Лабораторна робота №3

Експертні системи

Мета: сформувати поняття про експертні системи і головні моделі подання знань, навчитися проектувати найпростіші експертні системи.

Теоретичні відомості

Експертна система (ЕС) – це програма, яка оперує знаннями деякої предметної галузі з метою вироблення рекомендацій або розв'язання проблем. Експертна система може взяти на себе функції, які зазвичай потребують залучення досвіду людини-експерта, або грати роль асистента для людини, яка приймає рішення. Для експертних систем є типовими такі задачі:

- виділення корисної інформації з первинних даних;
- діагностика несправностей механізмів та хвороб людини;
- структурний аналіз складних об'єктів;
- вибір конфігурації складних багатокомпонентних систем;
- планування послідовності виконання операцій.

В загальному випадку експертна система складається з таких компонентів:

- база знань – зберігає дані і правила, які описують предметну галузь;
- машина логічного виводу – розв'язує поставлену задачу, використовуючи базу знань;
- компонент набуття знань – дозволяє розширювати базу знань;
- компонент пояснення – забезпечує пояснення, як саме була отримана відповідь на питання і які знання при цьому використовувались;
- інтерфейс з користувачем.

Структурні компоненти повинні забезпечувати такі функції ЕС:

- набуття знань;
- подання (зображення) знань;
- розв'язування задач (можливо, використовуючи не повністю визначені, достовірні дані);
- пояснення прийнятого рішення.

Експертна система характеризується способом *подання знань*. Теорія подання знань – це окрема галузь досліджень, яка вивчає методи асоціативного зберігання інформації, подібні до тих, які існують у мозку людини. Розглянемо деякі моделі подання знань.

Продукційна модель (модель, що ґрунтується на правилах) дозволяє описувати знання у вигляді правил (продукцій) вигляду `Якщо <умова> то <висновок>`. База знань складається з набору таких правил. Продукційна модель найчастіше використовується у промислових експертних системах. Вона характеризується наочністю, модульністю, простотою внесення доповнень і простотою механізму логічного виводу.

Семантична мережа – це орієнтований граф, вершинами якого є поняття, а дугами – відношення між ними. Поняттями зазвичай бувають абстрактні або конкретні об'єкти. Найчастіше відношення визначаються такими типами зв'язків:

- «частина-ціле» («клас-підклас», «елемент-множина»);
- функціональні (впливає);
- кількісні (більше, менше, дорівнює);
- просторові (близько, далеко, за, під, над);
- часові (раніше, пізніше, протягом);
- атрибутивні(мати властивість, мати значення);
- логічні (і, або).

Головною перевагою цієї моделі є відповідність сучасним уявленням про організацію пам'яті людини, недоліком – складність пошуку на семантичній мережі. Семантичні мережі використовуються в якості мови подання знань в таких ЕС, як PROSPECTOR, CASNBT, TORUS.

Фреймова модель була запропонована М. Мінським в 70-х роках. ***Фрейм*** – це структура даних для зображення стереотипних ситуацій. Всі знання про даний клас об'єктів або подій концентруються в єдиній структурі. Фрейм складається зі *слотів*, які зберігають значення атрибутів об'єкта. Кожен слот має ім'я і значення. Значенням слоту може бути практично будь-який об'єкт – числа, математичні вирази, тексти, посилання на інші слоти цього або іншого фрейму. Розрізняють фрейми-прототипи, які зберігаються в базі знань, і фрейми-екземпляри, які створюються для відображення реальних ситуацій на основі отриманих даних. При переході від прототипу до екземпляру (конкретизації фрейму) самому фрейму і його слотам надаються конкретні імена та відбувається заповнення слотів.

Перевагами фреймів як моделі подання знань є здатність відображати концептуальну основу організації пам'яті людини, гнучкість і наочність. Спеціальні мови подання знань в мережах фреймів FRL (Frame Representation Language) та інші дозволяють ефективно будувати промислові ЕС. Відомими фрейм-орієнтованими експертними системами є ANALYST і МОДИС.

Напишемо просту експертну систему «Визначник тварин», яка буде за даними характеристиками визначати відповідну тварину.

Нам знадобляться наступні вбудовані предикати:

`write/?` – виводить передані аргументи у вікні Dialog. Число аргументів може бути будь-яким. Атоми, числа, структури виводяться так, як записані в програмі, замість аргументів-змінних виводиться їх значення.

`nl/0` – переводить курсор у вікні Dialog на новий рядок.

`assert/1` – додає переданий в якості аргументу *факт* в кінець програми. При цьому сам файл з програмою залишається незмінним, факти лише завантажуються в пам'ять.

`retract/1` – видаляє з пам'яті всі факти, які уніфікуються з переданим фактом-аргументом.

Предикати assert/1 і retract/1 дозволяють змінювати програму (або базу даних) в ході її ж виконання.

Наша ЕС буде використовувати продукційну модель подання знань. Кожна тварина описуватиметься правилами вигляду

```
тварина(пінгвін):-
клас(птах), %пінгвін – птах,
невірно("літає"), % що не літає,
вірно("плаває"), %а плаває
вірно("має чорно-біле забарвлення"), !. %до того ж чорно-білий

клас(птах):-
вірно("має пір'я"), !. %птахи мають пір’я

клас(птах):-
вірно("літає"),
вірно("кладе яйця"), !. %або літають і кладуть яйця
```

Відсікання в кінці кожного правила потрібні для того, щоб Пролог-система не перевіряла зайвих альтернатив і знаходила лише одну тварину (наприклад, якщо стало відомо, що тварина має пір’я, то зрозуміло, що вона є птахом і немає сенсу перевіряти, чи кладе вона яйця).

Діалог з користувачем організуємо наступним чином: система буде ставити йому запитання щодо особливостей задуманої ним тварини, а користувач буде на них відповідати «так» або «ні». Оскільки різні тварини можуть мати однакові характеристики, то питання системи можуть повторюватися. Щоб запобігти цьому будемо зберігати відповіді користувача у вигляді фактів відповідь_так(X) (у випадку позитивної відповіді) та відповідь_ні(X) (для негативної відповіді). Тут X – деяка властивість тварини. Власне збереження відповідей в пам’яті реалізує предикат запам’ятати/2:

```
запамятати(X,так):-assert(відповідь_так(X)). %запам’ятати
позитивну відповідь – додамо відповідний факт
запамятати(X,ні):-assert(відповідь_ні(X)). %аналогічно з
негативною
```

Процес відгадування задуманої тварини проходитиме наступним чином. Викликом мети тварина(X) запускатиметься процес перебору всіх тварин з бази знань. При перевірці кожної тварини викликатимуться відповідні предикати вірно/1 або невірно/1. Ці предикати повинні визначити наявність або відсутність даної характеристики у тварини. Якщо людина вже дала відповідь (позитивну або негативну) стосовно цієї характеристики, то її можна знайти в базі знань, інакше потрібно задати відповідне питання і додати відповідь до бази. При цьому виклик вірно/1 повинен завершитися успішно лише у випадку наявності даної характеристики. Виклик невірно/1 навпаки, завершується успіхом, якщо відомо, що задумана тварина не має даної властивості.

```
вірно(X):-відповідь_так(X),!. %вже була дана позитивна відповідь,
перевірка закінчена
вірно(X):-
```

```
%якщо була негативна відповідь, то перевірка буде неуспішною (not
має неуспіх).
    not(відповідь_ні(X)),
%інакше (якщо ще не було ніякої відповіді), спитаємо людину
    спитати(X,так).
/*якщо відповідь була позитивною, то спитати(X, так) завершиться
успішно,
і весь предикат вірно/1 поверне істину; якщо ж відповідь негативна
то вся
перевірка буде неуспішною*/

%для невірно/1 все навпаки
невірно(X):-відповідь_ні(X),!. %була дана негативна відповідь
невірно(X):-
    not(відповідь_так(X)), %у випадку наявності позитивної
відповіді в базі знань перевірка завершується неуспішно
    спитати(X,ні). %інакше спитаємо користувача
```

Діалог з користувачем реалізує предикат `спитати/2`:

```
спитати(X,Answer):- %X - властивість, Answer - відповідь
    write("Воно ", X, "? (y/n)"),nl, %виведемо запитання
    readln("y"),!, %якщо користувач ввів "y" (позитивна
відповідь),
    запамятати(X,так),Answer=так; %то запам'ятаємо її
    запамятати(X,ні),Answer=ні. %інакше запам'ятаємо негативну
відповідь
```

Після того, як тварина визначена або не визначена (не знайдена в базі), потрібно видалити всі факти – відповіді людини, щоб при наступному запуску ЕС пам’ять була чистою. Для цього запишемо предикат `очистити/0`:

```
очистити:-retract(відповідь_так(_)),fail. %видалимо всі факти
відповідь_так
очистити:-retract(відповідь_ні(_)),fail. %видалимо всі факти
відповідь_ні
очистити. %врешті-решт виклик очистити/0 закінчимо успішно
```

`Retract/1` за один виклик видаляє один факт з бази, тому за допомогою `fail` задіємо механізм повернень. Тепер `retract/1` буде викликатися доти, доки ще є відповідні факти в базі знань. Коли всі факти буде видалено, `retract/1` завершиться невдачею, і система почне перевіряти інші альтернативи для `очистити/0`. Для того, щоб `очистити/0` після видалення всіх фактів повернув істину, ми додали відповідний факт.

Вся програма контролюється такими предикатами:

```
старт:-
    тварина(X), %запустимо процес визначення тварини
    write("Мабуть, це ",X, "."),nl,nl, %виведемо відповідь
    очистити. %видалимо всі дані про відповіді людини
старт:-
/*якщо в базі немає тварини з вказаними характеристиками, то
виклик тварина(X)
завершиться невдачею, і ми потрапимо сюди*/
    write("Я не знаю такої тварини."),nl,nl,
    очистити. %видалимо всі дані про відповіді людини
```

```
експерт:-
    старт,!, %почнемо діалог
    write("Спробуємо ще? (y/n)"),nl,
    readln("y"), експерт. %якщо людина хоче продовжувати, то
почнемо все з початку
```

Експертна система запускається запитом

```
експерт.
```

Отже, повний текст програми буде таким:

```
тварина(гепард):-
клас(ссавець),
клас("м'ясоїд"),
вірно("має рудий колір"),
вірно("має темні плями"),!.

тварина(тигр):-
клас(ссавець),
клас("м'ясоїд"),
вірно("має рудий колір"),
вірно("має темні смуги"),!.

тварина(жираф):-
клас(копитне),
вірно("має довгу шию"),
вірно("має довгі ноги"),
вірно("має темні плями"),!.

тварина(зебра):-
клас(копитне),
вірно("має темні смуги"),!.

тварина(страус):-
клас(птиця),
невірно("літає"),
вірно("має довгу шию"),
вірно("має довгі ноги"),
вірно("має чорно-біле забарвлення"),!.

тварина(пінгвін):-
клас(птиця),
невірно("літає"),
вірно("плаває"),
вірно("має чорно-біле забарвлення"),!.

тварина(альбатрос):-
клас(птиця),вірно("добре літає"),!.

клас(ссавець):-
вірно("має шерсть"),!.

клас(ссавець):-
вірно("годує дитинчат молоком"),!.

клас(птиця):-
```

```
вірно("має пір'я"),!.

клас(птиця):-
вірно("літає"),
вірно("кладе яйця"),!.

клас("м'ясоїд"):-
вірно("їсть м'ясо"),!.

клас("м'ясоїд"):-
вірно("має гострі зуби"),
вірно("має пазурі"),
вірно("має очі, спрямовані вперед"),!.

клас(копитне):-
клас(ссавець),
вірно("має копита"),!.

клас(копитне):-
клас(ссавець),
вірно("є жвачним"),!.

вірно(X):-відповідь_так(X),!. %вже була дана позитивна відповідь,
перевірка закінчена
вірно(X):-
%якщо була негативна відповідь, то перевірка буде неуспішною (not
має неуспіх).
    not(відповідь_ні(X)),
%інакше (якщо ще не було ніякої відповіді), спитаємо людину
    спитати(X,так).
/*якщо відповідь була позитивною, то спитати(X, так) завершиться
успішно,
і весь предикат вірно/1 поверне істину; якщо ж відповідь негативна
то вся
перевірка буде неуспішною*/

%для невірно/1 все навпаки
невірно(X):-відповідь_ні(X),!. %була дана негативна відповідь
невірно(X):-
    not(відповідь_так(X)), %у випадку наявності позитивної
відповіді в базі знань перевірка завершується неуспішно
    спитати(X,ні). %інакше спитаємо користувача

спитати(X,Answer):- %X - властивість, Answer - відповідь
    write("Воно ", X, "? (y/n)"),nl, %виведемо запитання
    readln("y"),!, %якщо користувач ввів "y" (позитивна
відповідь),
    запамятати(X,так),Answer=так; %то запам'ятаємо її
    запамятати(X,ні),Answer=ні. %інакше запам'ятаємо негативну
відповідь

запамятати(X,так):-assert(відповідь_так(X)).
запамятати(X,ні):-assert(відповідь_ні(X)).
```

```
очистити:-retract(відповідь_так(_)),fail. %видалимо всі факти
відповідь_так
очистити:-retract(відповідь_ні(_)),fail. %видалимо всі факти
відповідь_ні
очистити. %врешті-решт виклик очистити/0 закінчимо успішно

старт:-
    тварина(X), %запустимо процес визначення тварини
    write("Мабуть, це ",X, "."),nl,nl, %виведемо відповідь
    очистити. %видалимо всі дані про відповіді людини
старт:-
/*якщо в базі немає тварини з вказаними характеристиками, то
виклик тварина(X)
завершиться невдачею, і ми потрапимо сюди*/
    write("Я не знаю такої тварини."),nl,nl,
    очистити. %видалимо всі дані про відповіді людини

експерт:-
    старт,!, %почнемо діалог
    write("Спробуємо ще? (y/n)"),nl,
    readln("y"), експерт. %якщо людина хоче продовжувати, то
почнемо все з початку

/*коли в базі ще немає жодного факту відповідь_так або
відповідь_ні, звернення
до цих предикатів викличе помилку. Тому потрібно записати наступні
два речення*/
відповідь_так(_):-fail.
відповідь_ні(_):-fail.
```

Практична частина

Випробувати предикати для роботи з базою даних.

1. Додамо декілька фактів в базу даних. Запустіть інтерпретатор і, не відкриваючи ніякої програми, запустіть з вікна Dialog такі цілі:

```
assert(факт(1, "один")).
assert(факт(1, "ще один")).
assert(факт(2, "два")).
assert(факт(3, "три")).
```

2. Тепер перевіримо ці факти. Поставте такий запит:

```
факт(X, Y).
```

Ми додавали чотири факти, отже ви повинні отримати чотири розв’язки.

3. Тепер почнемо видаляти факти з пам’яті:

```
retract(факт(1, Y)).
```

Із аргументом уніфікуються два факти з бази, вони і будуть видалені – отримаємо два розв’язки. Тепер запит

```
факт(X, Y).
```

поверне значення лише з тих двох фактів, що залишилися.

4. Нарешті, видалимо всі інші факти:

```
retract(факт(X, Y)).
```

Спробуйте тепер спитати:

```
факт(X, Y).
```

Створити експертну систему «Визначник тварин» та перевірити її роботу.

1. Створіть нову програму, напишіть текст нашої експертної системи та збережіть її.

2. Запустіть експертну систему запитом

```
експерт.
```

Програма буде задавати питання у вікні Dialog. Для того, що відповісти на питання, потрібно у цьому вікні ввести «y» або «n» та натиснути Enter.

1. Перевірте, чи всі тварини правильно визначаються.

Завдання

Розробіть свою експертну систему за обраною темою (у кожного студента своя). Можливі теми:

1. Визначник акваріумних риб.
2. Визначник рослин (квіток).
3. Вибір професії.
4. Вибір місця відпочинку.
5. Визначник марки автомобіля (мотоцикла або ін.).
6. Вибір і купівля автомобіля.
7. Діагностика несправності автомобіля (або іншого пристрою).
8. Вибір і купівля комп’ютера.
9. Діагностика захворювань людини.
10. Прогнозування погоди.
11. Тема за вашим вибором.