажано організувати всі знання на основі концептуальних об'єктів.

2. З метою збільшення гнучкості системи слід зробити можливим представлення у вигляді комбінації декларативних і процедурних знань для опису пов'язаних з ними концептуальних об'єктів.

3. Оскільки концепт зазвичай має ієрархічну структуру, пов'язану з деяким ступенем абстракції, то і для подання знань слід застосовувати ієрархічну структуру.

4. При вирішенні складних проблем вважається, що різні стани виводу застосовуються в комбінаціях відповідно до ситуації. Тому і в поданні знань необхідні функції, що враховують цю обставину.

5. Придатність будь-якого способу управління виводу повинна визначатися властивостями проблеми, цілями і способом її вирішення. Тому в

середовищі дослідження інтелектуальних систем вирішення проблем з використанням методу проб і помилок має бути передбачена можливість вільного проектування та випробування користувачем різних способів управління висновком.

6. Для створення універсального дослідницького середовища недостатньо лише високої універсальності її мови – необхідно забезпечити можливість порівняно простого доповнення її різними функціями. Крім того, і сама мова має передбачати розширюваність.

Перераховані вимоги є загальними вимогами, обумовленими в середовищі досліджень по ШІ, крім того, їх можна назвати основними властивостями, які має мова представлення знань фреймами. Слід зазначити, що при проектуванні системи FMS серйозну увагу приділяли тому, щоб в достатній мірі використовувати ці особливості. В якості мови розробки системи був використаний Лісп і причина цього вибору також полягає в розширюваності і гнучкості даної мови. Крім того, спочатку система була оснащена тільки основними функціями, і передбачалося, що відповідно до застосувань вона буде потроху добудовуватися необхідними функціями. В дослідницькому середовищі вкрай важливе значення має завершеність користувача інтерфейсу. У FMS враховано і ця обставина; вона оснащена засобами допомоги та видачі інструкцій, які можна використовувати навіть в відсутність опису, а також у зручному застосуванні інтерфейсом для стандартних терміналів універсальних комп'ютерів.

2.4.2 Структура даних фрейму та слота в системі FMS

На рисунку 2.9 показана структура даних, фрейму, точніше у верхній частині цього рисунка показана структура даних фрейму, а в нижній частині структура даних слота. Всі фреймові системи складаються з окремих структур даних подібно показаній на рисунку 2.9.

Така структура даних також має широке поширення, а єдине представлення є надзвичайно важливим принципом з точки зору усунення зайвого ускладнення системи програмного забезпечення.

Ім’я фрейму	Тип фрейму
Слот AKO	
Слот D. Descendants	
Слот Description	
Слот Created By	
Слот Modified By	
Слот Created On	
Слот Modified On	
Слот 1	
…	
Слот *n*	

Ім’я фрейму	Вказівник спадкоємства	З	Тип даних	Значення	Необов’язковий параметр

Рисунок 2.9 – Структура даних фрейму та слота в системі FMS

Нижче наводиться опис цієї структури. Ім'я фрейму є ідентифікатором даного фрейму, однозначно ідентифікує його у всій фреймовій системі. Тип фрейму – це мітка, що показує, чи є даний фрейм шаблоном або фреймом класу. Слот AKO – покажчик, що показує, що даний фрейм є фреймом-батьком, причому цей слот є стандартним системним слотом. Слоти з першого по *n*-й (*n* – будь-яке ціле число) визначаються користувачем. Іншими словами, користувач за допомогою визначення цих слотів будує базу знань, представлених фреймами (також можлива відповідна побудова за допомогою системи при виконанні виводу). Фреймова система організується як система дочірніх фреймів спеціального фрейму ROOT, що приймається як стандартний.

Кожен слот містить ім'я слота, покажчик спадкування, тип даних, значення даних і необов'язковий параметр. Ім'я слота – це унікальний для даного фрейму ідентифікатор слота. Покажчик спадкування показує спосіб

успадкування атрибутів слота фрейму-батька дочірнім фреймом. U (Unique) – слот успадковується, але дані в кожному фреймі можуть брати будь-які значення; S (Same) – успадкування тих же значень даних; R (Range) – обмеження діапазону значень даних; M (Memper) – обмеження вибору з елементів, перерахованих у фреймі верхнього рівня; I (Independence) – без успадкування.

До вказівників даних відносяться наступні: ATOM (атом), TEXT (текст), TABLE (таблиця), BOOLIAN (булевий); INTEGER (цілочисельне значення), NUMBER (номер), STRING (рядок), EXPP (α – вираз), LISP (функція Ліспа, тобто приєднана процедура) і FRAME (ім'я фрейму, тобто покажчик). У полі «значення» присутні дані, тип яких вказано покажчиком даних, причому допускається визначення або підстановка тільки тих значень, які задані покажчиком успадкування. За допомогою покажчика LISP можна описувати процедуру виведення, а за допомогою покажчика FRAME – будувати міжфреймові мережі. Даними при покажчику LISP являється тільки ім'я функції, сама ж процедура зберігається у вигляді значення даних слота (ім'я якого збігається з ім'ям функції) спеціального фрейму LISPPROC, використовуваного для спільного зберігання всіх Лісп-процедур. Його покажчиком типу даних є EXPP. У полі необов'язкового параметра кожного слота можна підставляти будь-які значення, зокрема завдяки йому можна обійтися без перевірки, наприклад цілісності, системи.

2.4.3 Конфігурація системи

Система FMS складається з чотирьох основних модулів (рисунок 2.10): редактора фреймів, модуля перевірки правил, виконавчого механізму та модуля системних функцій. Редактор фреймів використовується для діалогової побудови фреймових систем, наприклад для визначення та оновлення бази знань і т. п. Модуль перевірки правил служить для виявлення помилок у базі

знань, побудованої за допомогою редактора фреймів. Виявлену помилку можна виправити з допомогою того ж фреймового редактора.

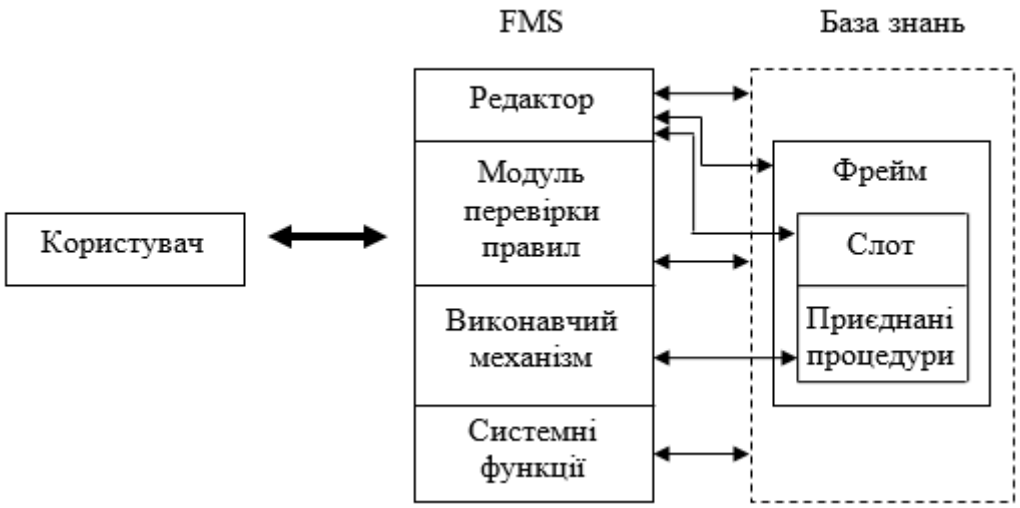


Рисунок 2.10 – Основні модулі системи FMS

Модуль перевірки правил служить для виявлення помилок у базі знань, побудованої за допомогою редактора фреймів. Виявлену помилку можна виправити з допомогою того ж фреймового редактора. Прості помилки, що стосуються, наприклад, вказівки спадкування або типу даних, виявляються редактором в процесі побудови бази даних, модуль перевірки правил використовується для виявлення більш складних помилок, наприклад протиріч між фреймами і т. п. Проте виявлення семантичних суперечностей у фреймовій системі вельми серйозна і складна задача, для вирішення якої поки не знайдені досить ефективні наукові підходи. Ця задача тісно пов'язана з дослідженнями з навчання, її належить вирішити в майбутньому. Виконавчий механізм служить для ініціювання виведення.

Модуль системних функцій містить групу Лісп-функцій, використовуваних, для полегшення написання приєднаних процедур. Деякі з цих функцій застосовувалися також і для розробки самої системи FMS. У таблиці 2.1 наведені основні системні функції і їх призначення. Функції управління слотами служать для генерації, знищення слота, звернення до нього,

а також для підстановки значення слота, звернення до нього та знищення цього значення. Найбільш важливою функцією, що стосується процедури управління виводу, є MESSAGE, яка використовується для запуску приєднаної процедури, що міститься в слоті іншого фрейму.

Таблиця 2.1 – Класифікація функцій системи FMS

а) *Функції управління фреймами*	
CREATEFRAME#	Створення
DELETEFRAME#	Видалення
GETFRAME#	Отримання фрейму з бази знань
GETFRTUPE#	Отримання типу-фрейму
GETSLOTNAMELIST	Отримання списку імен слотів, визначених у фреймі
INSTANCE#	Успадкування типу-фрейму
SUBCLASS#	Встановлення підкласу типу-фрейму
SUB-HIERLIST	Визначення ієрархічної структури бази знань

б) *Функції управління слотами*	
CREATESLOT#	Створення
DELETESLOT#	Знищення
GETSLOT#	Отримання слота фрейму
GETVALUE#	Отримання значення слота
GETOP	Отримання необов'язкового параметра слота
SETVAL#	Встановлення значення
SETOPTION#	Встановлення необов'язкового параметра
PRINTSLOT#	Друк

в) *Функція виклику іншого фрейму*	
MESSAGE#	Запуск приєднаної процедури
г) *Функції перевірки*	
CHK-TYPEVAL	Перевірка типу даних і їх значень
FRAMEP	Перевірка: визначений чи ні фрейм
EXPRP	Перевірка реєстрації функції
д) *Інші*	
ATOMLIST	Складання списку одиничної глибини
PRINT	Запис у файл
INTERSECTION	Загальна частина двох списків

Управління виведенням у системі FMS здійснюється так, як це спрощено зображено на рисунку 2.11. Виконавчий механізм запускає приєднану процедуру спеціального фрейму, яка, в свою чергу, ініціює роботу механізму виведення (МВ). Коли функція MESSAGE передає повідомлення в деякий фрейм в базі знань (БЗ), то БЗ повертає в МВ деяке значення в якості результату виконання. МВ оцінює це значення, знову передає в БЗ повідомлення, але тепер вже іншому фрейму, оцінює знову отримане значення, передає нове повідомлення і т. д., повторюючи цю процедуру, він просувається від початкового значення до цільового. У цьому процесі здійснюються генерація і знищення слотів, оновлення значень слотів і т. п. Цільовим станом називається встановлення передбачуваних оціночних значень в проблемі аналітичного характеру або стан побудови моделі об'єкта для проблеми, що носить проектний характер, і т. п. В системі FMS приєднана процедура запускається при передачі повідомлення. Крім того, оскільки механізм виведення так само реалізований, як приєднана процедура, то він запускається при передачі повідомлення в спеціальний фрейм

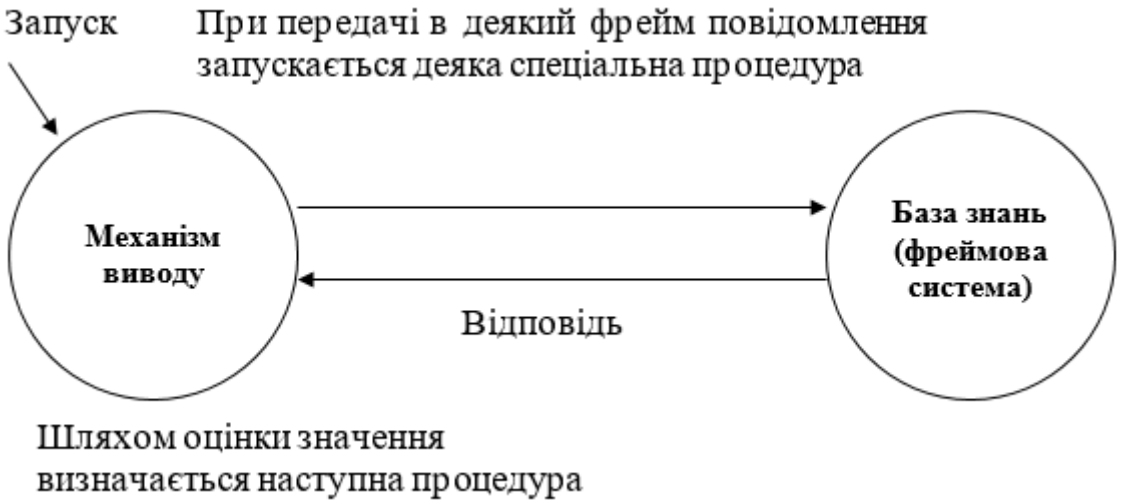
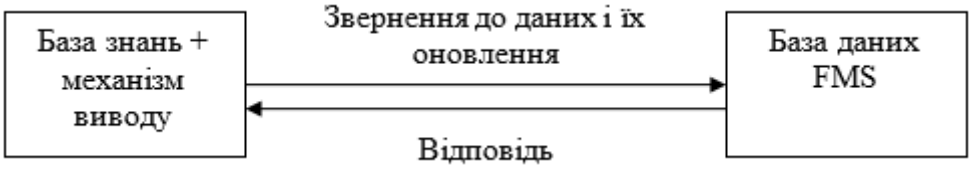


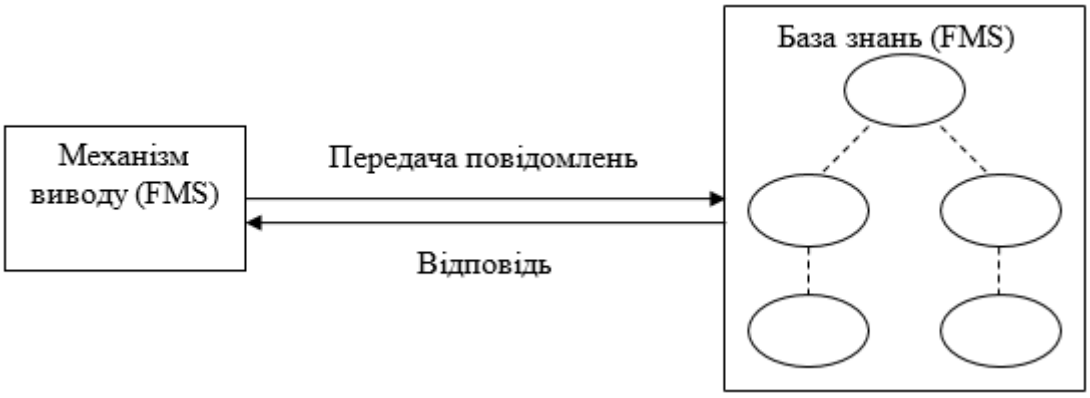
Рисунок 2.11 – Спосіб управління виводу в системі FMS

Фактично в приєднаній процедурі фрейму, що прийняв повідомлення, є функція MESSAGE, тому з неї також можлива передача повідомлення іншому фрейму, що ще більш ускладнює управління висновком.

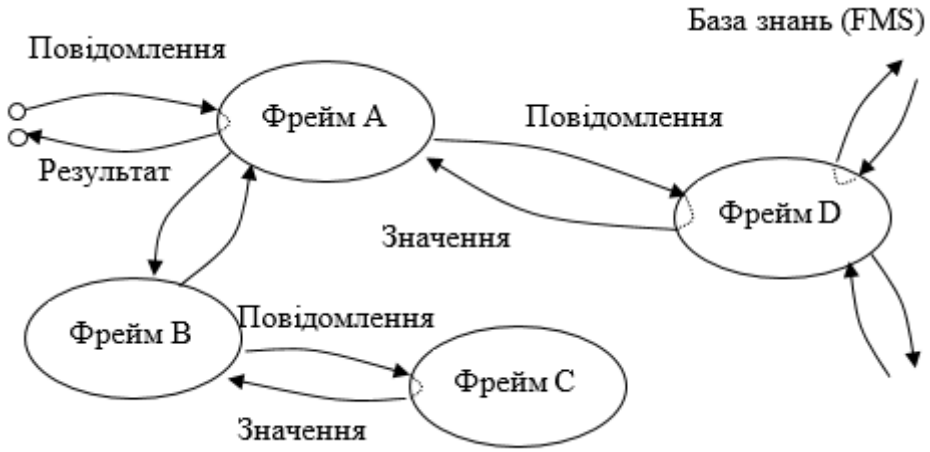
На рисунку 2.12 показані три загальноприйнятих способи управління виведенням у системі FMS. Перший спосіб (рисунок 2.12, а) використовується тільки для керування фактами. У даному випадку передбачається наявність зовнішніх бази правил і механізму виведення. Іншими словами, не допускається використання в продукційній системі (базі даних) структур даних типу фрейму.



а)



б)



в)

Рисунок 2.12 – Три зразки управління виводом в системі FMS

Другий спосіб (рисунок 2.12, б) передбачає використання фреймової системи, що містить приєднані процедури, в якості бази знань, наявність зовнішніх бази правил і механізму виведення і функціонування системи за принципом відповіді на повідомлення, передане з механізму управління виводу. Роль приєднаної процедури в цьому випадку обмежена, і демон, зокрема, виконує тільки допоміжні функції. Цей спосіб зазвичай використовується потужними універсальними продукційними системами з базою даних фреймового типу, забезпеченою механізмом успадкування та демона. Третій спосіб (рисунок 2.12, в) заснований на поступовому просуванні до мети за

допомогою почергової передачі повідомлень між фреймами. Цей спосіб відповідає найбільш високорівневому управлінню висновком, є найбільш типовим для об'єктно-орієнтованих мов. Даний спосіб дозволяє раціонально реалізувати будь-яке управління висновком, проте він вимагає досить ретельного проектування всієї системи. В іншому випадку не можна гарантувати, що система не буде переходити в режим «розносу». Крім того, як вже було сказано, в першому і в другому випадку передбачається наявність зовнішніх по відношенню до системи FMS бази правил і механізму виведення, однак частина фреймової системи може бути реалізована і всередині її. З цієї точки зору FMS може виступати в якості мови макетування систем з базами знань, і це є однією з найважливіших цілей розробки даної системи.

2.5 Висновки до розділу 2

У цьому розділі на основі теорії фреймів М. Мінського обговорено основні концепції моделі подання знань фреймами. Викладено алгоритми способів подання знань та управління висновком. Крім того, на прикладі FMS як універсальної мови представлення знань фреймами були розглянуті структура системи, представлення знань фреймами і механізм управління виводу.

3 БАЗА ЗНАНЬ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ

3.1 Структура програмної системи

Отримання знань є одним із найважчих процесів при проектуванні інтелектуальних систем. Такі системи застосовуються для розв'язання складних задач. Основна складність їх розв'язання пов'язана із використанням знань експертів, і логічне (змістовне) опрацювання інформації переважає на над обчислювальним. Тому в основі структури інтелектуальної системи лежить база знань, яка формується відповідно до предметної області, в якій застосовується система.

На рисунку 3.1 представлено основні компоненти системи.

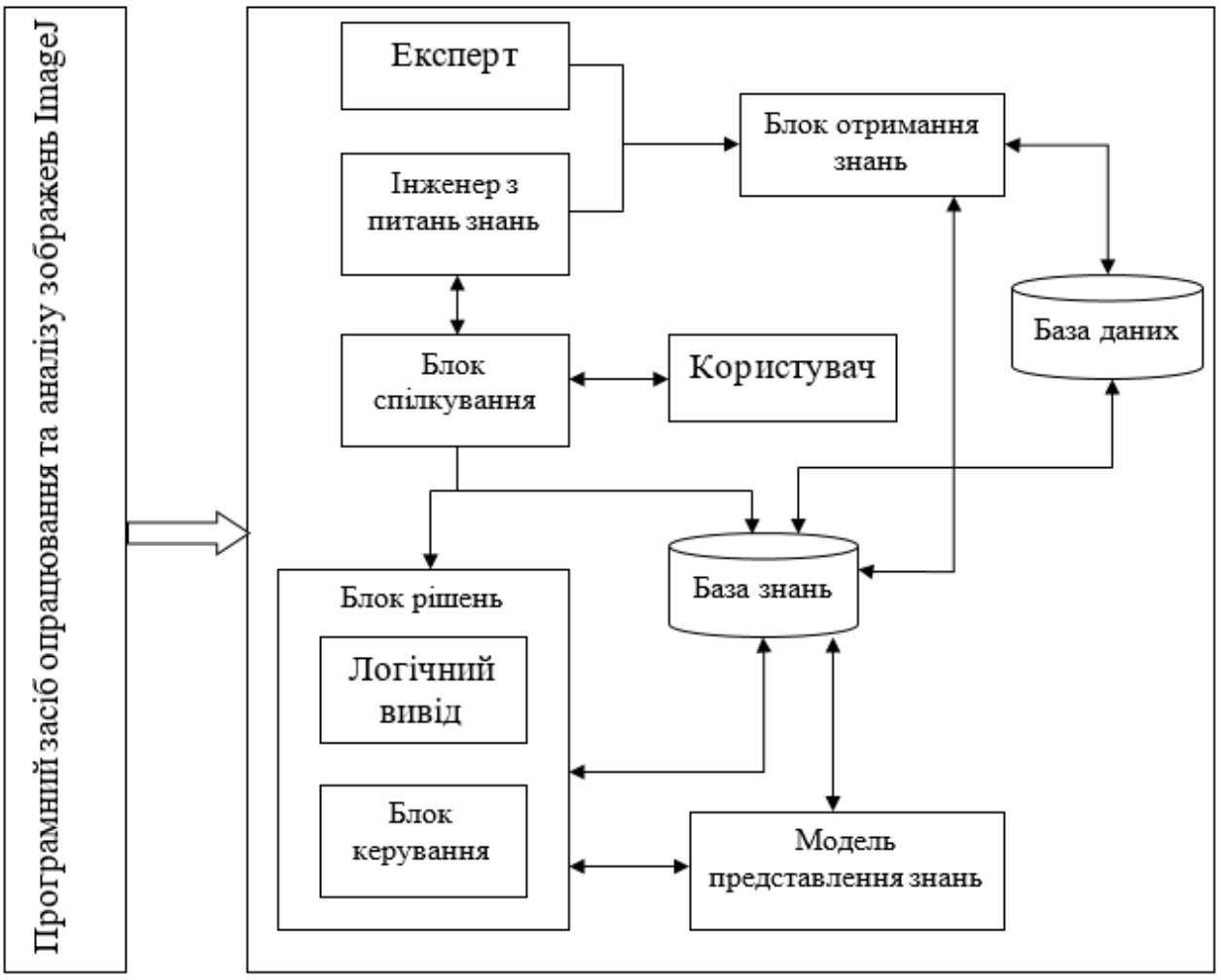


Рисунок 3.1 – Структурна схема СППР

Підсистема розроблена, як розширення (модуль) системи ImageJ, програми з відкритим кодом, призначеної для аналізу і опрацювання зображень (в тому числі і біомедичних) [34, 36-41]. ImageJ написана на мові Java і розповсюджується без ліцензійних обмежень.

Основні функціональні можливості системи:

− завантаження цитологічних зображень, що отримуються за допомогою цифрового мікроскопа та фотокамери;

− попередня обробка зображень (зменшення шуму, покращення контрасту);

− сегментація та контурний аналіз (застосування алгоритмів порогової та контурної сегментації);

− створення методик автоматичного опрацювання зображень;

− обчислення числових ознак мікрооб’єктів на цитологічних зображеннях;

− нечіткий логічний вивід діагностичних знань;

− формування звітів у форматах .xls, .csv та .arff.

Для створення програмного засобу використано програмне середовище IntellijIDEA, мови програмування Java, C++, бібліотеки OpenCV із використанням об’єктно-орієнтованого підходу [27].

Експерт – це спеціаліст в області цитології та гістології, який на основі візуальної інтерпретації зображень, встановлює взаємозв’язки та залежності між об’єктами, формує діагностичні правила на основі якісних ознак мікрооб’єктів та здійснює постановку діагнозу. Етапи отримання експертних знань наведено в таблиці 3.1.

Інженер з питань знань структурує знання, отримані від експерта та записує їх в БЗ із врахуванням правил побудови моделі представлення знань проектованої системи.

В результаті взаємодії інженера по знаннях та експерта створюється навчальна вибірка якісних ознак мікрооб’єктів цитологічних зображень.

Таблиця 3.1 – Етапи отримання експертних знань

Етапи	Характеристика
Перегляд цитологічного зображення	На цьому етапі експерт переглядає зображення та визначає чи являється воно інформативним
Оцінка зображення	Визначення поля зору, на яких є інформативні об’єкти
Виявлення якісних ознак	Експерт описує якісні (лінгвістичні) ознаки мікрооб’єктів на зображеннях
Формування діагностичних правил	На основі якісних ознак експерт формує правила для постановки діагнозу
Постановка діагнозу	На основі якісного опису ознак та сформованих діагностичних правил експерт виводить діагноз

Отже, основними функціями інженера по знаннях є: допомога експерту у виявленні та структуризації знань, формування бази знань якісних та кількісних ознак, вибір способу представлення знань, вибір методу обробки даних.

Блок отримання знань – дає змогу експерту завантажувати базу знань, а також виконувати їх редагування. Навчання системи можна звести до створення нових понять та правил на базі існуючих, а також підключення їх в базу знань таким чином, щоби не було суперечливості знань. Отже, функція цього блока полягає у формуванні емпіричних залежностей із неповних знань, тобто добуття знань першого роду на основі знань другого роду. Блок реалізовано у вигляді діалогового вікна, основними функціями якого є вибір зображення, визначення його класів, об’єктів, атрибутів, та присвоєння їм лінгвістичних значень, отриманих від експерта. Модуль дозволяє створювати нові атрибути та перелік їх можливих значень.

База знань – це набір відомостей про предметну область, для якої проектується система.

Для повного функціонування системи база знань наповнюється знаннями. Для цього запрошують висококваліфікованих експертів у галузі цитології та гістології, завдання яких – описати всі відомі знання для функціонування системи. У базі знань повинні бути наявні знання першого та другого роду. Знання першого роду – це загальновідомі факти, явища, закономірності, які визнані в предметній області й опубліковані. Знання другого роду – це набір емпіричних правил та інтуїтивних висновків, якими користуються експерти, приймаючи рішення в умовах невизначеності за наявності неповної суперечливої інформації.

В результаті експериментальних досліджень одержано інформацію про кількісні та якісні ознаки мікрооб'єктів. На основі цих даних створюється база знань. Процес створення бази знань представлено на рисунку 3.2.

Рисунок 3.2 – Етапи створення бази знань

На етапі створення БЗ на основі цитологічних зображень експериментально виділяються ракові клітини і обчислюються їхні кількісні (числові) морфометричні ознаки. Для кожного зображення експерт визначає у відповідність опис мікрооб’єктів та фону у вигляді лінгвістичних змінних на основі чого будуються функції належності. Враховуючи, що для опису одного об'єкта може використовуватись декілька кількісних ознак, то проводиться створення правил нечіткого логічного виводу на основі декількох лінгвістичних змінних.

Блок логічного виводу призначений для виконння дій, аналогічних до інтелектуальної діяльності експерта, під час прийняття рішенн. Основною функцією даного блоку є побудова логічного висновку на базі існуючих знань, які зберігаються в БЗ.

Блок керування контролює процес пошуку рішення, тобто визначає послідовність використання різних правил і процедур маніпулювання знаннями.

Модель представлення знань. Тут застосовується фреймова модель даних. Одиницею зображення у фреймовій моделі є фрейм (об'єкт). Він є формою зображення певної ситуації, яку можна описувати сукупністю понять і сутностей.

У різних галузях науки використовується поняття абстрактного зображення. Наприклад, слово «клітина» породжує зображення клітини: «ядро, цитоплазма, хроматин, вакуолі». В цьому описі є «гнізда» – невизначені значення певних атрибутів, наприклад, кількість ядер, площа цитоплазми. У теорії фреймів таке зображення кімнати називається фреймом кімнати.

Розрізняють фрейми-зразки (прототипи, що є описами) і фрейми-екземпляри (конкретні значення прототипів).

Приклад фреймової моделі представлено на рисунку 3.3.

У фреймі сутності предметної області зображуються у вигляді пар «атрибут-значення», або так званих слотів. Слоти, у свою чергу, можуть містити фасети, або обмеження на множину допустимих значень.

Потужність фреймової моделі обумовлена тим, що в ній використовується властивість успадкування фреймів.

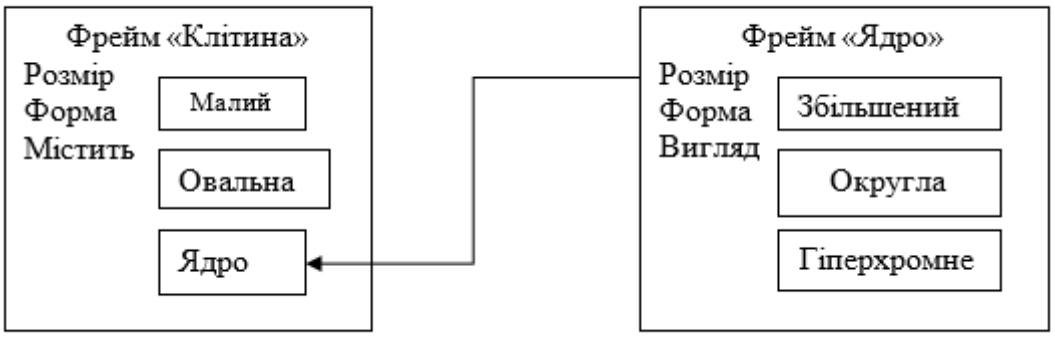


Рисунок 3.3 – Приклад частини фрейму «Клітина»

Блок спілкування з користувачем або інтерфейс користувача необхідний для організації діалогової взаємодії між системою і користувачем. Основна вимога до цього блоку – це реалізація спілкування природною мовою користувача. Він дозволяє переглянути проведені досліди, розпочати нове дослідження, вивести діагностичні знання, запустити модель представлення знань, переглянути методики опрацювання зображень, переглянути базу даних.

Студія розробника Microsoft Developer Studio – це інтегроване середовище для розробки, яке дозволяє функціонувати різним середовищам розробки, одним із яких є Visual C++ [42-46]. В студії розробника можна будувати звичайні програми на C та C++, створювати статичні та динамічні бібліотеки, Windows-програм на основі бібліотеки базових класів MFC (Microsoft Foundation Class Library). В зв’язку з тим, що сьогодні рівень складності програмного забезпечення дуже високий, розробка інформаційних технологій з використанням тільки певної мови програмування майже неможлива. Потрібно витрачати багато ресурсів на вирішення стандартних

задач по створенню багатовіконного інтерфейсу, налаштування програми, виправлення помилок. Щоб спростити завдання, майже всі сучасні C++ компілятори містять спеціальні бібліотеки класів. Ці бібліотеки включають майже весь програмний інтерфейс Windows і дозволяють використовувати більш сучасні інструменти програмування, ніж традиційні виклики функцій. В результаті значно спрощується розробка програм зі складним інтерфейсом користувача, спрощується взаємодія з базами даних, а значить, скорочується час розробки програмних інструментів.

3.2 Модуль фреймової моделі представлення знань

Для програмної реалізації використано об’єктно-орієнтований підхід. Окремі сутності предметної області представлено у вигляді класів, взаємодія між ними проходить визначеним способом через інтерфейс класу і технологію наслідування. Діаграму класів підсистеми наведено на рисунку 3.4.

Клас FileUtil містить допоміжні функції для роботи із файлами: fileXml – файл для завантаження, file – файл для збереження функції, prologFunction – функція для збереження, fileFrom – поточна назва файлу, fileTo – нова назва файлую.

У класі PrologUtil реалізовано наступні статичні методи: prologIDE - екземпляр класу середовища розробки на мові Пролог, mapValue – факти та їх значення, builder – додавання стрічки сформованої на Пролог.

Клас AbstractDialogMain – абстрактний клас основного вікна моделі представлення знань. Реалізований на платформі Java як додаток (модуль) програми ImageJ.

Клас AbstractFunctionEditor – абстрактний клас панелі редагування функції. Він містить наступні методи: prologFunction – функція Прологу, executeModel() – запуск моделі, createPrologFunction() – створення функції

Прологу в редакторі, testFunction() – тестування функції в середовищі Пролог, showExample() – приклад коду для моделі.

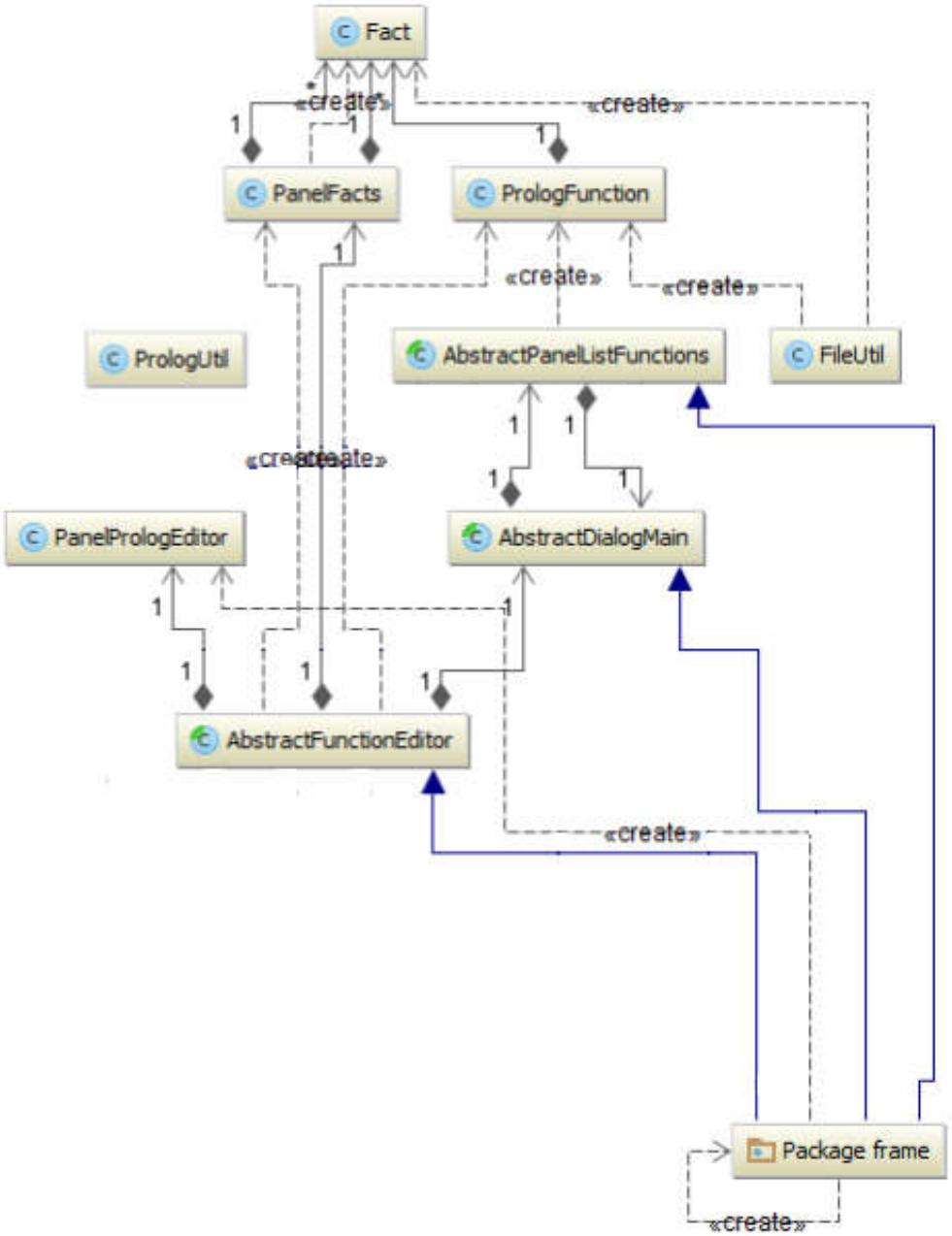


Рисунок 3.4 – Діаграма класів (фреймова модель представлення знань)

```
     * @param prologIDE екземпляр класу середовища розробки на мові
Пролог
     * @return текстове поле стартової функції з переданого prologIDE
     */
    public static JTextField getGoalFild(JavaIDE prologIDE);

    /**
     * @param mapValues факти та їх значення
     * @param builder  сюди додається сформована стрічка у Пролог
синтаксі
     * @return побудована строка по builder
     */
    public static String appendFacts(Map<Fact, Double> mapValues,
StringBuilder builder);
    /**
     * @param mapResult факти та їх значення
     * @return сформована строка у Пролог синтаксі
     */
    public static String getFacts(Map<Fact, Double> mapResult);
    /**
     * @return назва функції прологу з якої починається інтерпретування
     */
    public static String getStartGoal();
    /**
     * fix close operation and set window bounds to save
     * @return екземпляр середовища розробки для мови Пролог
     */
    public static JavaIDE createPrologIDE();
```

Клас AbstractPanelListFunctions є абстрактним класом панелі із списком функцій. Він містить наступні методи: getDirectory() – директорія із файлами функції моделі, saveSelectedFunction() – збереження вибраної функції, addFunction() – добавлення функції, enameSelectedFunction() – зміна назви вибраної функції, removeSelectedFunction() – видалення вибраної функції.

Для створення панелі редагування функції необхідно використати наступні методи: prologFunction – функції Прологу, functionName – назва функції, abstractDialogMain – основне вікно моделі.

```
package Diagnosis.knowledge_representation_models.prolog;

import Diagnosis.CPackage.CMessage;
import Diagnosis.CPackage.MyScroll;
import Diagnosis.CPackage.components.CJList;
import Diagnosis.knowledge_representation_models.GeneralUtil;

import alice.tuprolog.Prolog;
import alice.tuprolog.Theory;

import javax.swing.*;
import javax.swing.border.TitledBorder;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.FileFilter;
import java.util.Arrays;
import java.util.Set;
```

```
/**
 * Абстрактний клас панелі із списком функцій
 */
public abstract class AbstractPanelListFunctions extends JPanel implements ActionListener {
    protected final ListFunctions listFunctions;
    protected final JButton btnAddFunction = new JButton("Добавити");
    protected final JButton btnSave = new JButton("Зберегти");
    protected final JButton btnRename = new JButton("Переіменувати");
    protected final JButton btnRemove = new JButton("Видалити");
    protected final AbstractDialogMain abstractDialogMain;
    {
        this.btnAddFunction.addActionListener(this);
        this.btnSave.addActionListener(this);
        this.btnRename.addActionListener(this);
        this.btnRemove.addActionListener(this);
    }

    public AbstractPanelListFunctions(AbstractDialogMain abstractDialogMain) {
        this.abstractDialogMain = abstractDialogMain;
        this.listFunctions = new ListFunctions();
        this.setBorder(new TitledBorder(null, "Список функцій", TitledBorder.CENTER, TitledBorder.TOP));
        this.setLayout(new BorderLayout(2, 2));
        MyScroll myScroll = new MyScroll(listFunctions, 50, 50);
        myScroll.setBorder(null);
        this.add(myScroll, BorderLayout.CENTER);
        JPanel pnl = new JPanel(new GridLayout(0, 2, 2, 2));
```

```
        pnl.add(btnAddFunction);
        pnl.add(btnSave);
        pnl.add(btnRename);
        pnl.add(btnRemove);
        this.add(pnl, BorderLayout.SOUTH);
    }

    /**
     * @return директорія із файлами функцій моделі
     */
    public abstract String getDirectory();

    /**
     * Зберегти вибрану функцію
     */
    void saveSelectedFunction() {
        File file = listFunctions.getSelectedValue();
        if (file == null) {
            return;
        }

        try {
            Set<Fact> facts =
abstractDialogMain.getFunctionContainer().getPanelEditFunctionMain().getFacts();
            String prologText =
abstractDialogMain.getFunctionContainer().getPanelEditFunctionMain().getPrologE
ditorText();
            // check for error
            Prolog prolog = new Prolog();
```

```
                prolog.setTheory(new
Theory(abstractDialogMain.getKnowledgeRepresentationModelBody() + "\n\n" +
prologText));
                // prolog.addTheory(new Theory());
                // saveSelectedFunction
                PrologFunction prologFunction = new
PrologFunction(getFunctionName(file), prologText, facts);
                FileUtil.saveFunction(file, prologFunction);
            }
            catch (Exception ex) {
                //  ex.printStackTrace();
                CMessage.showError(abstractDialogMain, ex.getMessage());
            }
        }

        /**
         * Додати функцію
         */
        void addFunction() {
            String functionFileName =
JOptionPane.showInputDialog(abstractDialogMain, "Введіть назву функції");
            if (functionFileName != null && !functionFileName.trim().isEmpty()) {
                String error = GeneralUtil.checkFunctionName(functionFileName);
                functionFileName = functionFileName.trim();
                if (error.isEmpty()) {
                    String funcName = functionFileName;
                    functionFileName += FileUtil.getFileExtension();
                    for (File file :listFunctions.getData()) {
                        if (file.getName().equals(functionFileName)) {
```

```
                        CMessage.showError(abstractDialogMain, "Функція з назвою
\""+functionFileName+"\" уже існує");
                        return;
                    }
                }
                File newFunctionFile = new File(getDirectory() +
functionFileName);
                error = FileUtil.addFunction(newFunctionFile);
                if (!error.isEmpty()) {
                    CMessage.showError(abstractDialogMain, error);
                    return;
                }
                listFunctions.addElement(newFunctionFile);
            }
            else {
                CMessage.showError(abstractDialogMain, error);
            }

        }
    }

    /**
     * Змінити назву вибраної функції
     */
    void renameSelectedFunction() {
        File file = listFunctions.getSelectedValue();
        if (file != null) {
```

```
String newFunctionName =
JOptionPane.showInputDialog(abstractDialogMain, "Введіть нову назву функції");
if (newFunctionName != null &&
!newFunctionName.trim().isEmpty()) {
String error = GeneralUtil.checkFunctionName(newFunctionName);
newFunctionName = newFunctionName.trim();
if (error.isEmpty()) {
newFunctionName += FileUtil.getFileExtension();
for (File f :listFunctions.getData()) {
if (f.getName().equals(newFunctionName)) {
ChaosMessage.showError(abstractDialogMain, "Функція з
назвою \""+newFunctionName+"\" уже існує");
return;
}
}
File newfile = new File(getDirectory() + newFunctionName);
error += FileUtil.renameFunction(file, newfile);

if (!error.isEmpty()) {
CMessage.showError(abstractDialogMain, error);
return;
}

this.listFunctions.addElement(newfile);
// remove old file
this.listFunctions.removeElement(file);
}
catch (Exception ex) {
ex.printStackTrace();
```

```
                }
            }
            else {
                CMessage.showError(abstractDialogMain, error);
            }

        }
    }
}

/**
 * Видалити функцію
 * @param file файл функцію
 * @param showConfirmDialog показати вікно підтвердження
 */
void removeFunction(File file, boolean showConfirmDialog) {
    if (file != null) {
        if (showConfirmDialog) {
            int result = JOptionPane.showConfirmDialog(abstractDialogMain,
String.format(
                    "<html>Ви точно хочете видалити функцію
«<b>%s</b>»</html>", file.getName()),
                    "Підтвердження видалення функції",
JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE
            );
            if (result != JOptionPane.YES_OPTION) {
                return;
            }
        }
```

```
            String error = FileUtil.deleteProductionalFunction(file);
            if (error.isEmpty()) {
                this.listFunctions.removeElement(file);
            }
            else {
                CMessage.showError(abstractDialogMain, error);
            }
        }
    }

    /**
     * Видалити вибрану функцію
     */
    void removeSelectedFunction() {
        File file = this.listFunctions.getSelectedValue();
        this.removeFunction(file, true);
    }
```

```
        @Override
        public void actionPerformed(ActionEvent e) {
            if (e.getSource() == this.btnAddFunction) {
                this.addFunction();
            }
            else if (e.getSource() == this.btnRename) {
                this.renameSelectedFunction();
            }
            else if (e.getSource() == this.btnRemove) {
                this.removeSelectedFunction();
            }
            else if (e.getSource() == this.btnSave) {
                    this.saveSelectedFunction();
            }
        }

        class ListFunctions extends ChaosJList<File> implements
ListSelectionListener {
            private PrologFunction selectedPrologFunction = null;

            {
                this.setCellRenderer(new DefaultListCellRenderer(){
                    @Override
                    public Component getListCellRendererComponent(JList<?> list,
 Object value, int index, boolean isSelected, boolean cellHasFocus) {
                        JLabel lbl = (JLabel) super.getListCellRendererComponent(list,
 value, index, isSelected, cellHasFocus);    //To change body of overridden methods
 use File | Settings | File Templates.
                        File file = (File) value;
```

```
                if (file != null) {
                    lbl.setText(file.getName().substring(0, file.getName().length() -
FileUtil.getFileExtension().length()));
                }
                return lbl;
            }
        });
        this.addListSelectionListener(this);
        File[] files = new File(getDirectory()).listFiles(new FileFilter() {
            @Override
            public boolean accept(File pathname) {
                return pathname.isFile() &&
pathname.toString().toLowerCase().endsWith(FileUtil.getFileExtension());
            }
        });
        this.setData(Arrays.asList(files));
    }

    ListFunctions() {

    }
    public PrologFunction loadFunction(File file) {
        if (file != null) {

            try {
                PrologFunction prologFunction = FileUtil.loadFunction(file);
                return prologFunction;
            }
```

```
                catch (Exception ex) {
                  //  ex.printStackTrace();
                    String message = "Не вдається завантажити файл
\""+file.getName()+"\" по причині:\n" + ex.getMessage();
                    JOptionPane.showMessageDialog(abstractDialogMain, message);
                }
            }
            return null;
        }

        PrologFunction getSelectedPrologFunction() {
            return this.selectedPrologFunction;
        }

        @Override
        public void valueChanged(ListSelectionEvent e) {
            if (e.getSource() == this && !e.getValueIsAdjusting()) {
                File file = this.getSelectedValue();
                if (file == null) {
                    this.selectedPrologFunction = null;
```

```
abstractDialogMain.getFunctionContainer().setPanelEditFunctionMain(null);
                    }
                    else {
                        int selected = this.getSelectedIndex();
                        int previous = selected == e.getFirstIndex() ? e.getLastIndex() :
e.getFirstIndex();

                        this.selectedPrologFunction = this.loadFunction(file);
                        if (this.selectedPrologFunction == null) {

abstractDialogMain.getFunctionContainer().setPanelEditFunctionMain(null);
                        }
                        else {
                            String functionName = getFunctionName(file);

abstractDialogMain.getFunctionContainer().setPanelEditFunctionMain(createPanelF
unctionEditor(selectedPrologFunction, functionName,
                                    abstractDialogMain));
                        }

                    }

                }
            }
        }

        /**
         * Створення панелі для редагування функції прологу
         * @param prologFunction функція прологу
```

```
         * @param functionName назва функції
         * @param abstractDialogMain основне вікно моделі
         * @return створення панель для редагування функції прологу
         */
        protected abstract AbstractFunctionEditor
createPanelFunctionEditor(PrologFunction prologFunction, String functionName,
AbstractDialogMain abstractDialogMain);

        /**
          * @param f файл функції
          * @return назва функції по файлу
          */
         String getFunctionName(File f) {
            return f.getName().substring(0, f.getName().length() -
 FileUtil.getFileExtension().length());
        }
    }
```

Клас PanelFacts це панель для редагування фактів (додання, видалення, вибір статистичної функції).

Клас PanelPrologEditor це панель для редагування тексту на мові пролог та підсвітки його синтаксису.

Клас PrologFunction представляє функцію прологу.

Для створення системи було використано, мови програмування Java, C++, мову логічного програмування Prolog, бібліотеки OpenCV. Процедуру представлення було створено як підсистему програмної системи ImageJ.

OpenCV – бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення [47]. Основні бібліотеки: opencv_ml –модель машинного навчання (дерево рішень, SVM), opencv_flann – швидкий пошук найближчих сусідів, opencv_imgproc – опрацювання зображень (фільтрація, геометричне перетворення).

3.3 Тестування та верифікація програми

Підсистема під ОС Windows. Модуль розроблено на платформі Java [48] і здійснено підключення фреймової моделі представлення знань реалізованої мовою логічного програмування Пролог.

Пролог – мова логічного програмування, основними поняттями в якій є факти, правила логічного висновку і запити, що дозволяють описувати бази знань, процедури логічного висновку та прийняття рішень. Особливу роль у інтерпретаторі Прологу відіграють конкретні запити до бази знань, на які система генерує відповіді «істина» і «хибність». Для узагальнених запитів із змінними в якості аргументів створена система Пролог виводить конкретні дані на підтвердження істинності узагальнених відомостей і правил виводу [43].

Системні вимоги до апаратного забезпечення, яке необхідне для коректної роботи системи наведено в таблиці 3.2.

Для запуску програми необхідно запустити служби Apache та MySQL, щоби отримати доступ до бази даних (рисунок 3.5).

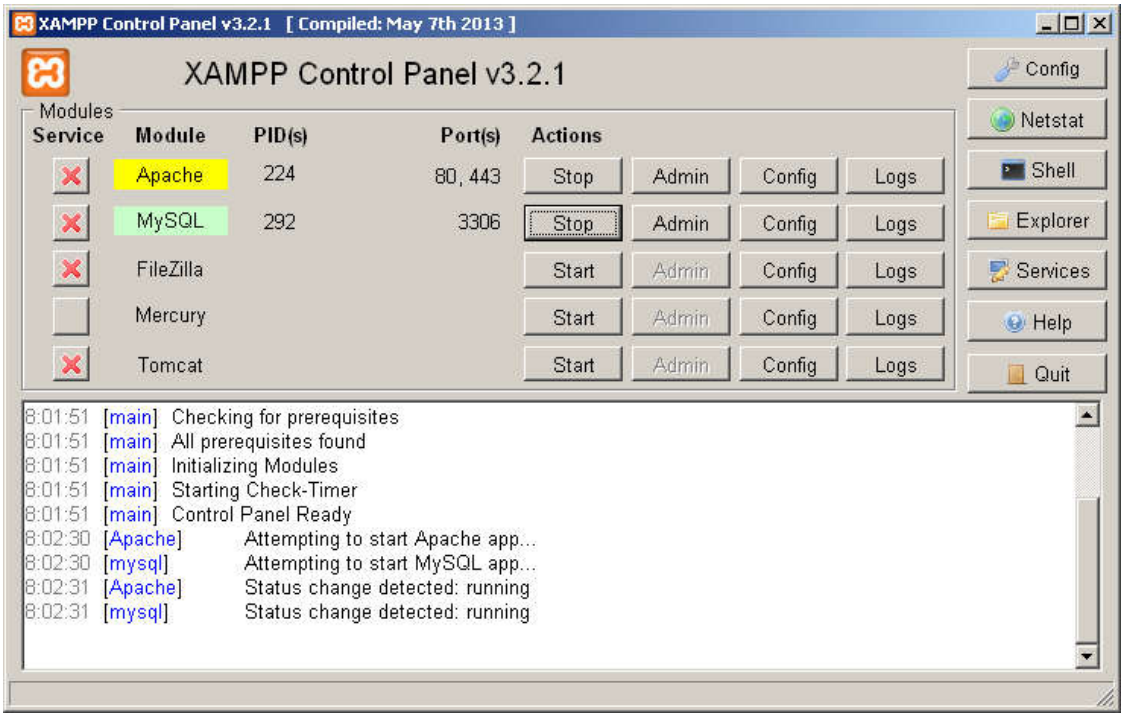


Рисунок 3.5 – Запуск служб Apache та MySQL

Таблиця 3.2 – Мінімальні вимоги до апаратного забезпечення

Характеристика	Параметри
Процесор	1200 Гц
Оперативна пам'ять	2048 Мб
Обсяг вільного дискового простору	150 Мб
Монітор	1024x768
ОС	Windows
Периферійні пристрої	Принтер (при необхідності)
Додаткові пристрої	CD, DVD-ROM

Запустивши файл ij.jar відкривається стартове вікно програми, яке представлене на рисунку 3.6.

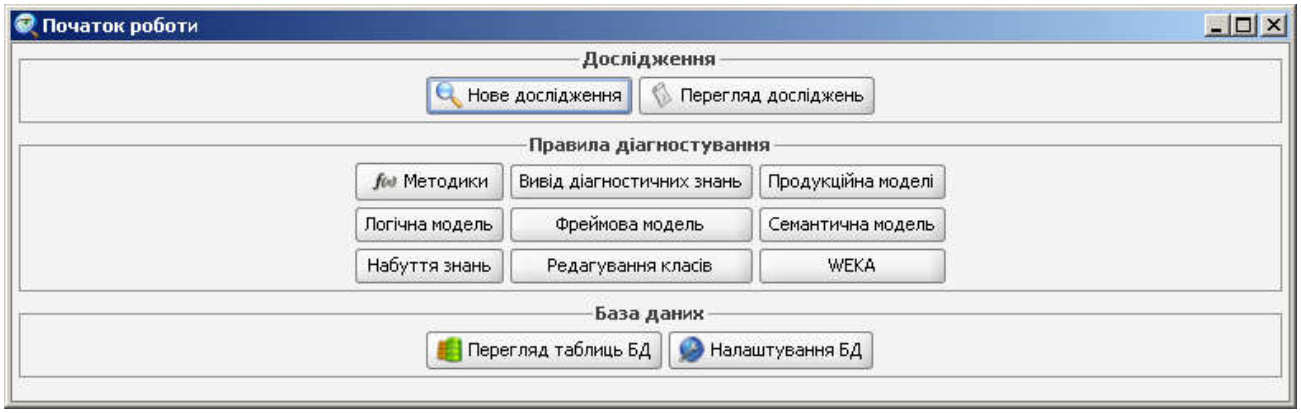


Рисунок 3.6 – Стартове вікно роботи системи

Основними важливими компонентами вікна є:

1. «Нове дослідження». Основний функціонал – створення нового експерименту, в якому запускається запис пацієнта, шлях до папки, в якій зберігаються зображення, вибір методу діагностики, визначення класів, за якими проводитиметься дослідження та вибір ознак (рис. 3.7), які будуть розраховані (площа, периметр) , мода, медіана, ядерно-цитоплазматичне співвідношення) Цей блок використовується для створення дослідження, в

якому будуть виявлені мікрооб'єкти та будуть визначені їх кількісні (числові характеристики).

2. «Перегляд дослідження» – дозволяє переглянути проведені дослідження, числові ознаки мікрооб’єктів (рисунок 3.8). Також цей компонент призначений для формування звіту по досліду у форматах .xls, .arff, .csv із можливістю вибору необхідних ознак (рисунок 3.9).

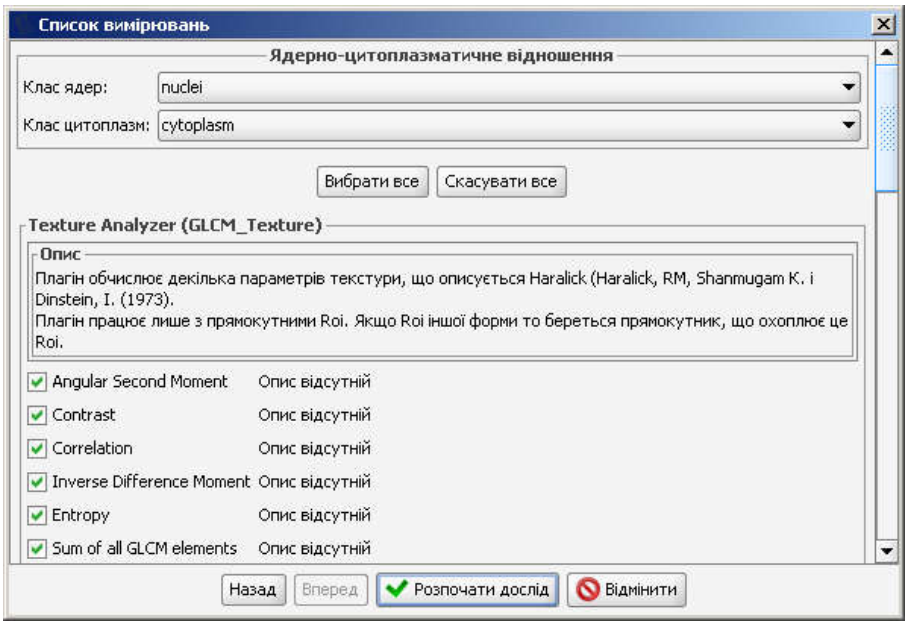


Рисунок 3.7 – Перелік числових ознак для обчислення

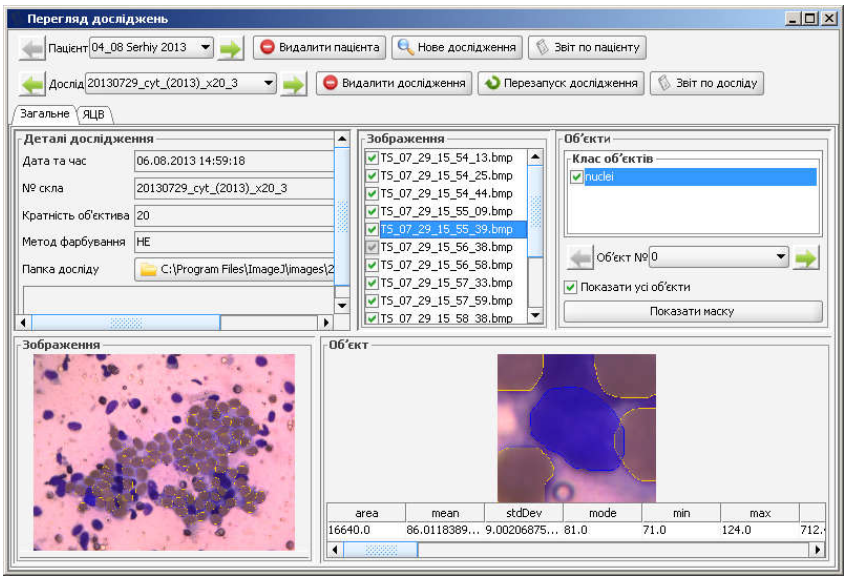


Рисунок 3.8 – Вікно перегляду проведеного дослідження

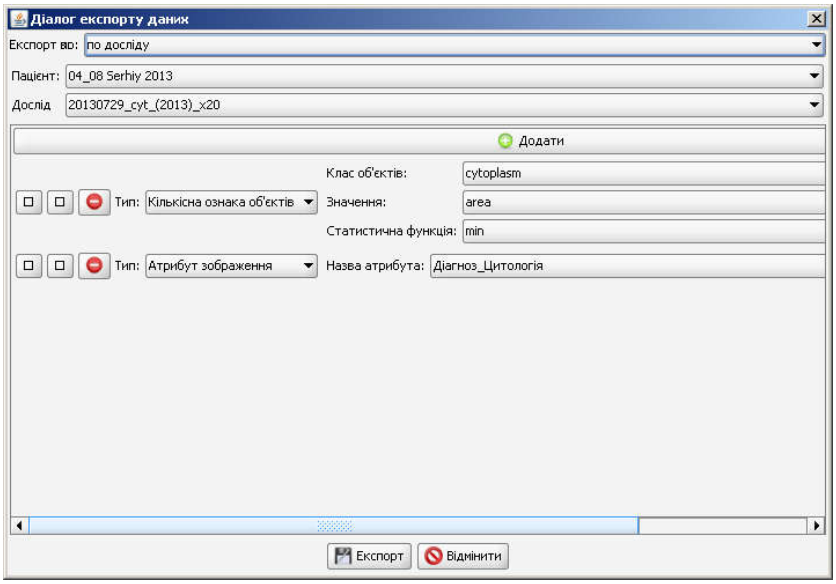


Рисунок 3.9 – Вибір ознак для експорту у файл

3. "Отримання знань" - цей інтерфейс призначений для роботи з експертом, його призначення - присвоєння якісних характеристик мікрооб'єктів на цитологічних зображеннях. Функції інтерфейсу дозволяють вибрати пацієнта, експеримент, зображення, клас об'єкта, надати значення атрибутам об'єктів, додати або опублікувати атрибути та список їх можливих значень (рисунок 3.10).

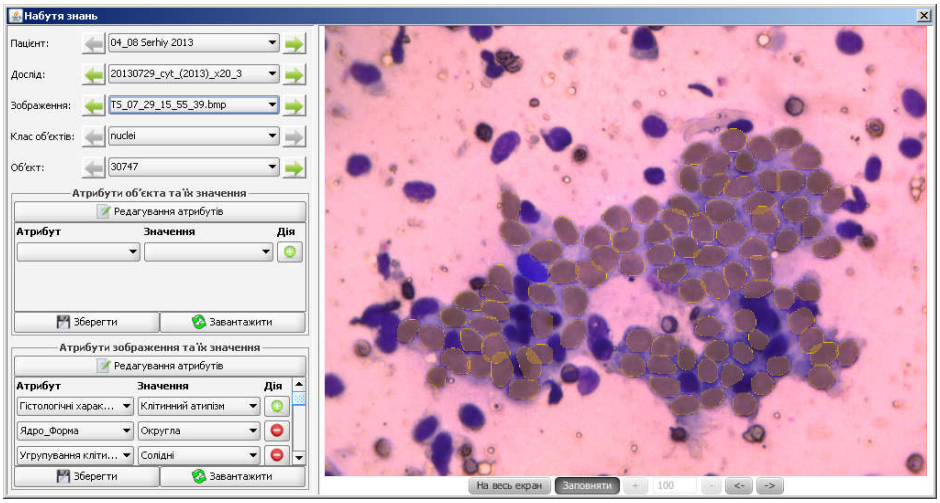


Рисунок 3.10 – Вікно для присвоєння якісних ознак мікрооб’єктів

Закінчивши етапи обчислення кількісних та опису якісних ознак мікрооб’єктів формуються діагностичні правила і вибирається модель представлення знань.

На рисунку 3.11 представлено вікно фреймової моделі представлення знань.

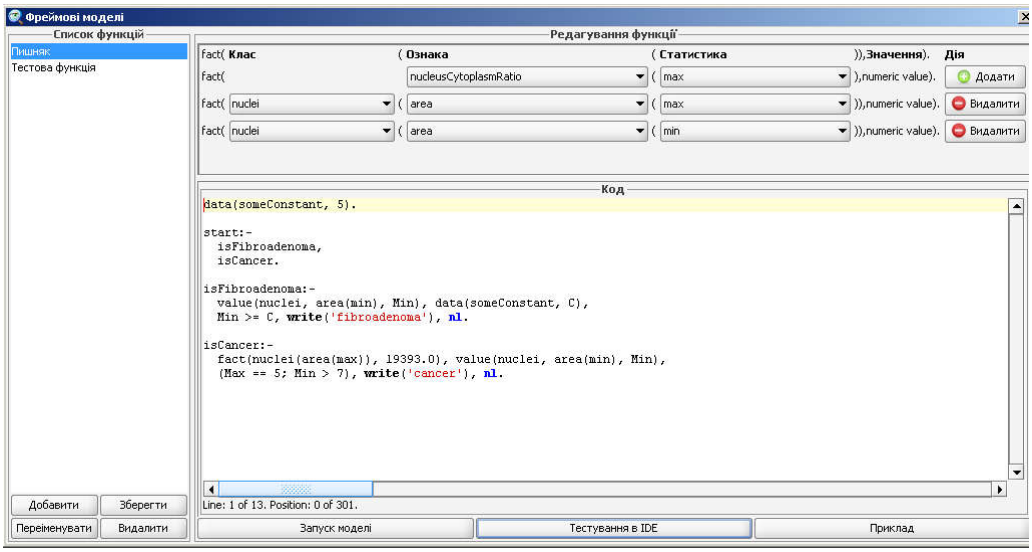


Рисунок 3.11 – Вікно фреймової моделі представлення знань

Дане вікно містить містить список створених функцій, які можна додавати, видаляти, переіменовувати, зберігати. Також можна редагувати функції, вибираючи класи мікрооб’єктів, ознаки та типи статистичних функції. У полі «Код» записуються діагностичні правила. Вони автоматично зберігаються у програмний код, який можна перегляну тинатиснувши кнопку «Тестування IDE» (рисунок 3.12).

На рисунку 3.12 представлено графічний інтерфейс редагування програмного коду фреймової моделі у середовищі tuProlog. Тіло фреймової моделі представлено у лістингу 3.1.

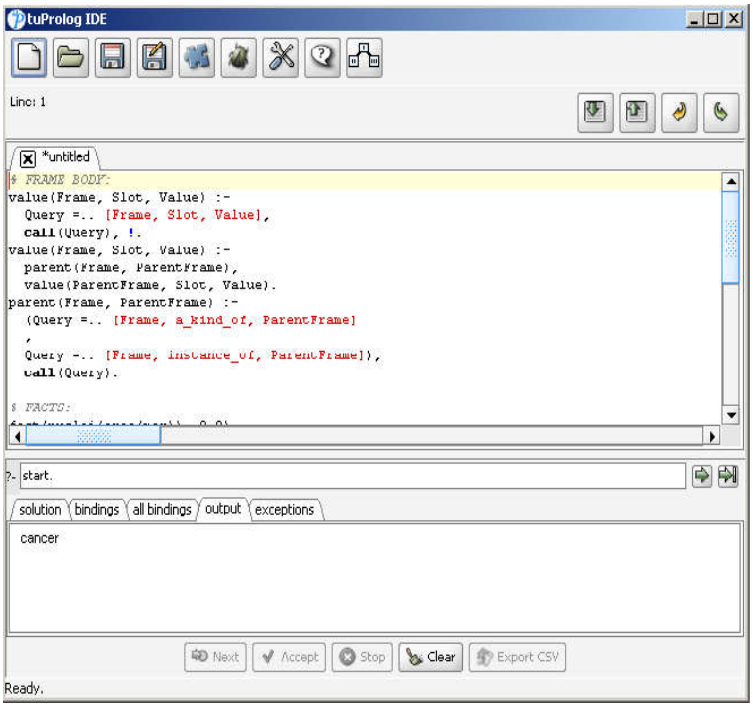


Рисунок 3.12 – Вікно редагування коду програми Пролог

Лістинг 3.1 – Тіло фреймової моделі

```
value(Frame, Slot, Value) :-
  Query =.. [Frame, Slot, Value],
  call(Query), !.
value(Frame, Slot, Value) :-
  parent(Frame, ParentFrame),
  value(ParentFrame, Slot, Value).
parent(Frame, ParentFrame) :-
  (Query =.. [Frame, a_kind_of, ParentFrame]
  ;
  Query =.. [Frame, instance_of, ParentFrame]),
  call(Query).
```

Натиснувши кнопку «Запуск моделі» можна здійснити вибір проведення дослідження (рисунок 3.13), переглянути статистику по дослідженню (рисунок 3.14) та текстовий вивід результату (рисунок 3.15).

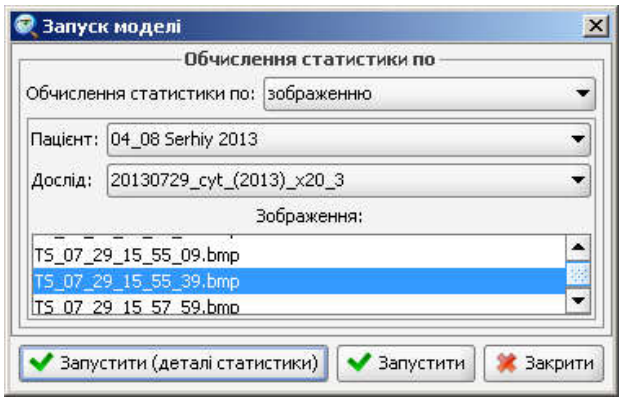


Рисунок 3.13 – Вибір об’єкта, за яким обчислюватиметься статистика

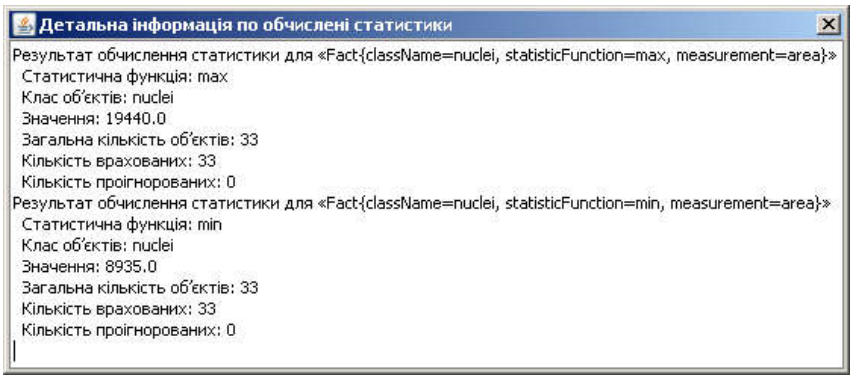


Рисунок 3.14 – Вивід статистичної інформації

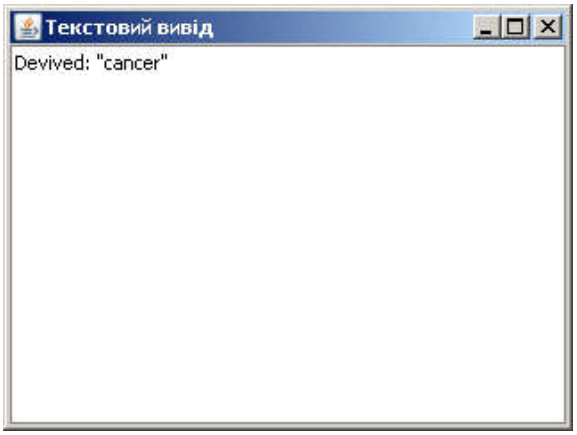


Рисунок 3.15 – Текстовий вивід діагнозу

Програма аналізує дані, на основі яких робить висновки про діагноз.

Фреймова модель має здатність описувати мета знання та формувати на їх основі нові знання, володіє гнучкістю та наглядністю при формалізації експертних знань. Має внутрішню структуру зв’язку, механізм наслідування властивостей та можливість переходу до мережевої моделі. Моделі є ефективні для структурного опису складних понять, проте потребують значних затрат для збереження залежностей між знаннями.

3.4 Висновки до розділу 3

В даному розділі зроблено проектування інтелектуальної системи, представлено основні компоненти системи, основні функціональні можливості системи, етапи отримання експертних знань. Програмно реалізована фреймова модель представлення знань, проведено тестування та верифікацію програми.

ВИСНОВКИ

В результаті виконання дипломної роботи отримано такі результати:

1. Проведено класифікацію типів знань.
2. Проаналізовано моделі представлення знань.
3. Формалізовано базу знань імуногістохімічних зображень на основі фреймової моделі.
4. Представлено основні компоненти системи, основні функціональні можливості системи, етапи отримання експертних знань.
5. Програмно реалізована фреймова модель представлення знань та проведено тестування та верифікацію системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Автандилов Г. Г. Основы количественной патологической анатомии. / Г. Г. Автандилов – М.: Медицина, 2002. – 240 с.

2. Мельник А. Н. Цитоморфологическая диагностика опухолей : монографія. К. : Здоров'я, 1983. 240 с.

3. Мачуляк М. В., Галан В. Ю., Іпіроті В. О., Николин І. П. Системи підтримки прийняття рішень в медичній діагностиці. *Інтелектуальні комп'ютерні системи та мережі* : тези доп. V Наук.-практ. конф. молодих вчених і студентів (2 груд. 2021 р.). Тернопіль : ЗУНУ, 2021. С. 12.

4. Галан В. Ю., Мачуляк М. В., Іпіроті В. О., Николин І. П. Моделювання знань в системах медичної діагностики. *Інтелектуальні комп'ютерні системи та мережі* : тези доп. V Наук.-практ. конф. молодих вчених і студентів (2 груд. 2021 р.). Тернопіль : ЗУНУ, 2021. С. 13.

5. Березький О. М., Дубчак Л. О., Мельник Г. М. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня "Магістр". Спеціальність: 123 - Комп'ютерна інженерія. Магістерська програма – «Комп'ютерна інженерія». Тернопіль : ЗУНУ, 2021. 32 с.

6. Представление и использование знаний / под ред. Х. Уэно, М. Исидзука; пер. с япон. к. т. н. И. А. Иванова. – М.: Мир, 1989. – 220 с.

7. Искусственный интеллект: в 3-х кн. Кн. 2. Модели и методы: справочник; под ред. Д. А. Поспелова. – М.: Радио и связь, 1990. – 304 с.

8. Gallaire H. Logic and data bases / H. Gallaire, J. Minker (eds.). – N. Y.: Plenum Press, 1978. – 458 p.

9. Гаврилова Т. А. Базы знаний интеллектуальных систем / Т. А. Гаврилова, В. Ф. Хорошевский. – СПб: Питер, 2000. – 384 с.

10.Хорошевский В. Ф. Управление проектами в интеллектуальной системе PIES Workbench / В. Ф. Хорошевский // Изв. РАН Серия «Техническая кибернетика». – 1993. – №5. – С. 71-98.

11.Ковригин О. В. Гибридные средства представления знаний в системе СПЭИС / О. В. Ковригин, К. Г. Перфильев // Тез. докл. Всесоюзной конференции по искусственному интеллекту, Переславль-Залесский, 1988. – Т. 2. – С. 490-494.

12.Попов Э. В. Динамические интеллектуальные системы в управлении и моделировании / Э. В. Попов. – М. МИФИ, 1996.

13.Скрегг Г. Семантические сети как модели памяти / Г. Скрегг // Новое в зарубежной лингвистике. Вып. 12. – М.: Радуга, 1983. – С. 228-271.

14.Цейтин Г. С. Программирование на ассоциативных сетях / Г. С. Цейтин // ЭВМв проектировании и производстве. Вып. 2. – Л.: Машиностроение, 1985. – С. 16-48.

15.Осипов Г. С. Приобретение знаний интеллектуальными системами / Г. С. Осипов. – М.: Наука, 1997.

16.Осипов Г.С. Лекции по искусственному интеллекту. – М.: Книжный дом «ЛИБРОКОМ», 2013. – 272 с.

17.Построение экспертных систем / под ред. Ф. Хейес-Роти, Д. Уотермена, Д. Лената. – М.: Мир, 1987.

18.Durkin J. Expert Systems: Catalog of Applications. – ICS, USA, 1998.

19.Таунсенд К. Проектирование и программная реализация экспертных систем на персональных ЭВМ / К. Таунсенд, Д. Фохт; пер. англ. – М.: Финансы и статистика, 1990. – 320 с.

20.Минский М. Фреймы для представления знаний / М. Минский. – М.: Энергия, 1979. – 205 с.

21.Шенк Р. Познать механизмы мышления / Р. Шенк, Л. Хантер. – М.: Мир, 1987.

22.Байдун В. В. Средства представления и обработки знаний в системе FRL/PS / В. В. Байдун, А. И. Бунин // Тез. докл. Всесоюзной конференции по искусственному интеллекту, Минск, 1990. – Т. 1. – С. 66-71.

23.Уотермен Д. Руководство по экспертным системам / Д. Уотермен; пер. с англ. – М.: Мир, 1989.

24.Стрельников Ю. Н. Разработка экспертных систем средствами инструментальной оболочки в среде MS Windows / Ю. Н. Стрельников, Н. А. Борисов – Тверь, ТГТУ, 1997.

25.Достоверный и правдоподобный вывод в интеллектуальных системах / [Вагин В.Н., Головина Е.Ю., Загорянская А.А., Фомина М.В.]. – М.: ФИЗМАТЛИТ, 2004. – 704 с.

26.Рак молочної залози. Діагностика та лікування. URL: https://www.pfizermed.com.ua/public/medical-content/рак-молочної-залози-діагностика-та-лікування/2741.

27.Рак молочної залози. URL: https://www.onco.cv.ua/rak-molochnoyi-zalozy/.

28.Свідоцтво про реєстрацію авторського права на твір №75359. База даних цифрових гістологічних та цитологічних зображень передракових та ракових станів молочної залози «BPCI2100». / О.М. Березький, Г.М. Мельник, С.О. Вербовий, О.Й. Піцун, В.Д. Николюк, Т.В. Дацко. Дата реєстрації 14.12.2017 р.

29.Свідоцтво про реєстрацію авторського права на твір №75360. Комп'ютерна програма «Інтелектуальна система діагностування передракових станів молочної залози на основі аналізу гістологічних та цитологічних зображень "HIAMS"». / О.М. Березький, О.Й. Піцун, Г.М. Мельник, П.Б. Лящинський, П.Б. Лящинський. Дата реєстрації 14.12.2017 р.

30.База даних цифрових гістологічних та цитологічних зображень різних форм раку молочної залози «CIFDB» («CIFDB») / Березький О. М., Мельник Г.М., Николюк В.Д., Дацко Т.В.: Свідоцтво про реєстрацію авторського права на твір № 52743 від 23.12.2013 р.

31.Березький О. М. База даних цитологічних та гістологічних зображень ауто- та ксеногенних тканин / О. М. Березький, Г.М. Мельник, Т.В. Дацко, С. О. Вербовий // Науковий вісник НЛТУ України: зб. наук.-техн. праць. – Львів: РВВ НЛТУ України. – 2014. – Вип. 24.10. – С. 338-345.

32.Березький О. М. Розроблення реляційної бази даних інтелектуальної системи автоматизованої мікроскопії / О.М. Березький, О.Й. Піцун, С.О. Вербовий, Т.В. Дацко // Науковий вісник національного лісотехнічного університету України: Збірник науково-технічних праць. - Львів: РВВ НЛТУ України. - 2017. –Т. 27, № 5. - С.125 -129.

33.Комп'ютерна програма «Інтелектуальна система для діагностування різних форм раку молочної залози на основі аналізу гістологічних та цитологічних зображень «IntelliD» («IntelliD») / Березький О. М., Мельник Г.М., Батько Ю.М., Дацко Т.В., Вальків В.: Свідоцтво про реєстрацію авторського права на твір № 52096 від 11.11.2013 р.

34.Березький О. М. Нечітка база знань інтелектуальної системи діагностування видів раку молочної залози / О. М. Березький, Г. М. Мельник, К. М. Березька // Вісник Хмельницького національного університету. Технічні науки. – 2013. – №6. – С.284-291.

35.Клещев А. С. Семантические порождающие модели. Общая точка зрения на фреймы и продукции в экспертных системах. Препринт. Владивосток: ИАПУ ДВНЦ РАН, 1986. – 39 с.

36.Березький О. М. Інтелектуальна система для діагностування різних форм раку молочної залози на основі аналізу гістологічних і цитологічних зображень / О. М. Березький, Г.М.Мельник, Ю.М. Батько, Т. В. Дацко // Науковий вісник НЛТУ України: зб. наук.-техн. праць. – Львів: РВВ НЛТУ України. – 2013. – Вип. 23.13. – С. 357-367.

37.Berezsky O. Database of histological and cytological images of different forms of breast cancer / Oleg Berezsky, Grygoriy Melnyk, Sergiy Verbovy // Proceedings of the International Conference Computer Science and Information Technologies (CSIT'2013), Lviv, Ukraine, November 11-16, 2013. – Lviv, 2013. – P. 67.

38.Березький О. М. Інтелектуальна система аналізу зображень ауто- та ксеногенних тканин / О. М. Березький, Г.М. Мельник, К. М. Березька, Т.В.

Дацко // Науковий вісник НЛТУ України: зб. наук.-техн. праць. – Львів: РВВ НЛТУ України. – 2014. – Вип. 24.11. – С. 323-330.

39.Berezsky O. An Intelligent System for Cytological and Histological Image Analysis / Oleh Berezsky, Grygoriy Melnyk, Tamara Datsko, Sergiy Verbovy // Proceedings of the 13 th International Conference «The Experience of Designing and Application of CAD Systems in Microelectronics» CADSM 2015, 24-27 February 2015, Polyana-Svalyava (Zakarpattya), Ukraine. – 2015. – P. 28-31.

40.Методи, алгоритми і програмні засоби опрацювання біомедичних зображень / Березький О. М., Батько Ю.М., Березька К.М., Вербовий С.О., Дацко Т.В., Дубчак Л.О., Ігнатєв І.В., Мельник Г.М., Николюк В.Д., Піцун О.Й. – Тернопіль: Економічна думка, ТНЕУ, 2017. – 330 с.

41.Березький О. М. Інтелектуальна система автоматизованої мікроскопії аналізу гістологічних та цитологічних зображень / О.М. Березький, О.Й. Піцун, П.Б. Лящинський, Г.М. Мельник // Штучний інтелект, Київ, 2017. - №2 (76). - С. 128-140.

42.Труб И. И. Объектно-ориентированное моделирование на C++: Учебный курс. / И. И. Труб - СПб.: Питер, 2006. - 411 с.

43.Братко И. Алгоритмы искусственного интеллекта на языке PROLOG. Третье издание / Иван Братко – Москва: Вильямс, 2004. – 640 с.

44.Баженова И.Ю. Язык программирования C++ АО "Диалог-МИФИ", 2017. 366 с.

45.Вебер Дж. Технология C++в подлиннике. QUE Corporation, 2016, "BHV-Санкт-Петербург",2017. 256 с.

46.Волш А. И. Основы программирования на C++для World Wide Web. IDG Books Worldwide,Inc.,1996, Издательство "Диалектика",2016. 458 с.

47.OpenCV. URL: https://uk.wikipedia.org/wiki/OpenCV.

48.Джамса К. Библиотека программиста Java. Jamsa Press, 2016, ООО "Попурри", 2016. 656 с.