Характерни особености на системите с изкуствен интелект

Лекция 2

- **План**
- **Предназначение на СИИ;**
- **Класификация и подходи за създаване на СИИ;**
- **Примери за задачи, които СИИ решават;**
- **Сравнителна таблица на програмните системи – СИИ / Обикновени програмни системи.**
- **Най-типичният представител на СИИ – Експертните системи (ЕС)**
- **Знания. Логически извод. Пример – родословно дърво;**
- **Особености на знанията, метазнания;**
- **Дедукция.**
- **Индукция – пълна и непълна.**
- **Евристика – подход за решаване на сложни задачи. Примери.**
- **Свойства на евристиката.**
- **Търсим универсален подход за решаване на задачи.**
- **Примерна стратегия за построяване на последователност от оператори (GPS).**
- **Опростена схема за моделиране на интелектуална дейност.**

Предназначение на СИИ - да решават задачи въз основа на използуване на знания.

Три гледни точки за разработването на СИИ:

- **СИИ се използват за разработване на модели и методи за решаване на задачи, които традиционно се считат за интелектуални.**
- **СИИ се използват за разработване на принципно нова технология за програмиране и нови архитектури на ЕИМ.**
- **СИИ се използват за създаване на системи за решаването на задачи, които до сега не е можело да се решат с ЕИМ.**

Оформяне на два подхода за създаване на СИИ:

- ***Когнитивен* – създават се системи за обработка на знания;**
- ***Конекционистичен* – създават се невронни мрежи, които моделират човешкото знание;**

Класификация

- **системи с общо предназначение;**
- **специализирани системи;**

Примери за такива задачи:

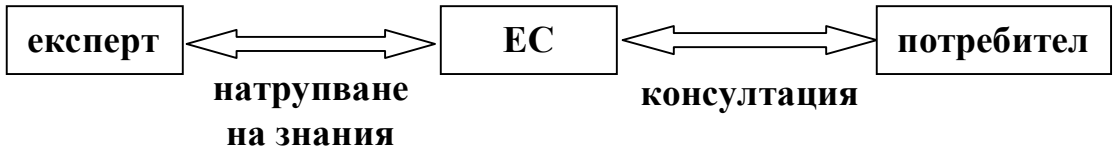
- Разбиране и синтез на текстове на естествен език;
- Възприемане и синтез на реч;
- Анализ, обработка и синтез на изображения;
- Превод от един на друг естествен език;
- Вземане на решения при непълна информация или в условията на изменящи се околни условия;
- Автоматизирано проектиране;
- Автоматизирано създаване на планове;
- Възприемане, разпознаване, изразяване на емоции;
- Доказателство на теореми;
- Представяне и обработка на знания;
- Обучение;
- Моделиране и изследване на поведение;
- Игрови стратегии.

Сравнителна таблица на програмните системи

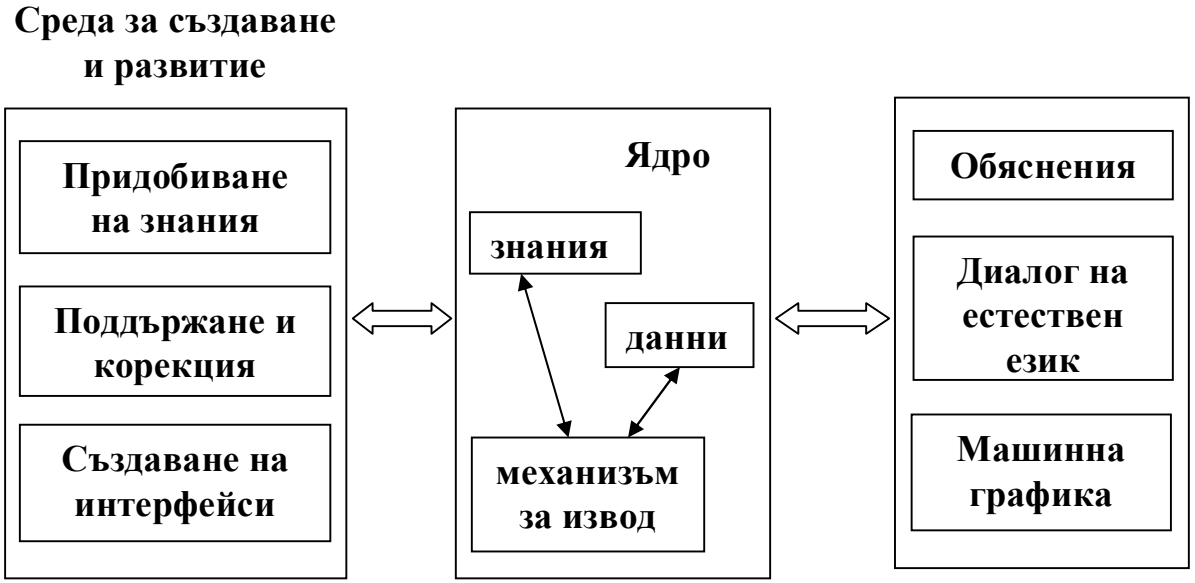
СИИ	Обикновени програмни системи
База от знания	База от данни
Обработка на знания	Обработка на данни
Преобладаване на :	
Декларативна спецификация	Процедурна спецификация
Дедуктивен механизъм	Алгоритъм
Индуктивен механизъм	
Евристичен подход за решаване на сложни задачи	

Най-типичния представител на СИИ са Експертните Системи (ЕС).

ЕС е програма, която въз основа на структурирани по определен начин знания решава сложни практически задачи, обосновава и обяснява своите решения, натрупва нови знания, реализира смислен диалог.

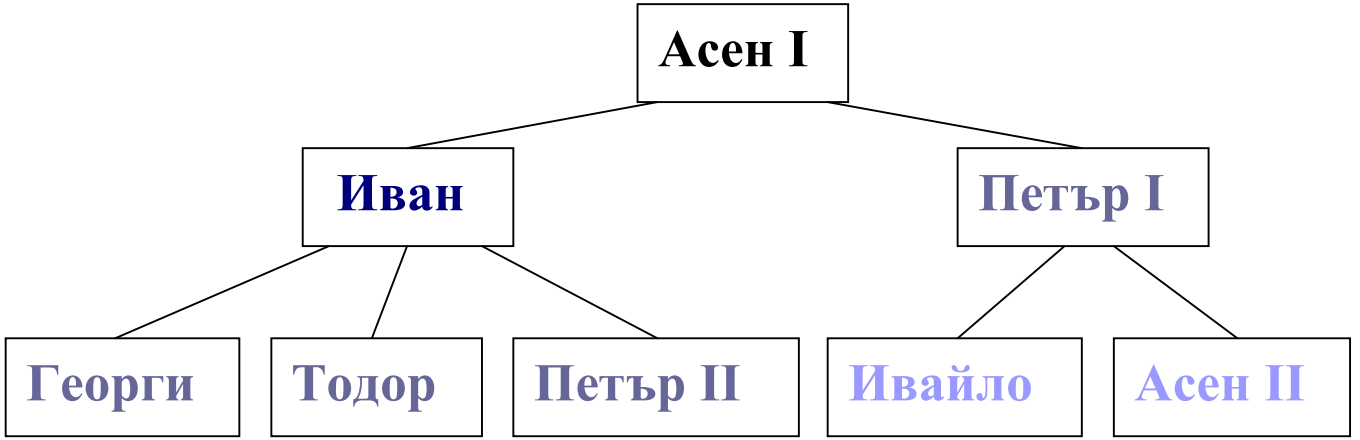


Структура на ЕС



База данни

Нека разгледаме следното родословно дърво от второто Българско царство.

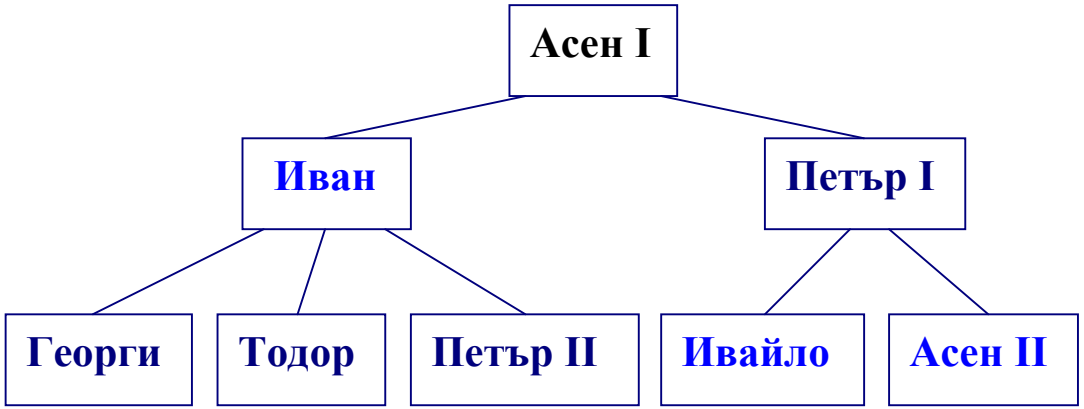


Сега да опишем това родословие в по-удобен за компютъра вид, като **използуваме низове от символи (стрингове)**. Обърнете внимание, че е използувана само една роднинска връзка – "е баща". Нека наречем нашето описание **база данни.**

Асен I е баща на Иван
Асен I е баща на Петър I
Иван е баща на Георги
Иван е баща на Тодор
Иван е баща на Петър II
Петър I е баща на Ивайло
Петър I е баща на Асен II

Знания

А сега да приведем няколко **правила за роднински връзки** и да ги групираме в структура, която ще наречем **База знания:**



База данни

Асен I	е баща на Иван
Асен I	е баща на Петър I
Иван	е баща на Георги
Иван	е баща на Тодор
Иван	е баща на Петър II
Петър I	е баща на Ивайло
Петър I	е баща на Асен II

Правило 1:
Ако 1) A и B са братя
2) A е баща на C

То: B е чичо на C

Правило 2:
Ако 1) A е баща на B
2) B е баща на C

То: A е дядо на C

Правило 3:
Ако: 1) A е баща на B
2) A е баща на C

То: B и C са братя

Логическият извод е процес на извеждане на заключения от данни и знания. Ето примери за реализиране на логически извод.

- ***Цел:*** **Иван е чичо на Ивайло?**

От цел към факти:

Намираме целта в правило 1.

Проверка на 1 – **Петър I е баща на Ивайло.**

Проверка на 2 – **Петър I и Иван са братя**

Намираме целта в правило 3

- Проверка на условия 1 и 2 на правило 3
- **Целта е постигната.**

Асен I е баща на Иван
Асен I е баща на Петър I
Иван е баща на Георги
Иван е баща на Тодор
Иван е баща на Петър II
Петър I е баща на Ивайло
Петър I е баща на Асен II

База данни

B=Иван
C=Ивайло
A=Петър I
D=Асен I

База знания

Правило 1:
Ако 1) **A е баща на B**
2) **A и B са братя**
То: **B е чичо на C**

Правило 2:
Ако 1) D е баща на B
2) B е баща на E
То: D е дядо на E

Правило 3:
Ако: 1) **D е баща на Иван**
2) **D е баща на Петър I**
То: A и B **са братя**

Логическият извод е процес на извеждане на заключения от данни и знания. Ето примери за реализиране на логически извод.

Цел: Иван е чичо на Ивайло?

- ***От факти към цел:***
- ***Братя са :***
  (Петър I и Иван);
  (Георги, Тодор, Петър II),
  (Ивайло, АсенII)
- ***Бащи са:***
  (АсенI на Иван);
  (Асен I на Петър I);…
  (Петър I , на Ивайло)
- **Следва извода.**

A= Петър I
B= Иван
C= Ивайло

База знания

Правило 1:
Ако 1) A и B са братя
2) A е баща на C
То: **B** е **чичо** на **C**

Правило 2:
Ако 1) D е баща на B
2) B е баща на E
То: D е дядо на E

Правило 3:
Ако: 1) D е баща на A
2) D е баща на B
То: A и B са братя

Особености на знанията.

а) вътрешна интерпретируемост – всяка информационна единица има свое име, то е уникално и може да я идентифицира.

б) структурираност–отношения от типа “част-цяло”, “род-вид”, “елемент-клас”.

в) свързаност – отношения от типа “причина - следствие”, “аргумент - функция”, “да бъдат едно до друго”.

г) семантична метрика – близост в семантичен смисъл
човек, маймуна, мислене – *мисленето е много по-близко до човека отколкото до маймуната*.

д) активност - до сега програмите са активни - данните са пасивни,
сега знанията са активни – изпълнението на програмата може да се активира от състояния на знанията.

Метазнания.

Метазнанията са знания за самите знания. Те описват знанията от различни гледни точки, например:

а) Кога и как ще се използват;

б) Как се групират;

в) Колко често се използват;

г) Имат ли отношение към даден проблем или не и т.н..

Метазнанията са полезни от гледна точка на поетапно изграждане на базата от знания; постигане на по-висока ефективност при използване на знанията; обясняване същността на знанията и т.н.

Правило (знание):

Ако има утечка от сярна киселина; **То**: да се използува вар.

Обяснение (метазнание): *Варта неутрализира киселината*.

Дедукция

От истинността на общото правило се прави извод за истинността на частния случай.

- ***Пример:***
- **Общо правило**: Всички метали са електропроводими.
- **Частен случай**: Желязото е метал.
- **Разсъждение и извод**: Ако всички метали са електропроводими и желязото е метал, то желязото е електропроводимо.

Всички **a** имат свойството B
Нито един елемент от C няма свойството B
$\therefore$ Нито един елемент от A не принадлежи и на C

a има свойството B
a принадлежи на множеството C
$\therefore$ Някои елементи на C имат свойство B

Всички елементи от A имат свойството ***B***
$x \in A$
$\therefore$ ***x*** има свойство ***B***

Индукция

Въз основа на знанията за отделните елементи от един клас се прави извод за общи свойства на елементите от класа.

Пълна индукция – общ извод се прави за целия клас въз основа на знанията за всички предмети от класа.

Всяко **а** от S_1 има свойството **в**,
всяко **а** от S_2 има свойството **в**,
S_1 и S_2 изчерпват цялото S.
$\therefore$ Всяко **а** от ***S има свойството*** **в.**

Непълна индукция - Прави се общ извод за целия клас въз основа на знанията за някои представители от класа.

A_1 има свойство **в**
A_2 има свойство **в**
A_3 има свойство **в**
$\therefore$ ***и A_4 има свойство*** **в**
и всички А имат свойство **в**

Качествени оценки за ***Кизследвани*** елемента
< 25% без увереност
≈ 50% с увереност
> 75% с голяма вероятност
90% почти сигурно

вероятност
мощност на цялото множество **К**
мощност на изследваната част **Кизсл.**

A_4 има свойството **в** с вероятност
К/Кизсл.

Евристика

Подход за решаване на сложни задачи с много варианти, при които въз основа на разумни критерии проверката се ограничава до малък брой от възможните варианти на решението.

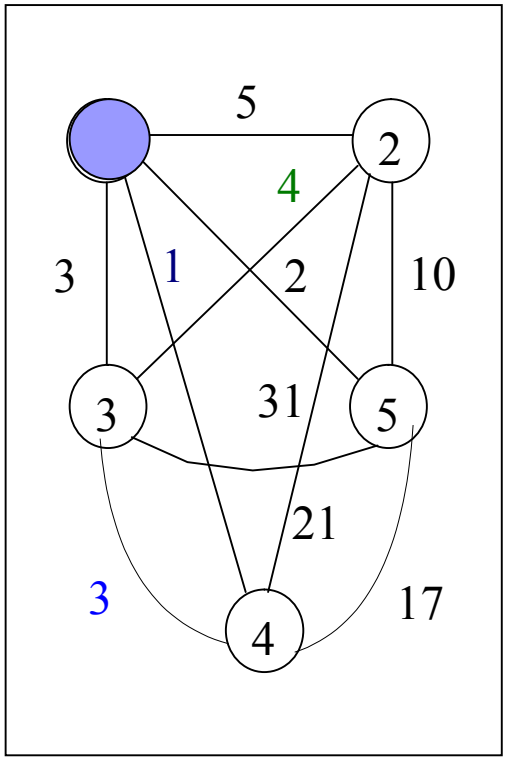
Евристиката е рационална идея за бързо намиране на решение.
Основава се на опит и интуиция. Решенията не винаги са оптимални;
Решението може и да не се намери.

Евристика се прилага при:
а) задачи, за които не са известни алгоритми
б) известните алгоритми изискват изпробването на голям брой варианти.

Пример
Разглеждаме задачата за търговския пътник.

Зададени са N града и цената на пътищата.
Да се определи маршрут с минимална цена, който има начало и край един и същ град. Градовете трябва да се обходят еднократно.

Разглеждаме задачата за търговския пътник.



Два варианта за решаване:

1) Метод на пълното изброяване.

2) Използва се правилото:

Ако: 1. намираме се в град j;
2. град i не е обходен;
3. цената на пътя $C_{j,i}$ е минимална в сравнение с цената на останалите не обходени градове.

То: избира се за следващ град i.

Маршрут: 1; 4; 3; 2; 5; 1;

Цена: 1 + 3 + 4 + 10 + 2 = 20

Пример за евристика:

Един човек има x-зайци и y-кокошки
Броят на главите е A=70
Броят на краката е B=180
Колко са зайците и колко кокошките?

Решение:

Съставяме и решаваме системата уравнения:

$x + y = A$

$4x + 2y = B$

Евристика:

Нека кокошките застанат на един крак, а зайците на задните си крака. Тогава сумата от краката на земята е $\frac{B}{2}$.

Но броя зайци е $\frac{B}{2}$ минус броя глави A т.е.:

$$\frac{B}{2} - A = x$$

Свойства на евристиката

1. Те удовлетворяват метода на планирането или при разчленяването на задачата на подзадачи - принцип за редукция на подцелите.
2. Ограничават броя на изпробваните варианти.
3. За разлика от алгоритмите не гарантират решаването на задачата (могат да отклонят в страни)
4. Използването им е високоефективно. Добрите евристики могат да ускорят работата в сравнение с алгоритмите на няколко порядъка.
5. Евристиките могат да се разглеждат като теория за поведението на човека при решаването на задачи.

Неформални методи за решаване на задачи

Търсим универсален подход за решаване на задачи, който да е приложим към всяка задача!!!

Обекти – физически или абстрактни обекти и въображаеми ситуации;

Оператори – действия, правила и преобразувания, които могат да се използуват.

Да се намери последователност от оператори, която трансформира един обект в друг обект.

Необходимо е да се намери процедура, която:

- да разпознава идентичност на два обекта;
- да определя различието между два обекта (а за това е необходимо подходящо описание на характеристиките на обектите).
- да използва различията за избор на подходящ оператор (следователно за всеки оператор трябва да се знае на кои характеристики и как влияе)

Необходимо е да се отдели описанието от стратегията на самото търсене на решение.

Всяка задача може да се опише чрез:

Семейство еднотипни задачи

1.Множество от обекти
2.Набор (списък) от оператори
3.Процедура за определяне на различия между обекти
4.Таблица на връзките “оператор-изменяеми характеристики” на обектите.

Конкретна задача

5.Конкретен обект, от който тръгват разсъжденията (начална ситуация)
6.Конкретен обект, който представлява желаната цел.

Примерна стратегия за построяване на последователност от оператори (GPS)

Процедура1: Преобразуване на обект А (начална ситуация) в обект В (цел):

-установяване на разликата между А и В (ако те са идентични край);

-избира се най-подходящ оператор за отстраняване на най-съществена разлика.

-Избраният оператор се прилага към А и се получава нов обект С (преминава се към най-горната стъпка като се работи вече с С вместо с А).

Процедура 2: Прилагане на оператор Р към обект А.

-изходният обект А се преобразува така, че към него да може да се приложи оператор Р.

-непосредствено прилагане на Р към А.

Последователно разлагане на подзадачи. (*преди да се направи еди какво си, е необходимо да се направи еди що си и т.н. до изчерпване*).

При погрешен ход да можем да се върнем към началното състояние.

Пример: Дефиниране на задача в пространство на състоянията. Вълкът, Козата, Фермерът и Зелката

- Фермерът и неговата овца, вълкът и зелката отиват на реката и искат да я преминат. Има лодка, но с места само за двама. Фермерът единствено може да гребе затова той трябва да пренесе със себе си още някого. Ако козата и зелката останат на едната страна на реката заедно, козата ще изяде зелката. Ако вълкът и козата останат на единия бряг на реката, то вълкът ще изяде козата. Налагат се серия от пресичания на реката така че всички да оцелеят.

- Състоянието на системата е означено със структурата **STATE** и тя показва къде се намират фермера, вълкът, козата и зелката. Целта е да се намери решение как да се достигне от началното състояние до крайното състояние чрез серия от валидни (безопасни) междинни състояния.

- Валидните състояния са ограничени от предиката **“unsafe”.**

- Изисква се също едно състояние да се състои само веднъж. Заради това изискване се формира списък на състоянията които вече са се състояли. **Гледат се положенията и на четиримата едновременно!**

- Предикатът **“go”** се извиква със началното и крайното състояние. В началното състояние всички са на източния бряг. В крайното състояние всички са на западния бряг.

- **state** е функтор който съдържа мястото на фермера, вълка, козата и зелката. Дали са на източния или на западния бряг.

- **Променливи – X, Y, F, W, G, C.**
- **Стойностите на променливите са две – east или west.**
- **Обекти –** фермерът (F), вълкът (W), козата (G), зелката (C);
- **Състоянието на системата** е означено със структурата **STATE. Тя има четири променливи. Всяка променлива е за един от обектите – фермер, вълк, коза и зелка.**
  - **STATE (**F, W, G, C**).**
- **Позициите на обектите в структурата винаги са следните:**

  ***Първа позиция показва дали фермерът е на източния или западния бряг**

  ***Втора позиция показва дали вълкът е на източния или западния бряг**

  ***Трета позиция показва дали козата е на източния или западния бряг**

  ***Четвърта позиция показва дали зелката е на източния или западния бряг**
- **X и Y са допълнителни променливи.**
  - Те може да се поставят на всяка позиция в структурата STATE.
  - Те дори може да се поставят на повече от една позиция в структурата STATE.
  - Стойностите на X и Y може да служат за да се постави нова стойност на обекта на чиято позиция са поставени. (източен или западен бряг)
  - Стойностите на X и Y може да служат и за да се провери дали съответния обект е на източния или западния бряг.
  - Ако тези променливи са поставени на две места в структурата то може да се провери дали съответните обекти са на един и същи бряг или не. Това е възможно защото всяка променлива може да има само една стойност в даден момент. Не може една променлива да има две различни стойности в един и същи момент.

- **Начално състояние STATE(east,east,east,east) - всички обекти са на източния бряг**

- **Крайно състояние STATE(west,west,west,west) - всички обекти са на западния бряг**

- **Целта** е да се намери решение как да се достигне от началното състояние до крайното състояние чрез серия от валидни (безопасни) междинни състояния.

- **Първото опасно състояние е следното**: **state(F,X,X,_) :- opposite (F, X)** – при него вълкът ще изяде козата защото двата X означават че те са от една и съща страна на реката, а в същото време стойностите на F и X са различни т.е. фермерът е на другия бряг на реката.
- **Второто опасно състояние е следното: state(F,_,X,X) :- opposite (F, X)** – при него козата ще изяде зелката защото двата X означават, че те са от една и съща страна на реката, а в същото време стойностите на F и X са различни т.е. фермерът не е на същия бряг на реката.

- Оператор за определяне на противоположната страна на реката:
  - opposite(east,west) – **east** е противоположно на **west**

■**Оператори за забрана на избирането на опасните състояния:**

- unsafe(state(F,X,X,_)):- opposite(F,X),!. %опасно състояние. Вълкът ще изяде козата.
- unsafe(state(F,_,X,X)):- opposite(F,X),!. %опасно състояние. Козата ще изяде зелката.

■**Оператори за преместване в следващото безопасно състояние:**

- ■move(state(X,X,G,C), state(Y,Y,G,C)):-opposite(X,Y). %фермерът + вълкът
- ■move(state(X,W,X,C), state(Y,W,Y,C)):-opposite(X,Y). %фермерът + козата
- ■move(state(X,W,G,X),state(Y,W,G,Y)):-opposite(X,Y). %фермерът + зелката
- ■move(state(X,W,G,C),state(Y,W,G,C)):-opposite(X,Y). %фермерът

Възможните безопасни състояния са следните:

STATE (east, east, east, east)
STATE (west, east, west, east)
STATE (east, east, west, east)
STATE (west, east, west, west)
STATE (east, east, east, west)
STATE (west, west, east, west)
STATE (east, west, east, west)
STATE (west, west, west, west)

■**Запомняне в списък на състоянията, които са минали за да не се повтори някое.**

■**Оператори за проверка дали новото състояние не се е случвало преди.**

- ■member(X,[X|_]):- !.
- ■member(X,[_|L]):- member(X,L).

Стратегии и/или алгоритми за прилагане на операциите над проблемната област. Механизъм за управление.

```
go(StartState, GoalState):-
    path(StartState, GoalState, [StartState], Path),
    write("Решението е: "),nl,
    write_path(Path).

path(StartState, GoalState, VisitedPath, Path):-
    move(StartState, NextState),                %търси следващо състояние
    not(unsafe(NextState)),                     %безопасно ли е състоянието
    not(member(NextState, VisitedPath)),        %повтаря ли се състоянието
    path(NextState,GoalState,[NextState|VisitedPath], Path),!.
path(GoalState, GoalState, Path, Path).  %достигнато е крайното състояние
```

Описаните по този начин правила се изпълняват от компилатора на Пролог, който има вграден механизъм за логически извод, за унификация и backtracking !

```
go(StartState, GoalState):-
    path(StartState, GoalState, [StartState], Path),
    write("Решението е: "),nl,
    write_path(Path).

path(StartState, GoalState, VisitedPath, Path):-
    move(StartState, NextState),              %търси преместване
    not(unsafe(NextState)),                   %безопасно ли е състоянието
    not(member(NextState, VisitedPath)),      %повтаря ли се състоянието
    path(NextState,GoalState,[NextState|VisitedPath], Path),!.
path(GoalState, GoalState, Path, Path).  %достигнато е крайното състояние

move(state(X,X,G,C), state(Y,Y,G,C)):-opposite(X,Y). %фермера + вълкът
move(state(X,W,X,C), state(Y,W,Y,C)):-opposite(X,Y). %фермера + козата
move(state(X,W,G,X),state(Y,W,G,Y)):-opposite(X,Y).  %фермера + зелката
move(state(X,W,G,C),state(Y,W,G,C)):-opposite(X,Y).  %фермера

opposite(east,west).
opposite(west,east).

unsafe(state(F,X,X,_)):- opposite(F,X),!.   %вълкът ще изяде козата
unsafe(state(F,_,X,X)):- opposite(F,X),!.   %козата ще изяде зелката

member(X,[X|_]):- !.
member(X,[_|L]):- member(X,L).

write_path([H1,H2|T]) :-
    write_move(H1,H2),
    write_path([H2|T]).

write_path([]).
```

```
write_move(state(X,W,G,C), state(Y,W,G,C)) :- !,
   write("Фермерът пресича реката от ",X," до ",Y),nl.
write_move(state(X,X,G,C), state(Y,Y,G,C)) :- !,
   write("Фермерът пренася вълкът от ", X, " до ", Y),nl.
write_move(state(X,W,X,C), state(Y,W,Y,C)) :- !,
   write("Фермерът пренася козата от ", X, " до ", Y),nl.
write_move(state(X,W,G,X), state(Y,W,G,Y)) :- !,
   write("Фермерът пренася зелката от ", X, " до ", Y),nl.
```

Цел:
go(state(west,west,west,west), state(east,east,east,east))

Решението е:
Фермерът пренася **козата** от east до west
Фермерът пресича реката от west до east
Фермерът пренася **зелката** от east до west
Фермерът пренася **козата** от west до east
Фермерът пренася **вълкът** от east до west
Фермерът пресича реката от west до east
Фермерът пренася **козата** от east до west
No solutions

Опростена схема за моделиране на интелектуална дейност

А. Множество обекти. Описание на свойствата, които имат обектите.

-Град (име, брой жители, географски координати);

-Автомобил (марка, двигател, мощност, вид, цвят, скорост);

В. Описание на множеството релации между обектите. Релациите въвеждат определена подредба между обектите.

-Градовете могат да се подредят по броя на жителите, по географски координати и т.н.

-Според отношението родител-дете може да се построи родословно дърво

-кубче (вляво, вдясно, отгоре, отдолу, отпред, отзад) – може да се строят различни фигури от кубчета.

С. Над обектите (техните свойства и релации) могат да се извършват определени операции, в т.ч. да се създават и унищожават обекти.

-Да се провери дадена релация

-Да се установи към кой клас принадлежи конкретен обект.

-Да се измени взаимното разположение на обекта и т.н.

Д. Прилага се механизъм за управление на дейността.

-Определя се последователността за прилагане на правилата

-Тръгва се от начално състояние и се преследва определена крайна цел.

При моделирането се съставят три вида зависимости:

-**Структурни зависимости** -структурно представяне на предметната област;

-**Правила** за описание на операциите;

-**Стратегии и/или алгоритми** за прилагане на операциите над проблемната област. Механизъм за управление.

Видовете модели са:

-**Алгоритмични модели** – При тях се използват известни алгоритми и имаме процедурна реализация;

-**Дедуктивни модели** – При тях не е известна предварително последователността на изпълнение на операторите.