»Лекционен курс

»Изкуствен интелект

Експертни системи

проф. Станимир Стоянов

Увод

- Експертната система е програма с изкуствен интелект, която има познания на експертно ниво за определена проблемна област и може да използва знанията си подготовка на експертизи
- В идеалния случай една експертна система може да бъде заместител на човек-експерт
- Едуард Файгенбаум от Станфордския университет дефинира експертните системи като „интелигентна компютърна програма, която използва знания и процедури за извод за решаване на проблеми, които са достатъчно трудни, за да изискват значителна човешка експертиза за техните решения“
- Това е клон на ИИ, въведен от изследователи в проекта за евристично програмиране на Станфорд

Определения

- Експертна система е компютърна програма, предназначена да действа като експерт в определена област на знания или област на експертиза
  - Експертните системи са известни още като системи, базирани на знания
- Експертните системи са сложни компютърни програми, които обработват знания за решаване на проблеми.
- Една експертна система предлага интелигентни съвети или взема интелигентно решение посредством машина за извод
- Експертна система е компютърна програма, която съдържа база знания и набор от алгоритми или правила (машина за извод), които извличат нови факти от знанията и от входни данни

Инженеринг на знания

- Методът, използван за конструиране на такива системи, инженеринг на знания, извлича набор от правила и данни от експерт (или експерти) обикновено чрез анкети
- След това този материал се организира във формат, подходящ за запитване, манипулиране и отговор
- Въпреки, че такива системи не могат напълно да заменят хората-експерти, те могат да служат като полезни помощници

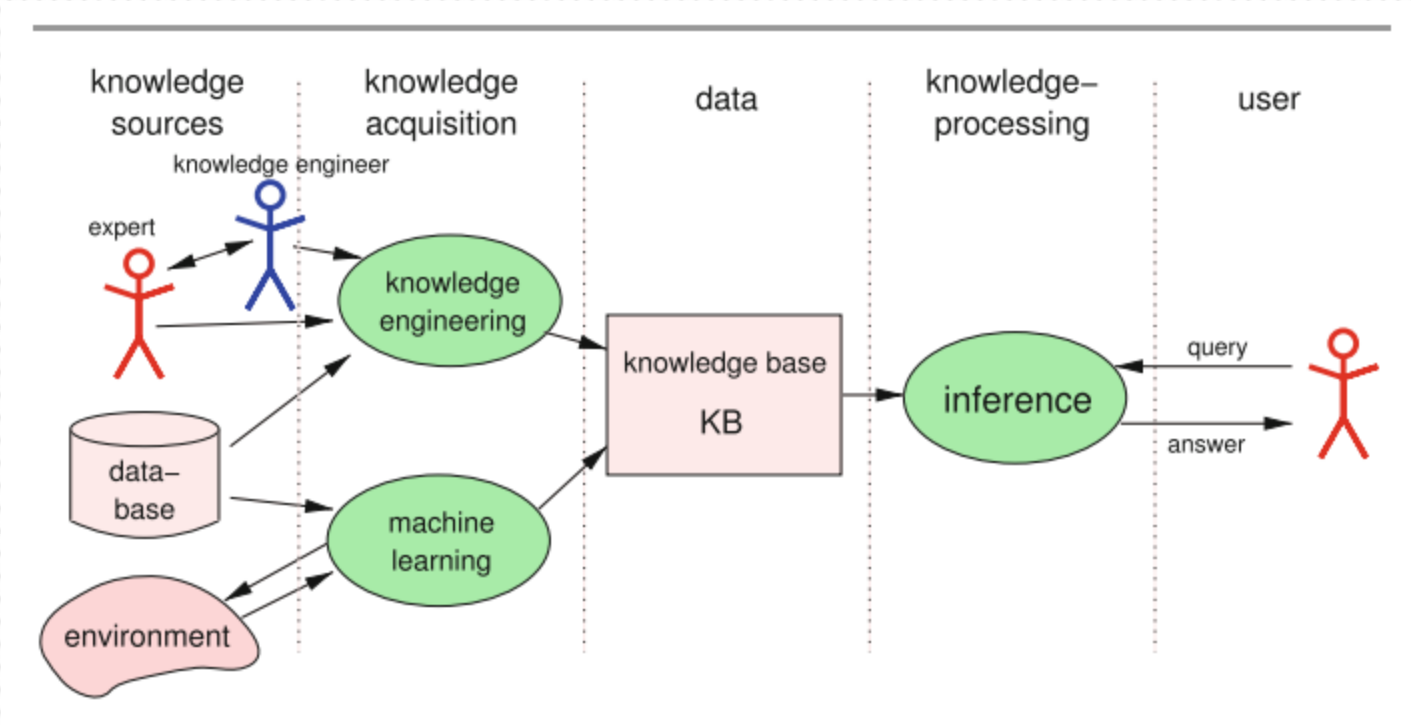
Характеристики на една експертна система

- Способността за решаване на сложни проблеми със същия (или по-висок) успех като човек-експерт
- Евристични разсъждения чрез емпирични правила
- Способност за работа с данни, които съдържат грешки, като се използват процедурни правила с несигурност
- Способност за разглеждане на множество хипотези едновременно

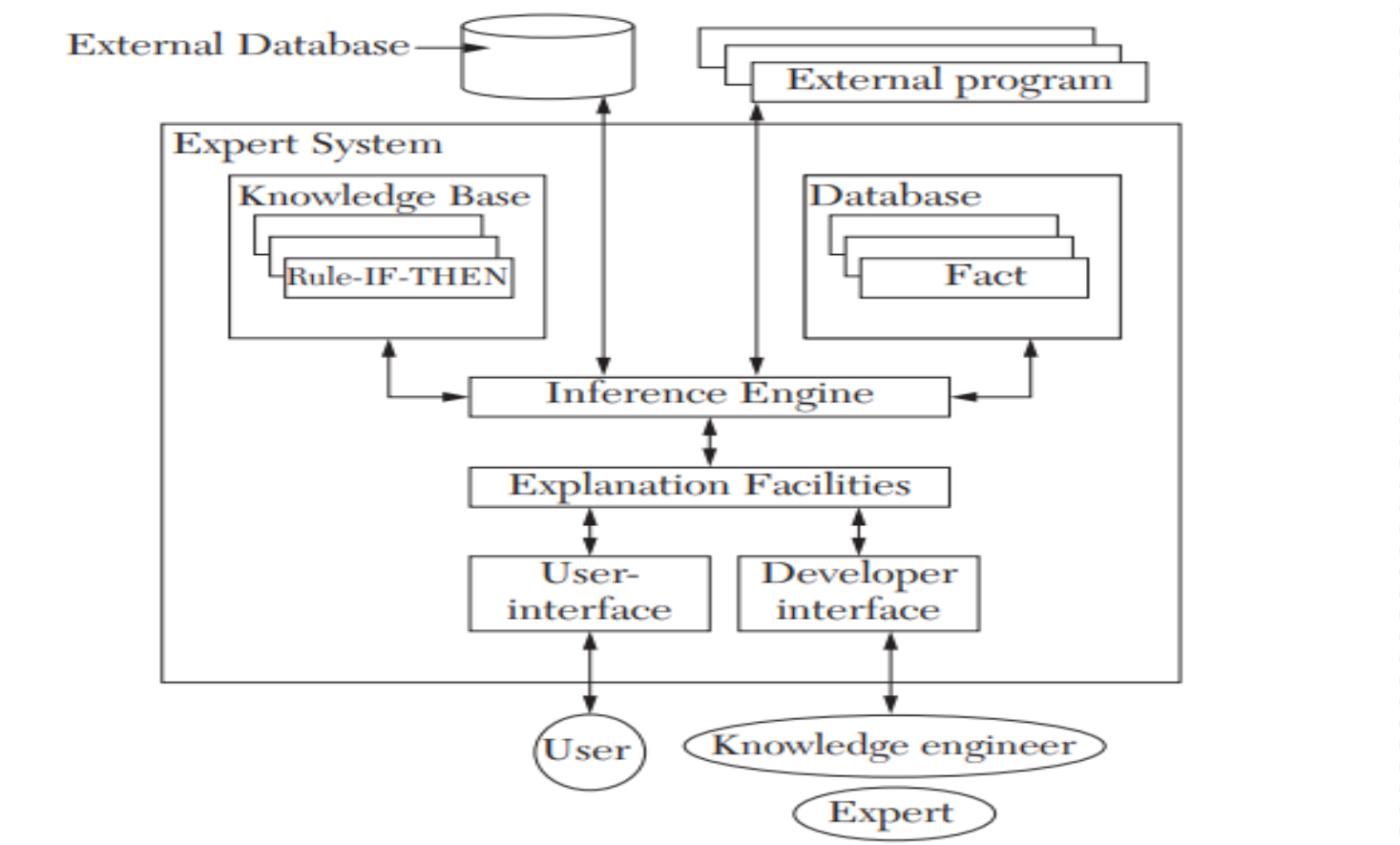
Характеристики на една експертна система

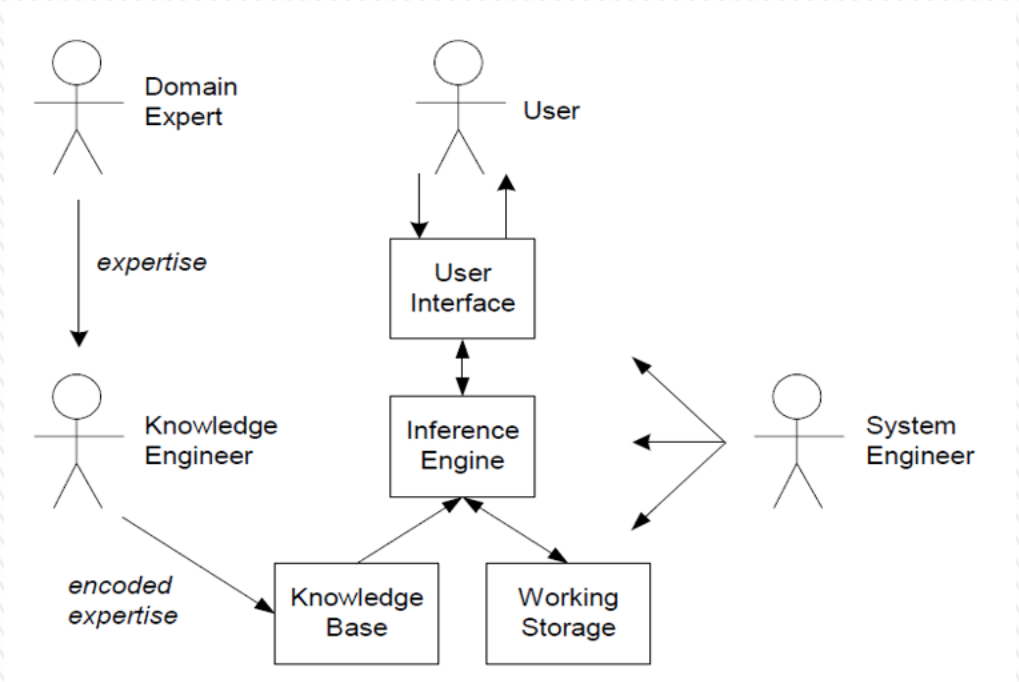
- Способност за отговор в разумен период от време
  - Времето е от решаващо значение, особено за системите в реално време
- Надеждност
- Не е черна кутия
  - Експертната система трябва да може да обясни стъпките на процеса на разсъждение
  - Тя трябва да е в състояние да обоснове заключенията си по същия начин, както хората-експерти обяснява как са стигнали до конкретно заключение
- Нуждаят се от големи инвестиции
  - Възвръщаемост на инвестициите

Структура на класическа система, използваща знания



Архитектура на една ЕС




Domain Expert
User
expertise
User Interface
Knowledge Engineer
Inference Engine
System Engineer
encoded expertise
Knowledge Base
Working Storage

Видове архитектури на ЕС

- Има два типа архитектури в експертните системи:
  - Архитектура на системата, базирана на правила
    - + Наричат се производствени системи
  - Архитектура на непроизводствена система

Базирана на правила ЕС

- Това е най-често срещаната форма на архитектура, използвана в експертни и други базирани на познания системи
- Този тип система използва знания под формата на производствени правила:
  - АКО: Условие 1 и условие 2
  - ТОГАВА: Вземете действие 3
- Всяко правило представлява малка част от знания, свързани с дадено област на експертиза
- Редица свързани правила заедно могат да съответстват на верига от изводи, които водят от някои първоначално известни факти до някои полезни заключения
- Постига се извод в производствена система чрез процес на верижно свързване на правила докато се стигне до някакво заключение:
  - Рекурсивно
  - Свързване напред
  - Свързване назад

Модули на една ЕС

- Основните модули на експертната система са:
  - База знания
  - Машина за извод
  - Потребителски интерфейс
  - Обяснителен модул
  - Модул за придобиване на знания
  - Външен интерфейс
  - База данни

База знания

- Базата от знания е организирана колекция от факти или правила конкретна приложна област
  - Съдържа специфични за приложната област смислени (значещи) знания
- Фактите трябва да са придобити от хора-експерти чрез интервюта и наблюдения
- Необходими са знания, за да се прояви интелигентност
- Успехът на всяка експертна система до голяма степен зависи от събирането на коректни ни и прецизни знания

Компоненти на базата знания

- Базата знания на една експертна система е хранилище за фактологични и евристични знания:
  - Фактически знания за приложната област
  - Евристични знания - генериране на точни преценки, способността на човек да прави оценки и прогнози

Представяне на знания

- Това е метод, използван за организиране и формализиране на знанията в базата от знания
- Представително ниво на знанията
  - Обикновено правила (производствени правила)
  - Структорно представяне на знанията – фрейми, семантични мрежи, обекти
- Правила:
  - Ако дадено условие (условия) е вярно, тогава може да се направи следното заключение (или да се предприемат следните действия).“
- Базата от знания на една голяма експертна система обикновено включва хиляди правила
- Към заключението на всяко от производствено правило може да бъде прекрепен вероятностен фактор (несигурно заключение)

Машина за извод

- Много важен модул на една експертна система
- Знанията трябва да се съхраняват в базата знания във формализирана форма, която да е разбираема за машината за извод
- Машината за извод включва следните функционални елементи:
  - Система за управление - определя реда на пробване на правила от базата знания
  - Интерпретатор на правила – определя приложимост на правила и активира приложимите. Решава също конкурентност (повече от едно приложими правила)
  - Механизъм за обяснение – обяснява на потребителя процеса на разсъждение и генерира отчети

Машина за извод

- Машината за извод многократно прилага правилата към работната памет, като добавя нова информация (получена от заключенията на правилата) към нея, докато се генерира или потвърди целевото състояние
- Съществуват различни стратегии - машината за извод за системи, базирани на правила, обикновено използва следните две стратегии:
  - Свързване напред
  - Свързване назад

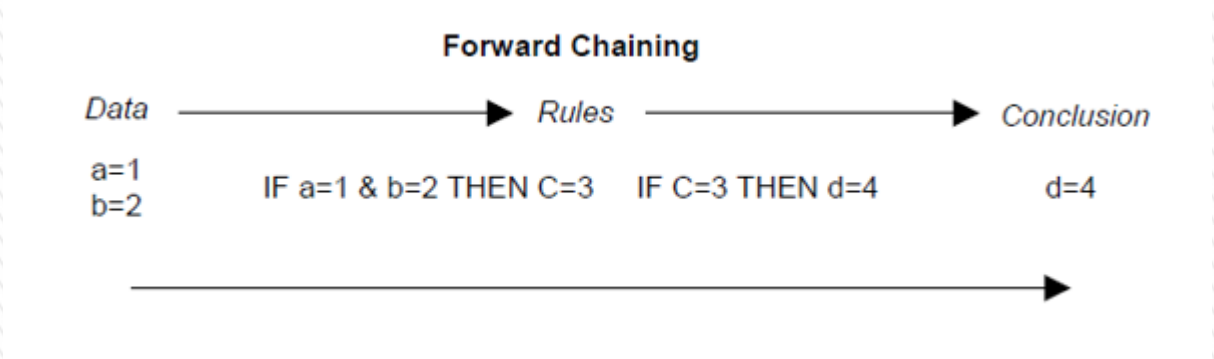
Свързване напред

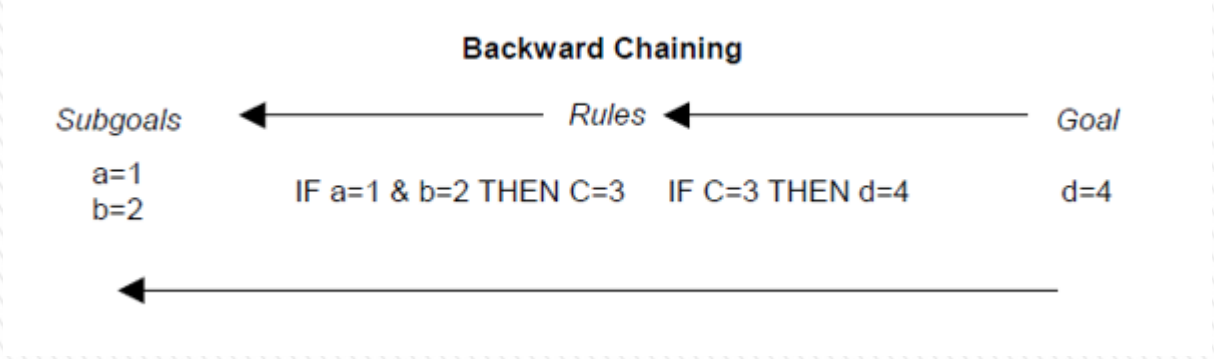
- Стратегия, направлявана от данни
- Процесът на заключение преминава от фактите за конкретния случай към целта (заключението)
- Стратегията се ръководи от наличните факти в работната памет и от предпоставките, които могат да бъдат удовлетворени
- Машината за извод се опитва да унифицира (съпостави) условната част (IF) на всяко правило в базата знания с фактите, налични в момента в работната памет
- Обикновено се използва за решаване на проблеми, включващи планиране
  - Напр., конфигуриране на сложен продукт, установка, ...

Свързване назад

- Машината за извод се опитва да съпостави предполагаемо заключение със състоянието на целта или подцелта в заключението (THEN) на правилото
- Ако се намери такова правило, неговата предпоставка се превръща в новата подцел
- В експертна система с малко възможни целеви състояния това е добра стратегия
- Ако предполагаемото целево състояние не може да бъде подкрепено от предпоставките, системата ще се опита да докаже друго целево състояние
  - Следователно, възможно заключение е да се извърши преглед, докато не се срещне целево състояние, което може да бъде подкрепено от предпоставките
- Най-подходяща за приложения, в които възможните заключения са ограничени по брой и добре дефинирани

Работа с правила





Потребителски интерфейс

- Потребителят трябва да има възможност за комуникация със системата
- Компонентът на експертна система, който помага на потребителя да комуникира с нея, се нарича потребителски интерфейс
- Функцията на потребителския интерфейс е да осигури средства за двупосочна комуникация, при която потребителят описва проблема и системата отговаря с решения или препоръки
- Потребителският интерфейс помага да се обясни как експертната система е стигнала до конкретна препоръка.
- Обяснението може да се появи в следните форми:
  - естествен език, показан на екрана
  - словесни разкази на естествен език
  - списък с номерата на правилата, показани на екрана

Обяснителен модул

- Това е част от потребителския интерфейс
- Позволява на потребителя да попита експертната система как се достига до конкретно заключение и защо е необходима конкретна задача (факт)
- Експертна система трябва да може да обясни своите аргументи и да обоснове своите съвети, анализи или заключения

Механизъм за придобиване на знания

- Основната пречка в развитието на една експертна система може да бъде придобиването на знания
- Това включва извличане, събиране, анализ, моделиране и валидиране на знания (инженеринг и управление на знания)
- Съществуват различни техники за придобиване на знания

Външен интерфейс

- Това позволява на експертна система да работи с външни системи, вкл.:
  - Програми, написани на конвенционални езици за програмиране като Java, Python,C, C++, C#, Pascal, Fortran, Basic, ….
  - Комуникационна връзка между експертната система и външната среда
  - Сензорни мрежи
  - …
- Експертни системи в реално време са от огромна значение в управлението на промишлени процеси, атомни електроцентрали, летателни апарати, космическа техника, ...
- Комуникационната подсистема е част от външния интерфейс, който позволява на системата да комуникира с глобална база данни

База данни

- Базата данни е колекция от данни, която е организирана така, че да може лесно да бъде достъпна, управлявана и актуализирана
- „Бележник“, който машината за изводи може да използва, за да съхранява данни, докато работи по даден проблем
- Съдържа всички данни за текущата задача, включително:
  - Отговорите на потребителя на въпроси
  - Данни от външни източници
  - Междинни резултати от разсъжденията
  - Междинни изводи, направени до момента

База данни

- Има ясно разграничение между базата знания и базата данни
- Базата знания съдържа ноу-хау и може да се приложи в много различни случаи
- Веднъж създадена, базата знания ще бъде запазена и използвана многократно
- Базата данни съдържа данни за конкретния случай, който се изпълнява в момента
- Пример:
  - Базата знания относно проблемите с продажбите може да бъде приложима за всеки малък производствен бизнес
  - Базата данни ще съдържа данни за конкретна компания
- Базата данни може да се нарече още „модел на света“.

Задача: разработване на конкретна експертна система

Формат на правилата

```
rule <rule id>:
[<N>: <condition>, .......]
==>
<action>, ....].
```

където:

- `rule id` – уникален идентификатор на правилото;
- `N` – опционална идентификация на условието;
- `condition` – шаблон за унификация с работната памет;
- `action` – действие, което трябва да се извърши.

Машина за извод

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                          %
% SIE: Simple Inferance Engine             %
% ===================================     %
% File: oops.pl                            %
%                                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A toy production system interpreter. It uses a forward chaining,
% data driven, rule based approach for expert system development.
%
% Rule structure support
:-op(800,xfx,==>). %used to separate LHS and RHS of rule
:-op(500,xfy,:).      %used to separate attributes and values
:-op(810,fx,rule).   %used to define rule
:-op(700,xfy,#).    %used for unification instead of =

main :-
        welcome,
        supervisor.

welcome :-
        nl, nl,
        write(' A Toy Production System '), nl, nl,
        write(' This is an interpreter for files containing rules coded in the '), nl,
        write(' OOPS format.'), nl, nl,
        write(' The => prompt accepts two commands:'), nl, nl,
        write(' go. - starts the inference '), nl,
        write(' exit. - does what you would expect '), nl, nl.
```

Машина за извод

```
%the supervisor, uses a repeat fail loop to read and process commands
%from the user
supervisor :-
        repeat,
        write('=>'),
        read(X),
        doit(X),
        X = exit.

doit(X) :-
        do(X).

%actions to take based on commands
do(exit) :-
        !.
do(go) :-
        initialize,
        go,
        !.
do(_) :-
        write('invalid command').
```

Машина за извод

```
%assert each of the initial conditions into working storage
initialize :-
        initial_data(X),
        assert_list(X).

%working storage is represented by database terms stored
%under the key "fact"
assert_list([]) :-
        !.
assert_list([H|T]) :-
        assertz(fact(H)),
        !,
        assert_list(T).
```

Машина за извод

```
%the main inference loop, find a rule and try it. if it fired, say so
%and repeat the process. if not go back and try the next rule. when
%no rules succeed, stop the inference
go :-
        call(rule ID: LHS ==> RHS),
        try(LHS,RHS),
        write('Rule fired '), write(ID), nl,
        !,
        go.
go.

%find the current conflict set.
%conflict_set(CS) :-
%bagof(rule ID: LHS ==> RHS,
%[rule ID: LHS ==> RHS, match(LHS)],CS).
%match the LHS against working storage, if it succeeds, process the
%actions from the RHS
try(LHS,RHS) :-
        match(LHS),
        process(RHS,LHS),
        !.
```

Машина за извод

```
%recursively go through the LHS list, matching conditions against
%working storage
match([]) :-
        !.
match([N:Prem|Rest]) :-
        !,
        (fact(Prem)
        ;
        test(Prem) %a comparison test rather than a fact
        ),
        match(Rest).
match([Prem|Rest]) :-
        (fact(Prem) %condition number not specified
        ;
        test(Prem)
        ),
        match(Rest).
```

Машина за извод

```
%various tests allowed on the LHS
test(not(X)) :-
        fact(X),
        !,
        fail.
test(not(X)) :-
        !.
test(X #Y) :-
        X = Y,
        !.
test(X > Y) :-
        X > Y,
        !.
test(X >= Y) :-
        X >= Y,
        !.
test(X < Y) :-
        X < Y,
        !.
test(X =< Y) :-
        X =< Y,
        !.
test(X = Y) :-
        X is Y,
        !.
test(member(X,Y)) :-
        member(X,Y),
        !.
```

Машина за извод

```
%recursively execute each of the actions in the RHS list
process([],_) :-
        !.
process([Action|Rest],LHS) :-
        take(Action,LHS),
        !,
        process(Rest,LHS).

%if its retract, use the reference numbers stored in the Lrefs list,
%otherwise just take the action
take(retract(N),LHS) :-
        (N == all
        ;
        integer(N)
        ),
        retr(N,LHS),!.
take(A,_) :-
        take(A),!.
take(retract(X)) :-
        retract(fact(X)),
        !.
take(assert(X)) :-
        asserta(fact(X)),
        write(adding-X), nl,
        !.
```

Машина за извод

```
take(X #Y) :-
        X = Y,
        !.
take(X = Y) :-
        X is Y,
        !.
take(write(X)) :-
        write(X),
        !.
take(nl) :-
        nl,
        !.
take(read(X)) :-
        read(X),
        !.
take(prompt(X,Y)) :-
        nl, write(X), read(Y),
        !.
take(member(X,Y)) :-
        member(X,Y),
        !.
take(list(X)) :-
        lst(X),
        !.
```

Машина за извод

```
%logic for retraction
retr(all,LHS) :-
        retrall(LHS),
        !.
retr(N,[]) :-
        write('retract error, no '-N), nl,
        !.
retr(N,[N:Prem|_]) :-
        retract(fact(Prem)),
        !.
retr(N,[_|Rest]) :-
        !,
        retr(N,Rest).

retrall([]).
retrall([N:Prem|Rest]) :-
        retract(fact(Prem)),
        !, retrall(Rest).
retrall([Prem|Rest]) :-
        retract(fact(Prem)),
        !, retrall(Rest).
retrall([_|Rest]) :- %must have been a test
        retrall(Rest).
```

Машина за извод

```
%list all of the terms in working storage
lst :-
        fact(X),
        write(X), nl,
        fail.
lst :-
        !.

%lists all of the terms which match the pattern
lst(X) :-
        fact(X),
        write(X), nl,
        fail.
lst(_) :-
        !.

%utilities
member(X,[X|Y]).
member(X,[Y|Z]) :-
        member(X,Z).
```

Правила

rule id1: IF has(X,hair)
THEN X is mammal.
rule id2: IF gives(X,milk)
THEN X is mammal.
rule id3: IF has(X,feathers)
THEN X is bird.
rule id4: IF flies(X) and lays_eggs(X)
THEN X is bird.
rule id5: IF eats_meat(X)
THEN X is carnivore.

Правила

rule id6: IF has(X,pointed_teeth) and has(X,claws) and has(X,forward_eyes)
THEN X is carnivore.
rule id7: IF isa(X,mammal) and has(X,hoofs)
THEN X is ungulate.
rule id8: IF isa(X,mammal and chews_cud(X)
THEN X is ungulate and even_toed(X).
rule id9: IF isa(X,mammal) and isa(X,carnivore) and has(X,tawny_color) and has(X,dark_spots)
THEN X is cheetah.
rule id10: IF isa(X,mammal) and isa(X,carnivore) and has(X,tawny_color) and has(X,black_stripes)
THEN X is tiger.

Правила

rule id11: IF isa(X,ungulate) and has(X,long_neck) and has(X,long_legs) and has(X,dark_spots)
THEN X is giraffe.
rule id12: IF isa(X,ungulate) and has(X,black_stripes)
THEN X is zebra.
rule id13: IF isa(X,bird) and does_not_fly(X) and has(X,long_neck) and has(X,long_legs) and has_attr(X,black_and_white)
THEN X is ostrich.
rule id14: IF isa(X,bird) and does_not_fly(X) and swims(X) and has_attr(X,black_and_white)
THEN X is penguin.
rule id15: IF isa(X,bird) and flies_well(X)
THEN X is albatross.
rule id16: IF isa(Animal,Type) and parent(Animal,Child)
THEN Child is Type.
rule id17: IF even_toed(X) and has_attr(X,slow) and isa(X,ungulate)
THEN X is sloth.

База знания

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                         %
% Knowledge Base: "Animal"                %
% ====================================    %
% File: animal.pl                         %
%                                         %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Rules for animal identification.
% The first three rules are an input loop.
% Enter attributes that match the patterns in the rules.
% For example: has(robie,hair), or lays_eggs(suzie). These facts will
% help identify robie and suzie. Enter "end" to end the input loop.
%
% The attributes can also be put in the list of initial_data.
% Example: (working memory)
% This should lead to the identification of dennis and diana as zebras.
%-----------------------------------------------------------------------
% В зависимост от това, кой режим е включен (без коментар)
% Ако и двата, тогава, този, зададен физически първи
% initial_data([goal(animal_id)]).  %потребителят трябва да зададе атрибутите
initial_data([has(dennis,hair), %атрибутите се четат от тук (работната памет)
              has(dennis,hoofs),
              has(dennis,black_stripes),
              parent(dennis,diana)
              ]
).
```

База знания

```
%rules
rule 1:
        [1: goal(animal_id)]
        ==>
        [assert(read_facts),
        retract(1)].
rule 2:
        [1: end,
        2: read_facts]
        ==>
        [retract(all)].
rule 3:
        [1: read_facts]
        ==>
        [prompt('Attribute ? ',X),
        assert(X)].
```

База знания

```
rule id1:
        [1: has(X,hair)]
        ==>
        [assert(isa(X,mammal)),
        retract(all)].
rule id2:
        [1: gives(X,milk)]
        ==>
        [assert(isa(X,mammal)),
        retract(all)].
rule id3:
        [1: has(X,feathers)]
        ==>
        [assert(isa(X,bird)),
        retract(all)].
```

Заявка към ЕС

```
?- main.


 A Toy Production System

 This is an interpreter for files containing rules coded in the
 OOPS format.

 The => prompt accepts two commands:

 go. - starts the inference
 exit. - does what you would expect

=>go.
adding-isa(dennis,mammal)
Rule fired id1
adding-isa(dennis,ungulate)
Rule fired id7
adding-isa(dennis,zebra)
Rule fired id12
adding-isa(diana,zebra)
Rule fired id16
=>|: exit.

true .
```

Архитектура на непроизводствената система

- Вместо правила, тези системи използват по-структурирани схеми за представяне като:
  - Семантична (асоциативна) мрежа
  - Фрейми
  - Дървовидна структура (дървета на решенията)
  - Невронни мрежи

Семантични мрежи

- Схемите за представяне на семантичните мрежи са мрежи, съставени от възли, свързани с насочени дъги
- Възлите представляват обект, атрибути, концепции или други основни единици, а дъгите, които са етикетирани, описват връзката между двата възела, които свързват
- Специалните мрежови връзки включват връзките IS-A и HAS-PART, които обозначават обект като същество определен тип обект (принадлежащ към клас на обекта) и е подчаст от друг обект, респ

Семантични мрежи

- Асоциативните мрежови представяния са особено полезни при изобразяване на йерархична структура на знания, където наследяването на свойства е често срещано
- Обектите, принадлежащи към клас от други обекти, могат да наследят много от характеристиките на класа
- Наследяването може да се третира като форма на извод по подразбиране
- Това улеснява съхранението на информация, когато се споделя от много обекти, както и процеса на извод

CASNET

- Например, една експертна система, базирана на използването на асоциативно мрежово представяне, е CASNET (Casual Associative Network), която е разработена в университета Rutgers през 70-те години (Weiss et al., 1978)
- CASNET се използва за диагностициране и препоръчване на лечение на глаукома, една от водещите причини за слепота

CASNET

- Мрежата в CASNET е разделена на три вида знания:
  - Наблюдения на пациента (тестове, симптоми и други признаци):
    - Те се предоставят от потребителя по време на интерактивна сесия със системата
    - Системата се представя на потребителя по време на интерактивна сесия
    - Системата представя заявки от типа на менюто и потребителят избира един от няколкото възможни избора
  - Патофизиологични състояния:
    - Тези наблюдения помагат да се установи анормалното състояние, причинено от болестния процес
    - Състоянието се установява чрез случайния мрежов модел като част от причинно-следствената връзка, свързана със симптомите и други признаци на заболявания
  - Категории на заболяванията:
    - Изводът се постига чрез преминаване на мрежата, следвайки най-възможните пътища на причини и последици
    - След като се определи достатъчно силен път през мрежата, диагностичните заключения се извеждат с помощта на класификационни таблици, които интерпретират моделите на случайната мрежа
    - Тези таблици са подобни на интерпретациите на правила.

Фреймови архитектури

- Рамките са структурирани набори от тясно свързани знания, като например име на обект или концепция, основните атрибути на обекта, съответните му стойности и евентуално някаква прикачена процедура (процедури ако е добавено, ако е необходимо или ако е премахнато)
- Стойностите и процедурата на атрибута се съхраняват в определени слотове и факти за слотовете на рамката
- Отделните рамки обикновено са свързани заедно като мрежа и, подобно на възлите, това е асоциативна мрежа, включваща наследяване на свойства и разсъждения по подразбиране
- Няколко експертни системи са конструирани с рамкова архитектура и редица строителни инструменти, които създават и манипулират рамково структурирани системи бяха разработени

Фреймови архитектури

- Например, PIP (Present Illness Program) беше използвана за диагностициране на пациенти с помощта на ниска цена, лесно достъпна информация
- Медицинските познания в PIP са организирани в рамкови структури, където всеки кадър е съставен от категориите слотове с имена като:

Фреймови архитектури

- типични находки
- критерии за логично решение
- допълващи отношения с други рамки
- диференциална диагноза
- точкуване.

» Специален IS-достатъчен слот се използва за потвърждаване на наличието на заболяване, когато ключовите констатации корелират със съдържанието на слота

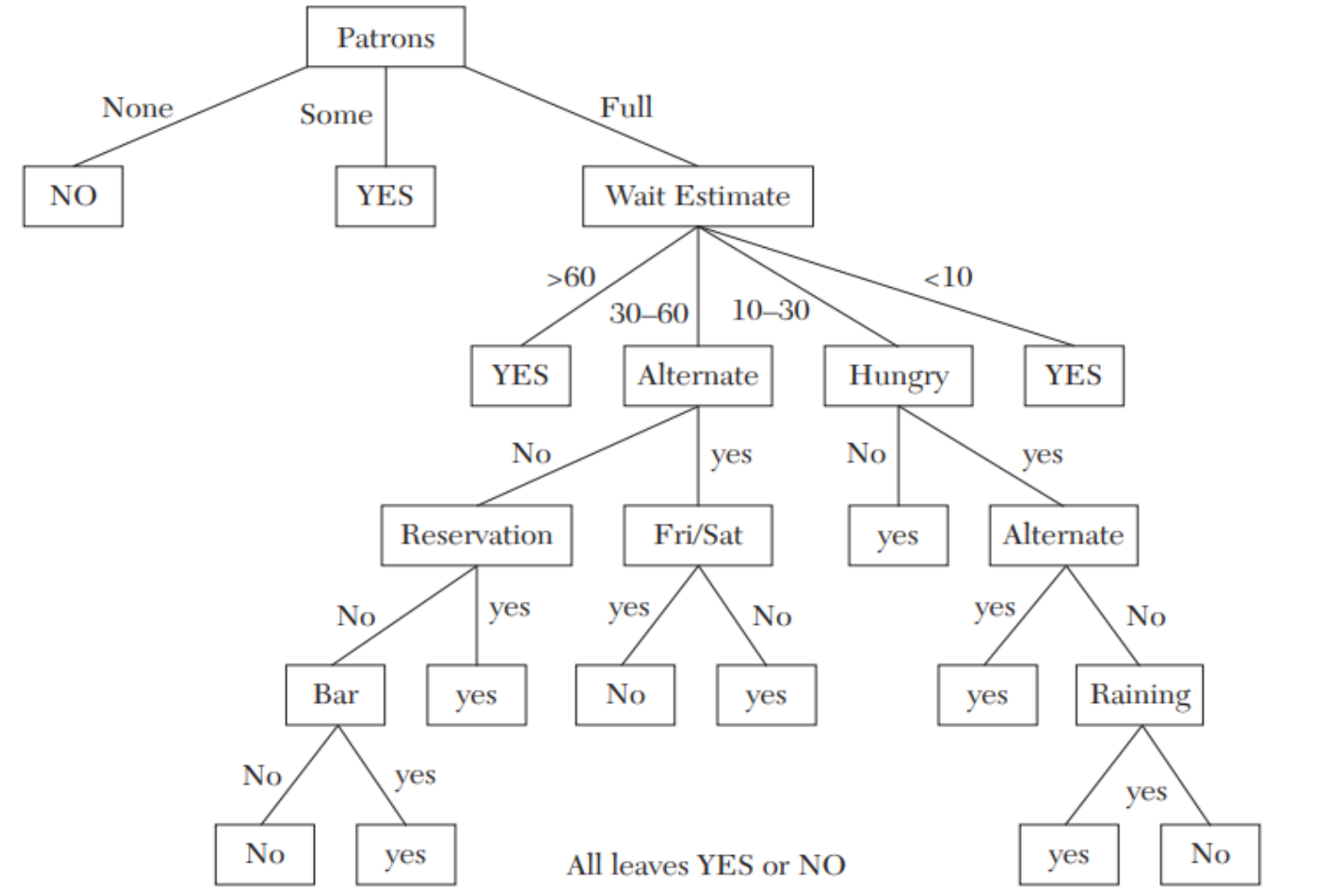
Архитектури, използващи дърво на решения

- Когато знанието може да бъде структурирано по начин отгоре до долу, то може да се съхранява под формата на дърво на решенията
- Например, идентифицирането на обекти (неизправности на оборудването, заболявания на физически обекти и други подобни) може да съответства на атрибута на обекта, а крайните възли могат да съответстват на идентичностите на обектите
- Дървото на решенията взема вход от даден обект
- чрез набор от свойства и извежда булева стойност (решение да/не)
- Всеки вътрешен възел в дървото съответства на тест на едно свойство
- Клоновете са обозначени с възможни стойности на теста.

Архитектури, използващи дърво на решения

- Например, да предположим, че проблемът е в чакане на маса в ресторант
- Дървото на решенията решава дали да изчака (или не) в дадена ситуация
- Ето някои от следните атрибути:
  - Алтернатива: Алтернативен ресторант в близост
  - Бар: Бар зона за изчакване
  - Пет/събота: вярно в петък и събота
  - Гладен: Независимо дали сме гладни
  - Покровители: Колко души са в ресторанта (никой, някои или пълни)
  - Цена: Ценови диапазон ($, $$ или $$$)
  - Дъжд: Навън вали
  - Резервация: Дали сме направили резервация
  - Тип: Вид ресторант (френски, италиански, тайландски или бургер)
  - Приблизително време за изчакване (<10, 10–30, 30–60 или >60).

Дърво на решение



Предимства на архитектурата на дървото на решенията

- Дърветата на решенията са отворени системи, тъй като е лесно да се свърже края на пътя в дървото на решенията, за да се започне друго дърво на решенията.
- Дърветата на решенията са прости естествени програми, които могат да се адаптират към сложност и хаотични условия.
- Дърветата на решенията са „бяла кутия“, което означава, че са прозрачни и лесни за разбиране и тълкуване. Хората са в състояние да разбират дърветата на решенията и следователно те са предназначени за организирано задържане на знания.
- Дърветата на решенията могат да имат стойност много бързо, дори и с малък брой възли. От тяхното използване могат да се получат важни прозрения, които често стимулират идеи за еволюция на знанието, които не са били очевидни в началото.

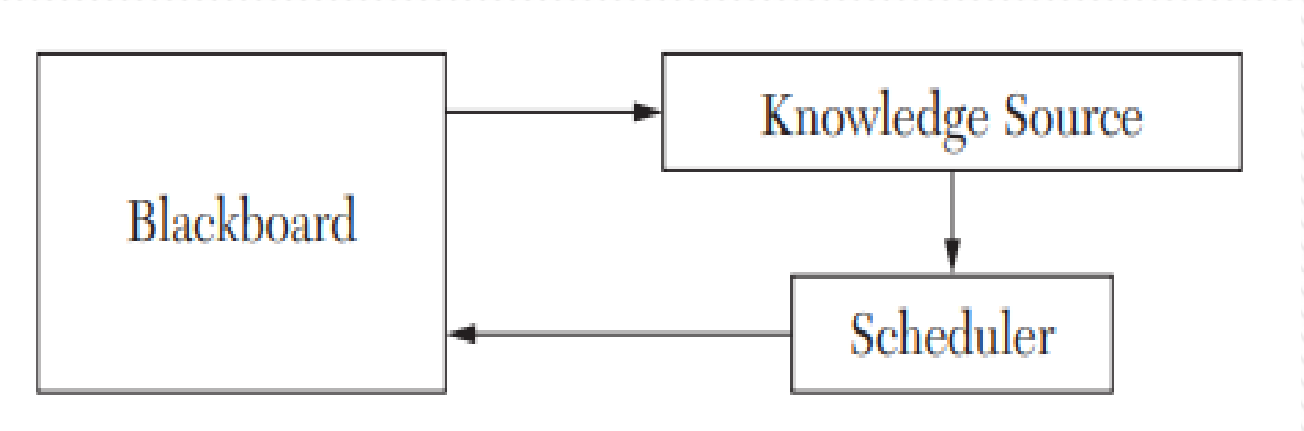
Предимства на архитектурата на дървото на решенията

- Дърветата на решенията могат да бъдат създадени от експерти по темата без нужда от софтуерни специалисти
- Разработването на дървото на решенията обогатява индуктивните и дедуктивните разсъждения, тъй като те се фокусират върху пътищата и резултатите. Тази стойност до голяма степен е разредена с експертна система, тъй като е в ръцете на инженера на знанието, а не на експертите по темата
- Изграждането на дървото на решенията не е фокусирано само върху бизнес логиката, а върху добър диалог и избори, които влияят върху поведението и вземането на решения. Автоматизираният анализ на поведението позволява на дървото на решенията да приеме определена поведенческа динамика
- Нови възли и клонове могат да се добавят към дървото, когато са необходими допълнителни атрибути за по-нататъшно разграничаване между новите обекти. С натрупването на опит стойността, свързана с клоновете, може да бъде променена или системата може да върне по-точни резултати

Архитектура на черна дъска

- Системната архитектура на Blackboard се отнася до специален тип система, базирана на знания, която използва форма на опортюнистично разсъждение
- Тя използва както предна, така и обратна верига и ги избира динамично на всеки етап от процеса на решаване на проблема
- Архитектурата на системата за черна дъска се състои от три функционални компонента:
  - Черна дъска
  - Източник на знания
  - Контролна информация/планировчик

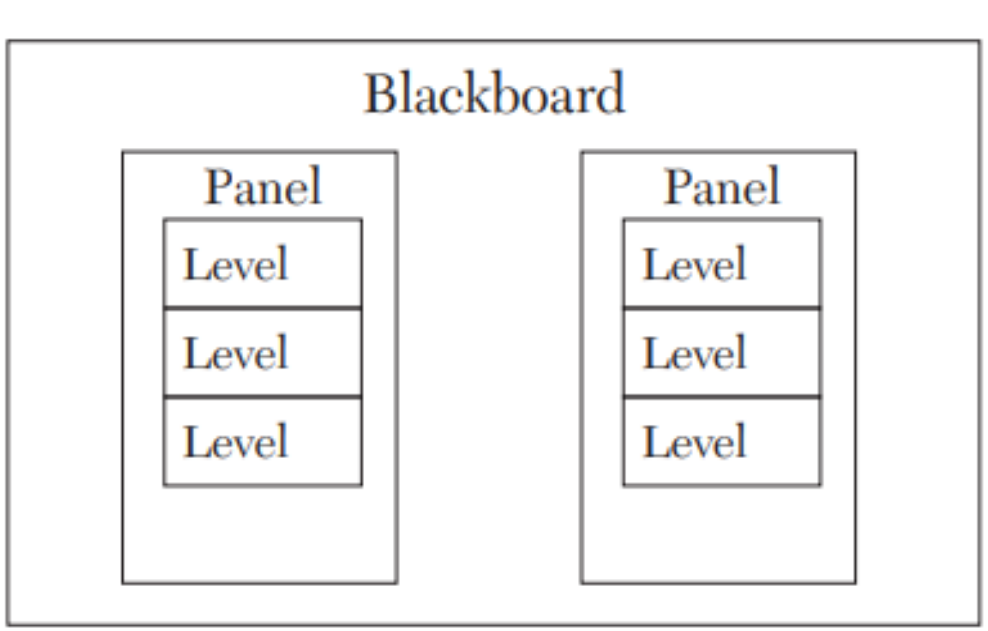
Архитектура на черна дъска



Черна дъска

- Черната дъска е общата структура от данни на източници на знания
- Черната дъска е в състояние да представи всички състояния на някакво проблемно пространство
- Черната дъска съдържа няколко нива на описание по отношение на проблемното пространство
- Тези нива може да имат няколко връзки помежду си, като Е-ЧАСТ-ОТ
- Нивата са части от една и съща структура от данни
- Ако са необходими отделни структури от данни, черната дъска се добавя в панели
- Всеки панел от своя страна може да съдържа няколко нива

Черна дъска


Blackboard
Panel
Level
Level
Level
Panel
Level
Level
Level

Източник на знания

- Това е компонент, който допринася за решението на проблема
- Може да е всичко, което чете от някакво ниво на черната дъска и предполага някаква промяна в части от черната дъска
- Най-често срещаната му форма е правилото за производство
- Източниците на знания са напълно несвързани с други източници на знания

Планировчик

- Определя кой източник на знания получава възможността да смените черната дъска
- Всеки цикъл на изпълнение забелязва промени на черната дъска, активира съответния източник на знания и избира един от тях и го изпълнява
- Например, Hearsay-II е програма за разпознаване на реч. Речта може да се разпознава на няколко нива.

Архитектура на невронни мрежи

- Невронните мрежи са изчислителни системи, моделирани по мрежеста мрежа на човешкия мозък от взаимосвързани обработващи елементи, наречени неврони
- Разбира се, невронните мрежи са много по-прости от човешкия мозък (който се смята, че има повече от 100 милиарда неврона)
- Подобно на мозъка обаче, такива мрежи могат да обработват много части от информация едновременно и могат да се научат сами да разпознават модели и програми, за да решават свързани проблеми сами
- Невронната мрежа е масив от взаимосвързани обработващи елементи, всеки от които може да приема входове, да ги обработва, и произвеждат единичен изход с цел да имитира операцията на човешкия мозък

Архитектура на невронни мрежи

- Знанието се представя в невронните мрежи чрез модела на връзките, образувани по време на обработката на елементите и чрез настройване теглата на тези връзки
- Силата на невронните мрежи е в приложенията, които изискват сложно разпознаване на модели
- Най-голямата слабост на невронните мрежи е, че те не дават обяснение за изводите, които правят
- Невронна мрежа може да бъде обучена да разпознава определени модели и след това да приложи наученото към нови случаи, където може да различи моделите.

Жизнен цикъл на една експертна система

- Тук се обсъжда жизнен цикъл на експертна система и ние очертаваме задачите и дейностите, които трябва да се изпълняват на всеки етап от разработката
- Жизненият цикъл подчертава ролята на алтернативните парадигми за развитие и значението на социалните и организационни характеристики при трансфера на системата към потребителите
- Има пет основни етапа в развитието на експерт система
- Всеки етап има свои собствени уникални характеристики и връзка с другите етапи

Жизнен цикъл на една експертна система

- Тук се обсъжда жизнен цикъл на експертна система и ние очертаваме задачите и дейностите, които трябва да се изпълняват на всеки етап от разработката
- Жизненият цикъл подчертава ролята на алтернативните парадигми за развитие и значението на социалните и организационни характеристики при трансфера на системата към потребителите
- Има пет основни етапа в развитието на експерт система
- Всеки етап има свои собствени уникални характеристики и връзка с другите етапи

Етап на идентификация

- Първата стъпка в придобиването на знания за експертна система е да се характеризират важните аспекти на проблема
- Включва:
  - Идентифициране на участниците
  - Характеристики на проблемите, ресурсите и целите

Идентификация и роли на участниците

- Преди да започнем усвояването на знания, трябва да изберем участниците и техните роли
- Обикновено това е взаимодействието между един експерт в областта и един инженер по знанието
- Процесът на придобиване на знания може да включва и други участници
- Те могат да бъдат експерти по множество области и множество инженери на знания

Идентификация на проблем

- След като сме избрали инженера на знанието и домейна
- експерт, те могат да пристъпят към идентифициране на разглеждания проблем
- Това включва неформален обмен на мнения по различни аспекти на проблема, неговата дефиниция, характеристики и подпроблеми
- Целта е да се характеризира проблемът и неговата поддържаща структура от знания, така че да може да започне развитието на базата от знания

Идентификация на ресурса

- Необходими са ресурси за придобиване на внедрените в системата знания и тестване
- Типичните ресурси са източници на знания, време, изчислителни съоръжения и пари

Идентификация на целта

- Най-вероятно експертът в областта ще идентифицира целите или задачите на изграждането на експертната система в хода на идентифицирането на проблема
- Полезно е да се отделят целите от конкретните задачи на проблема

Етап на концептуализация

- Ключовите понятия и връзки, споменати по време на етапа на идентификация, са изрични по време на етапа на концептуализация
- Може да е полезно за инженера на знанието да диаграмира тези концепции и връзки
- Преди да продължите с процеса на концептуализация, трябва да отговорите на следните въпроси:

Етап на концептуализация

- Какви видове данни са налични?
- Какво се дава и какво се извежда?
- Имат ли имена подзадачите?
- Стратегиите имат ли име?
- Има ли разпознаваеми частични хипотези, които обикновено се използват?
- Как са свързани обектите в домейна?
- Какви процеси участват в решаването на проблема?
- Какви са ограниченията за тези процеси?
- Какъв е информационният поток?

Етап на формализиране

- Процесът на формализиране включва картографиране на ключови концепции, подпроблеми и характеристики на информационния поток, свързани по време на концептуализацията, в по-формални представяния, базирани на различни инструменти или рамки за инженерство на знания
- Инженерът на знанието сега поема по-активна роля, като разказва на експерта в областта за съществуващите представяния на инструменти и типове проблеми, които изглежда отговарят на проблема, ако в резултат на неформална експериментира с предварителен прототип, инженерът на знанието вярва, че има тясно съответствие със съществуващ инструмент или рамка

Етап на изпълнение

- Внедряването включва картографиране на формализираните знания от предишния етап в репрезентативната рамка, свързана с инструмента, избран за проблема
- Тъй като знанията в тази рамка са последователни и съвместими и са организирани, за да дефинират конкретен контролен и информационен поток, то се превръща в изпълнима програма
- Инженерът на знанието разработва полезно представяне на знанията и го използва за разработване на прототип на експертна система
- Базата от знания на прототипа се реализира чрез използване на всички средства за инженерство на знания, които са налични за представянето (редактори, интелигентни редактори или програми за придобиване)
- Когато съществуващите помощни средства са неадекватни, инженерът на знанието трябва да разработи нови

Етап на тестване

- Етапът на тестване включва оценка на прототипната система и формулярите за представяне, използвани за нейното прилагане
- След като прототипната система работи от началото до края на два или три примера, тя трябва да бъде тествана с различни примери, за да се определят слабостите в базата от знания и структурата на изводите.
- Елементите, за които обикновено се установява, че причиняват лошо представяне поради неправилни настройки, са входните/изходните характеристики, правилата за извод, стратегиите за управление и тестовите примери
- Тестването дава възможност да се идентифицират слабостите в структурата и изпълнението на системата и да се направят подходящи корекции.

Процес на инженерство на знанията

- Процесът на изграждане на експертна система преминава през няколко етапа
- Той е подобен в много отношения на жизнения цикъл на софтуерното инженерство:
  - Анализ на изискванията: Изискванията на клиентите се установяват.
  - Придобиване на знания: Опитът за решаване на проблеми се прехвърля от някакъв източник на знания към програма.
  - Архитектурно проектиране: Организация на системата на високо ниво
  - Проектиране на системата: Подробно проектиране на (под)системата
  - Реализация: Кодиране
  - Внедряване: Инсталиране, експлоатация и поддръжка.
- Всяка от тези фази включва подходящите тестове за валидиране, проверка и осигуряване на качеството

Придобиване на знания

- Придобиването на знания може да се разглежда като метод, при който инженерът на знанието събира информация главно от експерти, но също и от учебници, технически ръководства, научни статии и други авторитетни източници и превежда тази информация в база от знания, която е разбираема за както машини, така и хора
- Лицето, което предприема придобиването на знания (инженерът на знанието), трябва да преобразува придобитите знания в електронен формат, който може да използва компютърна програма
- В процеса на придобиване на знания за проект на експертна система, инженерът на знанието изпълнява основно четири основни задачи последователно:

Придобиване на знания

- Първо, инженерът гарантира, че разбира целта и целта на предложената експертна система, за да получи усещане за потенциалния обхват на проекта.
- Второ, инженерът развива работни познания за проблемна област, като овладеете нейната терминология чрез търсене нагоре дефиниции в технически речници и терминологични бази данни. За тази задача се идентифицират ключовите източници на знания като учебници, документи, технически доклади, ръководства, кодекс на практиката, потребители и експерти в областта.
- Трето, инженерът на знанието взаимодейства с експерти чрез срещи или интервюта, за да придобие, провери и потвърди техните знания.
- Четвърто, инженерът на знанието създава документ или група документи (в днешно време в електронен формат), които образуват междинен етап в превода на знанието от източника към компютърната програма

Трудности при придобиване на знания

- Придобиването на знания от експерти не е лесна задача
- Следният списък включва някои фактори, които добавят към сложността на придобиването на знания от експерти и трансфера на знания към компютър:

Трудности при придобиване на знания

- Експертите може да не знаят как да изразят знанията си или може да са не може да го направи
- Експертите може да нямат време или да не желаят да си сътрудничат
- Тестването и усъвършенстването на знанията са сложни
- Методите за извличане на знания може да са слабо дефинирани
- Системните създатели са склонни да събират знания от един източник, но съответните знания могат да бъдат разпръснати в няколко източника
- Системните създатели могат да се опитат да съберат документирани знания, вместо да използват експерти. Събраните знания може да са непълни
- Трудно е да се разпознаят конкретни знания, когато са смесени с неподходящи данни
- Експертите могат да променят поведението си, когато бъдат наблюдавани или
- интервюиран
- Проблемните междуличностни комуникационни фактори могат да засегнат инженера на знанието и експертите

Стратегии за придобиване на знания

- Има няколко начина за придобиване на знания. Някои от най-известните методи са разгледани по-долу:
- **Анализ на протокола**: Това е набор от техники, известни като анализ на вербален протокол. Това е метод, чрез който инженерът на знанието придобива подробни знания от експерта. Протоколът е запис или документация за поетапната обработка на информацията и поведението на експерта при вземане на решения. Експертът е помолен да говори за нещо навън високо, докато изпълнявате задачата или решавате проблема под наблюдение. При този метод инженерът на знанието не прекъсва, докато експертът работи

Наблюдения

- В много отношения това е най-очевидният и ясен подход към придобиването на знания
- При този метод инженерът на знанието наблюдава експерта, изпълняващ задача
- Това не позволява на инженера на знанието да се намеси неволно в процеса, но го прави не дават никаква представа защо се вземат решения.

Анализ на интервю

- Това е изрична техника, която се появява в няколко варианта
- Тя включва директен диалог между експерта и инженера на знанието
- Процесът на интервюто може да бъде досаден
- Това поставя големи изисквания към експерта в областта, който трябва да може не само да демонстрира експертиза, но и да я изрази
- Интервютата могат да бъдат неструктурирани, полуструктурирани или структурирани
- Успехът на сесията за интервю зависи от зададените въпроси и способността на експерта да изрази своите знания

Интроспекция

» Експертът става инженер на знанието и след това разчита на комбинация от интроспекция и познания за архитектурата на експертната система, за да преобразува ноу-хау в базата от знания.

Обратно обучение

» Инженерът на знанието се опитва да предаде информацията обратно на експерта, който след това предоставя корекции и попълва пропуските

Обобщение

- Това са някои методи, които помагат за придобиване на знания от експерти
- По принцип никой инженер на знанието не се придържа към един метод, но приема комбинация от тези метод
- Инженерът на знанието трябва да се стреми да извлече повече от дълбоките знания, които ще помогнат за разбирането на основите на областта

Предимства на експертните системи

- Наличност: Експертните системи са лесно достъпни поради голямото производство на софтуер.
- Скорост: Експертните системи предлагат голяма скорост. Те намаляват количеството работа, която човек полага.
- Нисък процент на грешки: процентът им на грешки е по-нисък от този на хората.
- Постоянен отговор: Те работят стабилно, без да се чувстват емоционални, напрегнати или уморени, докато човешките експерти, под стрес, в лошо настроение или когато времето е ограничено, или правят погрешни предположения, или забравят съответните фактори.
- Възпроизводимост: Могат да се правят много копия на експертна система, но обучението на нови човешки експерти отнема време и е скъпо, докато времето за дублиране на експертна система е много кратко

Предимства на експертните системи

- Възстановяване: Експертните системи могат да се комбинират с други системи или познания за бази данни за справяне с по-сложни ситуации.
- Намаляване на риска: Те могат да работят в среди, опасни за хората.
- Последователност: С експертните системи подобни транзакции се обработват по същия начин. Системата ще направи сравними препоръки за подобни ситуации.
- Многоизмерност: Експертната система играе три основни роли: ролята на решаващ проблеми, преподавател и архив. Въпреки че системните интерфейси на естествен език са много примитивни, днешните експертни системи изпълняват тези роли много добре. Експерт човек, който е добър решаващ проблеми, не трябва да бъде добър учител.
- Ефективност: Експертна система прави нещата по-ефективни, като намалява времето, необходимо за решаване на проблеми. Експертните системи предоставят стратегически и сравнителни предимства, които могат да създадат проблеми за конкурентите.

Ограничения на експертните системи

- Различните видове многоизмерни проблеми, с които се сблъскват различните потребители по време на извършване на дейности, не могат да бъдат ефективно преодоляни от експертни системи
- Експертните системи не реагират добре на ситуации извън техния обхват на експертиза
- Някои от типичните експертни системи понякога не са в състояние да направят достъпно здраво знание и широкообхватна контекстуална информация
- Няма гъвкавост или способност за адаптиране към променящата се среда

Ограничения на експертните системи

- Процесът на изграждане на експертна система е трудоемък. В момента са необходими много ресурси
- Експертните системи се фокусират върху много специфични теми, като компютърни грешки, радиология и диагностични умения. Основната причина за тази ситуация е трудността при извличане на знания и изграждане и поддържане на голяма база от знания.
- Проверката на коректността на всяка голяма компютърна система е трудна за доказване, а експертните системи са особено трудни за проверка. Това е сериозен проблем, тъй като технологията на експертната система се прилага към критични приложения като контрол на въздушното движение, операции с ядрен реактор и оръжейни системи
- Има малко поуки от опита. Текущите експертни системи са ръчно изработени; след като системата е завършена, нейната производителност няма да се подобри без допълнително внимание от страна на програмистите

Примери за експертни системи

» Следват някои успешни експертни системи в различни области, които не само помогнаха за пионер в разработването на нови техники и инструменти, но също така доказаха, че системите за изкуствен интелект могат да бъдат ужасно успешни в избрани области на експертиза.

DENDRAL

- Разработена в Станфорд през 60-те години. DENDRAL беше една от първите системи, които се конкурираха с производителността на експертите по домейни
- DENDRAL съхранява и аргументира със знания от областта на органичната химия, използвайки планирана парадигма за генериране-тест за търсене
- По-конкретно, задачата беше да се определи молекулярната структура на органичната молекула
- Получава като свой вход молекулярна формула с набор от ограничения, които служат за ограничаване на възможните взаимовръзки между атомите
- Генерира се списък с всички възможни начини за сглобяване на атомите в молекули
- Те са подредени с помощта на базата от знания, за да се правят тестуеми прогнози за кандидат-молекулите.

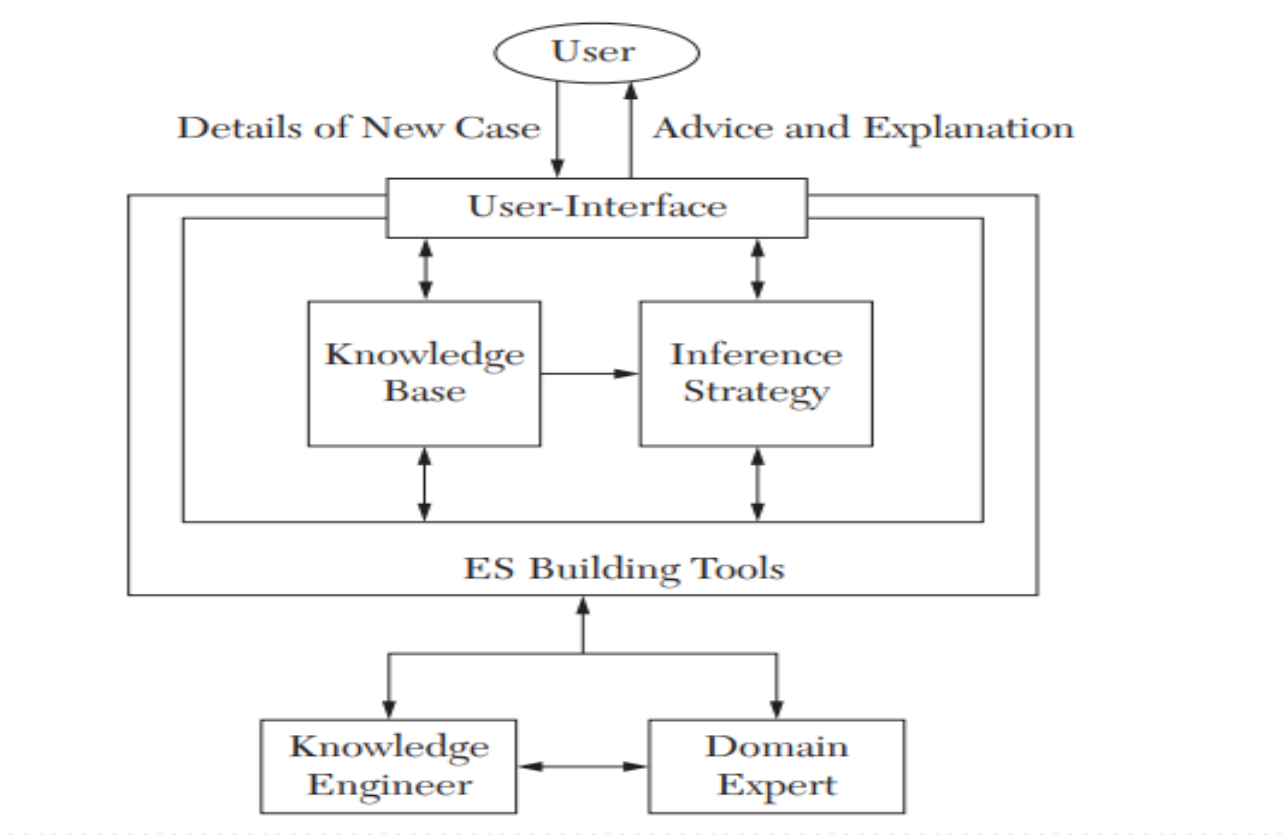
DENDRAL

- Това позволява съкращаването на списъка с кандидати. Тъй като органичните молекули обикновено са много големи, броят на възможните структури за тези молекули обикновено е огромен. DENDRAL се занимава с проблема на това голямо пространство за търсене, като прилага евристични познания на експертни химици към проблема за изясняване на структурата. Методите на DENDRAL се оказаха изключително ефективни. META-DENDRAL добави способност за машинно обучение под формата на индуктивно обучение по правила
- базиран на алгоритъм за изкачване на хълм, използващ необработени масспектрографски данни. След това новата евристика беше използвана за извеждане на структурата на неизвестни молекули от техните масови спектри. Въпреки че METADENDRAL вече не е активна програма, нейният принос към идеите за учене и откриване се прилага в нови области

MYCIN

- Станфорд също беше дом на друга влиятелна експертна система, наречена MYCIN. MYCIN създаде методологията на съвременните експертни системи. MYCIN първоначално е написан на INTERLISP, диалект на езика за програмиране LISP.
- MYCIN е медицинска експертна система, която помага на лекар, който не е такъв експерт в областта на антибиотиците при лечение на кръвни инфекции.
- MYCIN се състои от пет модула:
  - знание
  - база данни за пациенти
  - консултативна програма
  - програма за обяснение
  - програма за придобиване на знания

MYCIN


User
Details of New Case
Advice and Explanation
User-Interface
Knowledge Base
Inference Strategy
ES Building Tools
Knowledge Engineer
Domain Expert

MYCIN

- Знанието е организирано като серия от правила АКО-ТОГАВА
- Определени фактори могат да бъдат свързани със знанието
- Информацията за пациента се съхранява в контекстуална форма
- Това включва данни като кръвни проби, скорошни оперативни процедури и лекарства
- Изборът се извършва след поставяне на диагнозата
- Състои се от избор на кандидат-лекарства и след това избор на предпочитан антибиотик
- Самият MYCIN никога не е бил използван в клинични условия, но наследниците на програмата са били

EMYCIN (Empty MYCIN)

- Тази система позволява архитектурата на MYCIN да бъде приложена към друга медицинска област, освен кръвните заболявания
- Това не е архитектура за решаване на проблеми с общо предназначение, а е по-подходяща за диагностични задачи в медицината
- Системата PUFF беше първата програма, създадена с помощта на EMYCIN
- Домейнът на PUFF е интерпретацията на тестове за белодробна функция при пациенти с белодробни заболявания
- Програмата може да диагностицира наличието и тежестта на белодробно заболяване и да изготвя доклади за досието на пациента
- Програмата за придобиване на знания TEIRSIAS е създадена, за да помогне на експертите в областта да усъвършенстват базата от знания EMYCIN
- TEIRSIAS разработи концепцията за знание на мета-ниво, т.е. знание, чрез което една програма не може само да използва знанията си директно, но може да ги изследва, да разсъждава за тях и да насочва използването му

PROSPECTOR

- Класическа експертна система, която определя вероятното местоположение и вида на рудните находища, въз основа на геоложка информация за даден обект
- PROSPECTOR се опитва да предскаже минералите, които ще бъдат открити там
- Подобно на MYCIN, PROSPECTOR е система, базирана на правила, която използва фактори за сигурност, за да представи силните страни на правилата
- PROSPECTOR се занимава с геоложки настройки, структурни контроли и видове скали, минерали и алтернативни продукти, които присъстват или се предполагат
- Сравнява наблюденията със съхранени модели на рудни находища, отбелязва приликите, разликите и липсващата информация, изисква допълнителна информация, ако е необходимо, и след това оценява минералния потенциал на перспективата


Благодаря за вниманието!