



Inteligența Artificială

Universitatea Politehnica București
Anul universitar 2013-2014

Adina Magda Florea



Curs nr. 9

Invatare automata

- **Tipuri de invatare**
- **Invatarea prin arbori de decizie**
- **Invatarea conceptelor disjunctive din exemple**
- **Invatarea prin cautare in spatiul versiunilor**



1. Tipuri de invatare

- Una dintre caracteristicile esentiale ale inteligentei umane este capacitatea de a învăța
- Învatarea automată este domeniul cel mai provocator al inteligentei artificiale si, în acelasi timp, cel mai rezistent încercărilor de automatizare completă



Reguli de inferenta utilizate in invatare

- La baza procesului de învățare stau o serie de forme inferentiale nevalide: inductia, abductia si analogia
- O metodă de învățare poate folosi una sau mai multe astfel de forme de inferență, cat si forme de inferență valide, cum este deductia



Inferenta inductiva

- O proprietate adevărată pentru o submultime de obiecte dintr-o clasă este adevărată pentru toate obiectele din acea clasă

$$\frac{P(a_1), P(a_2), \dots, P(a_n)}{(\forall x)P(x)}$$



Inferenta inductiva

- Se poate generaliza la sintetizarea unei întregi reguli de deductie pe baza exemplelor

$$P(a_1) \rightarrow Q(b_1)$$

$$P(a_2) \rightarrow Q(b_2)$$

$$\vdots$$

$$P(a_n) \rightarrow Q(b_n)$$

$$\frac{P(a_n) \rightarrow Q(b_n)}{(\forall x)(\forall y) (P(x) \rightarrow Q(y))}$$



Inferenta abductiva

- Se utilizeaza cunostinte cauzale pentru a explica sau a justifica o concluzie, posibil invalidă

$$\frac{Q(a) \quad (\forall x)(P(x) \rightarrow Q(x))}{P(a)}$$



Inferenta abductiva

- Exemplul 1
 - **Udã(iarba)**
 - **$(\forall \mathbf{x}) (\text{PlouaPeste}(\mathbf{x}) \rightarrow \text{Udã}(\mathbf{x}))$**
 - Se poate infera abductiv că a plouat
 - Cu toate acestea, abductia nu poate fi aplicată consistent în oricare caz

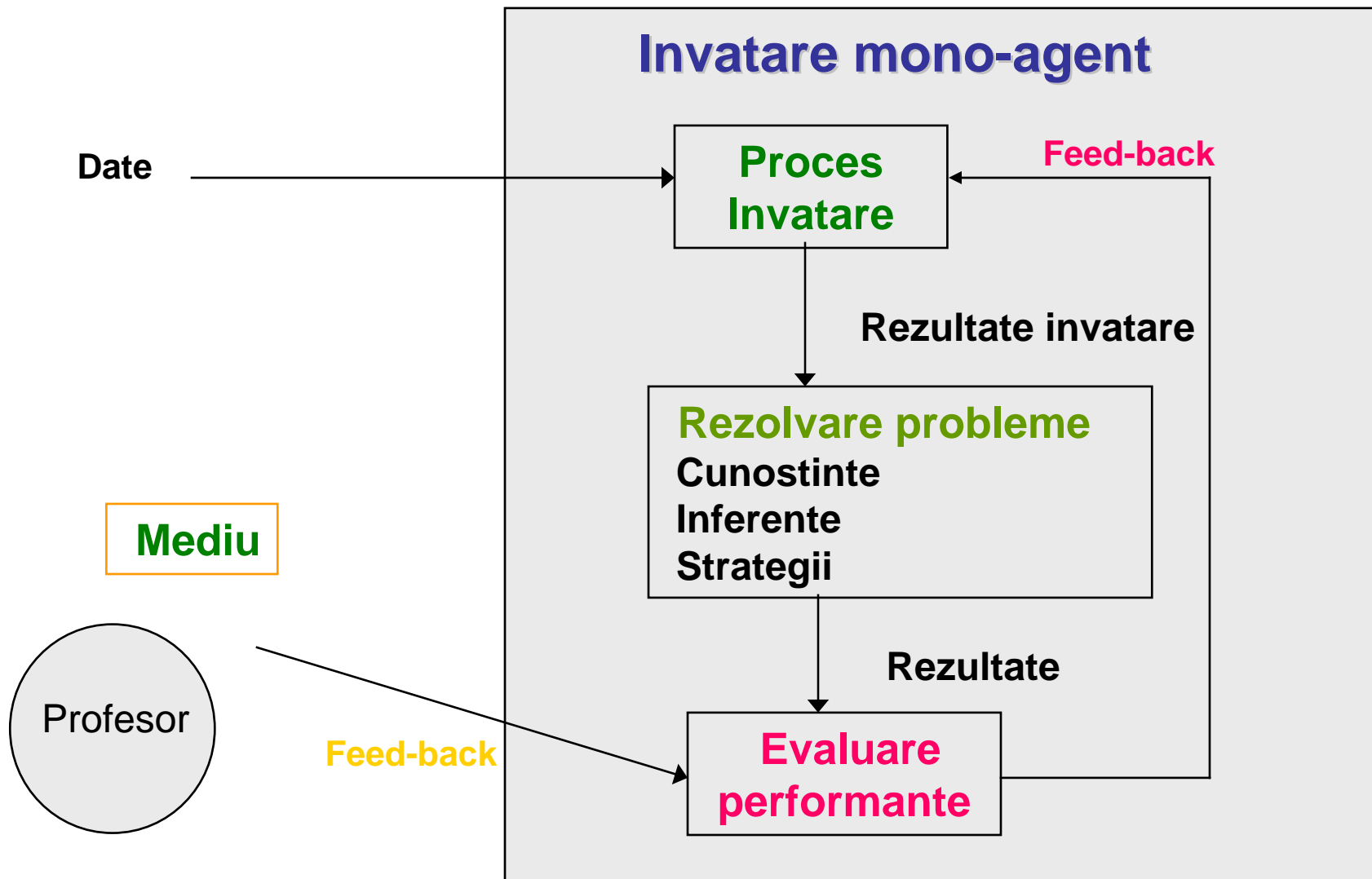


Inferenta analogica

- Situatii sau entități care tind să fie asemănătoare sub anumite aspecte sunt asemănătoare în general
- Este o combinatie a celorlalte forme de inferență: abductive, deductive si inductive

$$\frac{P(x) \xrightarrow{r} Q(x)}{P'(x) \xrightarrow{r} Q'(x)}$$

Modelul conceptual al unui sistem de invatare automata





Modelul conceptual al unui sistem de învățare automată

- În funcție de diferența între nivelul informației oferite de mediu și cel al informației din baza de cunoștințe, se pot identifica patru tipuri de învățare
 - **învățarea prin memorare**
 - **învățarea prin instruire**
 - **învățarea prin inducție (din exemple)**
 - **învățarea prin analogie**



2. Arbori de decizie. Algoritmul ID3

- Invatare inductiva
- Algoritmul ID3 **învată inductiv** concepte din exemple
- Conceptele se reprezintă ca un **arbore de decizie**, ceea ce permite clasificarea unui obiect prin teste asupra valorii anumitor proprietăți (attribute) ale sale
- **Arbore de decizie** - arbore care contine în noduri câte un test pentru o anumită proprietate, fiecare arc fiind etichetat cu o valoare a proprietății testate în nodul din care pleacă arcul respectiv, iar în fiecare frunză o clasă



Prezentarea algoritmului ID3

- Algoritmul ID3 urmează principiul conform căruia explicația cea mai simplă (arborele de decizie cel mai simplu) este și cea adevărată – Ockham's razor
- Ordinea testelor este importantă, punându-se accent pe criteriul alegerii testului din rădăcina arborelui de decizie



Construirea arborelui de decizie

- Mai intai, se construiește arborele de decizie
- După aceea, se folosește arborele de decizie pentru a clasifica exemple necunoscute
- Exemplele necunoscute pot fi clasificate astfel:
 - aparțin unei clase (YES sau C_i)
 - nu aparțin unei clase (NO)



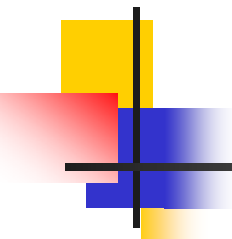
Exemplu simplu de clasificare

No.	Forma	Culoare	Dim	Clasa
1	cerc	rosu	mic	+
2	cerc	rosu	mare	+
3	triunghi	galben	mic	-
4	cerc	galben	mic	-
5	triunghi	rosu	mare	-
6	cerc	galben	mare	-

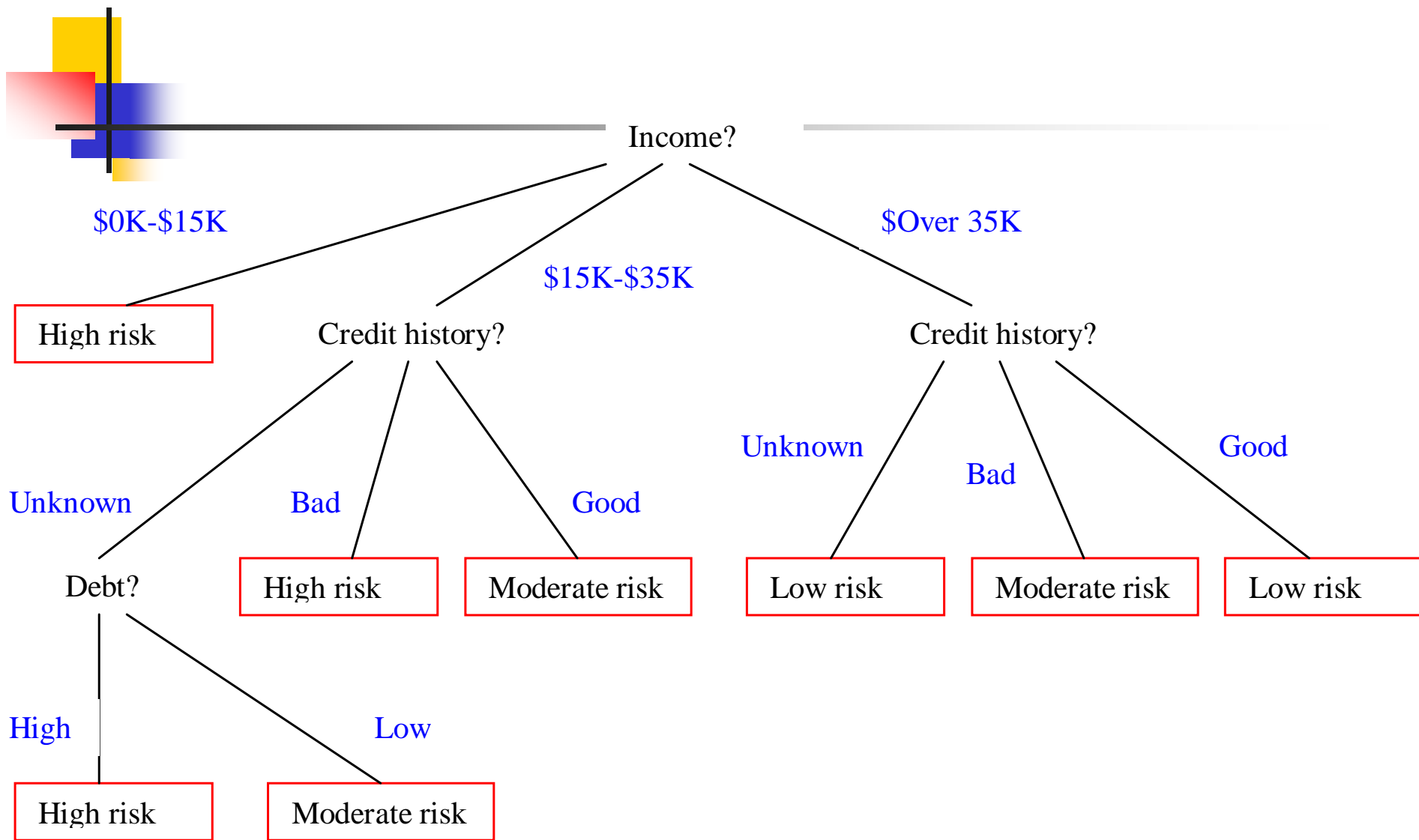


Problema acordarii unui credit

- Problema estimării riscului acordării unui credit unei anumite persoane, bazat pe anumite proprietăți: comportamentul anterior al persoanei atunci când i-au fost acordate credite (istoria creditului), datoria curentă, garanții și venit



<i>No.</i>	<i>Risk (Classification)</i>	<i>Credit History</i>	<i>Debt</i>	<i>Collateral</i>	<i>Income</i>
1	High	Bad	High	None	\$0 to \$15k
2	High	Unknown	High	None	\$15 to \$35k
3	Moderate	Unknown	Low	None	\$15 to \$35k
4	High	Unknown	Low	None	\$0k to \$15k
5	Low	Unknown	Low	None	Over \$35k
6	Low	Unknown	Low	Adequate	Over \$35k
7	High	Bad	Low	None	\$0 to \$15k
8	Moderate	Bad	Low	Adequate	Over \$35k
9	Low	Good	Low	None	Over \$35k
10	Low	Good	High	Adequate	Over \$35k
11	High	Good	High	None	\$0 to \$15k
12	Moderate	Good	High	None	\$15 to \$35k
13	Low	Good	High	None	Over \$35k
14	High	Bad	High	None	\$15 to \$35k



Algoritm pentru construirea arborelui de decizie

functie **ind-arbore** (set-exemple, attribute, default)

1. **daca** set-exemple = vid **atunci intoarce** frunza etichetata cu default
2. **dacă** toate exemplele din set-exemple sunt în aceeași clasă **atunci întoarce** o frunză etichetată cu acea clasă
3. **dacă** attribute este vidă **atunci întoarce** o frunză etichetată cu disjunctia tuturor claselor din set-exemple

- 4.
- selectează un atribut A, creaza nod pt A si eticheteaza nodul cu A
 - sterge A din attribute \rightarrow attribute1
 - m = majoritate (set-exemple)
 - **pentru** fiecare valoare V a lui A **repeta**
 - fie partitie_V multimea exemplurilor din set-exemple, cu valoarea V pentru A
 - creaza $\text{nod}_V = \text{ind-arbore}(\text{partitie}_V, \text{attribute1}, m)$
 - creează legatura nod A - nod_V etichetată cu V

sfarsit



Observatii

- Se pot construi mai multi arbori de decizie, ponind de la multimea data de exemple
- Adancimea arborelui de decizie necesar pentru a clasifica o multime de exemple variaza in functie de ordinea in care attributele sunt testate
- Pentru problema acordarii unui credit, se obtine arborele de decizie cu adancimea cea mai mica in cazul cand in radacina se testeaza atributul “income”
- Algoritmul ID3 alege cel mai simplu arbore de decizie care acopera toate exemplele din multimea initiala



Selectarea atributelor pentru construirea arborelui de decizie

- Consideram fiecare atribut al unui exemplu ca având o anumită contribuție de informație la clasificarea respectivului exemplu
- Euristica algoritmului ID3 măsoară câștigul informațional pe care îl aduce fiecare atribut și alege ca test acel atribut care maximizează acest câștig



Notiuni despre teoria informatiei

- Teoria informatiei furnizează fundamentul matematic pentru măsurarea conținutului de informație dintr-un mesaj
- Un mesaj este privit ca o instanță dintr-un univers al tuturor mesajelor posibile
- Transmiterea mesajului este echivalentă cu selecția unui anumit mesaj din acest univers



Notiuni despre teoria informatiei

- Continutul informational al unui mesaj depinde de mărimea universului si de frecventa fiecărui mesaj
- Continutul informational al unui mesaj se defineste ca fiind probabilitatea de aparitie a oricărui mesaj posibil



Notiuni despre teoria informatiei

- Având un univers de mesaje
- $M = \{m_1, m_2, \dots, m_n\}$
- si o probabilitate $p(m_i)$ de aparitie a fiecărui mesaj, continutul informational al unui mesaj din M se defineste astfel:

$$I(M) = \sum_{i=1}^n -p(m_i) \log_2(p(m_i))$$



Notiuni despre teoria informatiei

- Informatia dintr-un mesaj se măsoară in biti
- Algoritmul ID3 foloseste teoria informatiei pentru a selecta atributul care ofera cel mai mare câstig informational în clasificarea exemplilor de învățare
- Consideram un arbore de decizie ca având informatie despre clasificarea exemplilor din multimea de învățare
- Continutul informational al arborelui este calculat cu ajutorul probabilităților diferitelor clasificări



Continutul de informatie I(T)

- $p(\text{risk is high}) = 6/14$
- $p(\text{risk is moderate}) = 3/14$
- $p(\text{risk is low}) = 5/14$
- Continutul de informatie al arborelui de decizie este:

$$I(Arb) = \sum_{i=1}^n - p(Cl = C_i) * \log_2 p(Cl = C_i)$$

- $I(Arb) = 6/14 \log(6/14) + 3/14 \log(3/14) + 5/14 \log(5/14)$



Castigul informational $G(A)$

- Pentru un anumit atribut A , câstigul informational produs de selectarea acestuia ca rădăcină a arborelui de decizie este egal cu continutul total de informatie din arbore minus continutul de informatie necesar pentru a termina clasificarea (construirea arborelui), dupa selectarea atributului A ca radacina
- $G(A) = I(Arb) - E(A)$



Cum calculam $E(A)$

- Cantitatea de informație necesară pentru a termina construcția arborelui este media ponderată a conținutului de informație din toți subarborii
- Presupunem că avem o mulțime de exemple de învățare C
- Dacă punem atributul A cu n valori în rădăcina arborelui de decizie, acesta va determina partitionarea mulțimii C în submulțimile $\{C_1, C_2, \dots, C_n\}$



Cum calculam $E(A)$

- Estimarea cantității de informație necesară pentru a construi arborele de decizie, după ce atributul A a fost ales ca rădăcină, este:

$$E(A) = \sum_{i=1}^n \frac{|C_i|}{|C|} I(C_i)$$



Problema acordarii unui credit

- Daca atributul “Income” este ales ca radacina a arborelui de decizie, aceasta determina impartirea multimii de exemple in submultimile:

- $C_1 = \{1, 4, 7, 11\}$

- $C_2 = \{2, 3, 12, 14\}$

- $C_3 = \{5, 6, 8, 9, 10, 13\}$

$$G(\text{income}) = I(\text{Arb}) - E(\text{Income}) = 1,531 - 0,564 = 0,967 \text{ bits}$$

$$G(\text{credit history}) = 0,266 \text{ bits}$$

$$G(\text{debt}) = 0,581 \text{ bits}$$

$$G(\text{collateral}) = 0,756 \text{ bits}$$



Performanta invatarii

- Fie S mult de ex
- Imparte S in set de invatare si set de test
- Aplica ID3 la set de invatare
- Masoare proc ex clasificate corect din set de test
- Repeta pasii de mai sus pt diferite dimensiuni ale set invatare si set test, alese aleator
- Rezulta o predictie a performantei invatarii
- Grafic X- dim set invatare, Y- procent set test
- Happy graphs



Observatii

- Date lipsa
- Attribute cu valori multiple si castig mare
- Attribute cu valori intregi si continue
- Reguli de decizie



3. Invatarea conceptelor din exemple prin clusterizare

- Generalizare si specializare

Exemple de invatare

1. (galben piram lucios mare +)
2. (bleu sfera lucios mic +)
3. (galben piram mat mic +)
4. (verde sfera mat mare +)
5. (galben cub lucios mare +)
6. (bleu cub lucios mic -)
7. (bleu piram lucios mare -)



Invatarea conceptelor prin clusterizare

nume concept: NUME

parte pozitiva

cluster: descriere: (galben piram lucios mare)

ex: 1

parte negativa

ex:

nume concept: NUME

parte pozitiva

cluster: descriere: (_ _ lucios _)

ex: 1, 2

parte negativa

ex:

1. (galben piram lucios mare +)
2. (bleu sfera lucios mic +)
3. (galben piram mat mic +)
4. (verde sfera mat mare +)
5. (galben cub lucios mare +)
6. (bleu cub lucios mic -)
7. (bleu piram lucios mare -)



Invatarea conceptelor prin clusterizare

nume concept: NUME

parte pozitiva

cluster: descriere: (_ _ _ _)

ex: 1, 2, 3, 4, 5

parte negativa

ex: 6, 7

suprageneralizare

1. (galben piram lucios mare +)
2. (bleu sfera lucios mic +)
3. (galben piram mat mic +)
4. (verde sfera mat mare +)
5. (galben cub lucios mare +)
6. (bleu cub lucios mic -)
7. (bleu piram lucios mare -)



Invatarea conceptelor prin clusterizare

nume concept: NUME

parte pozitiva

cluster: descriere: (galben piram lucios mare)

ex: 1

cluster: descriere: (bleu sfera lucios mic)

ex: 2

parte negativa

ex: 6, 7

1. (galben piram lucios mare +)
2. (bleu sfera lucios mic +)
3. (galben piram mat mic +)
4. (verde sfera mat mare +)
5. (galben cub lucios mare +)
6. (bleu cub lucios mic -)
7. (bleu piram lucios mare -)



Invatarea conceptelor prin clusterizare

nume concept: NUME

parte pozitiva

cluster: descriere: (galben piram _ _)

ex: 1, 3

cluster: descriere: (_ sfera _ _)

ex: 2, 4

parte negativa

ex: 6, 7

1. (galben piram lucios mare +)
2. (bleu sfera lucios mic +)
3. (galben piram mat mic +)
4. (verde sfera mat mare +)
5. (galben cub lucios mare +)
6. (bleu cub lucios mic -)
7. (bleu piram lucios mare -)



Invatarea conceptelor prin clusterizare

nume concept: NUME

parte pozitiva

cluster: descriere: (galben _ _ _)

ex: 1, 3, 5

cluster: descriere: (_ sfera _ _)

ex: 2, 4

parte negativa

ex: 6, 7

A daca *galben* sau *sfera*

1. (galben piram lucios mare +)
2. (bleu sfera lucios mic +)
3. (galben piram mat mic +)
4. (verde sfera mat mare +)
5. (galben cub lucios mare +)
6. (bleu cub lucios mic -)
7. (bleu piram lucios mare -)

Invatare prin clusterizare

1. Fie S setul de exemple
2. Creaza PP si PN
3. Adauga toate ex- din S la PN (*vezi coment) si elimina ex- din S
4. Creaza un cluster in PP si adauga primul ex+
5. $S = S - \text{ex+}$
6. **pentru** fiecare ex+ din S e_i **repeta**
 - 6.1 **pentru** fiecare cluster C_i **repeta**
 - Creaza descriere $e_i + C_i$
 - **daca** descriere nu acopera nici un ex-
atunci adauga e_i la C_i
 - 6.2 **daca** e_i nu a fost adaugat la nici un cluster
atunci creaza un nou cluster cu e_i

sfarsit



4. Invatarea prin cautare in spatiul versiunilor

Operatori de generalizare in spatiul versiunilor

- Inlocuirea const cu var

color(ball, red) color(X, red)

- Eliminarea unor literali din conjunctii

shape(X, round) \wedge size(X, small) \wedge color(X, red)

shape(X, round) \wedge color(X, red)

- Adaugarea unei disjunctii

shape(X, round) \wedge size(X, small) \wedge color(X, red)

shape(X, round) \wedge size(X, small) \wedge (color(X, red) \vee color(X, blue))

- Inlocuirea unei proprietati cu parintele din ierarhie

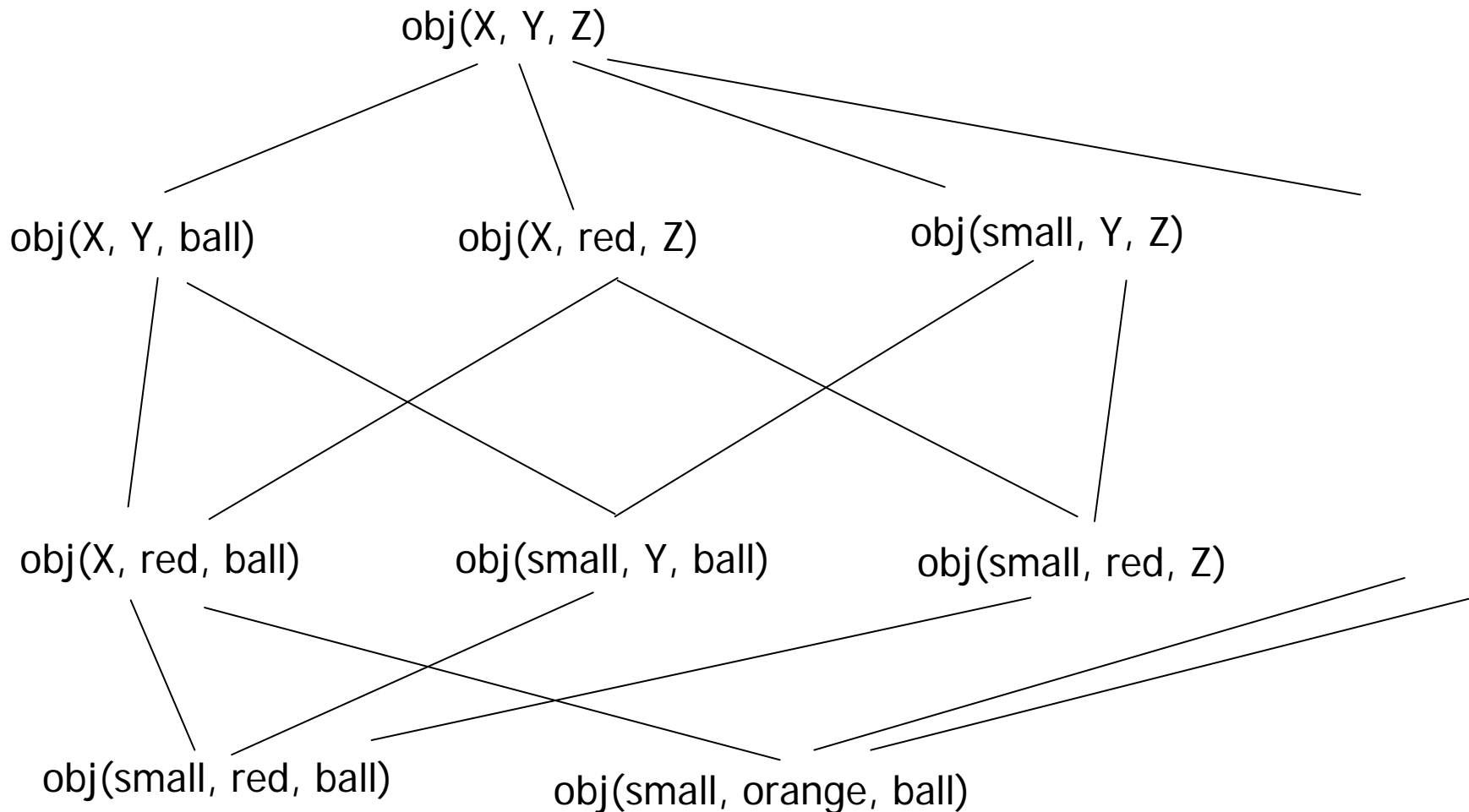
is-a(tom, cat) is-a(tom, animal)



Algoritmul de eliminare a candidatilor

- **Spatiul versiunilor** = multimea de descriere a conceptelor consistente cu exemplele de invatare
- **Idee** = reducerea spatiului versiunilor pe baza ex inv
- 1 algoritm – de la specific la general
- 1 algoritm – de la general la specific
- 1 algoritm – cautare bidirectionala = algoritmul de eliminare a candidatilor

Algoritmul de eliminare a candidatilor - cont





Generalizare si specializare

- P si Q – multimile care identifica cu p, q in FOPL
- Expresia **p este mai generala decat q** **daca si numai daca**

$$P \supseteq Q$$

$$\text{color}(X, \text{red}) \supseteq \text{color}(\text{ball}, \text{red})$$

- p mai general decat q - **$p \geq q$**

$$\forall x \, p(x) \rightarrow \text{pozitiv}(x)$$

$$\forall x \, q(x) \rightarrow \text{pozitiv}(x)$$

- p acopera q **daca si numai daca:**

$q(x) \rightarrow \text{pozitiv}(x)$ este o consecinat logica a $p(x) \rightarrow \text{pozitiv}(x)$

- **Spatiul conceptelor** $\text{obj}(X, Y, Z)$



Generalizare si specializare

- Un concept **c** **este maxim specific** daca acopera toate exemplele pozitive, nu acopera nici un exemplu negativ si pentru $\forall c'$ care acopera exemplele pozitive, $c \leq c'$. - **S**
- Un concept **c** **este maxim general** daca nu acopera nici un exemplu negativ si pentru $\forall c'$ care nu acopera nici un exemplu negativ, $c \geq c'$. - **G**

S – multime de ipoteze (concepte candidate) = **generalizarile specifice maxime**

G – multime de ipoteze (concepte candidate) = **specializarile generale maxime**



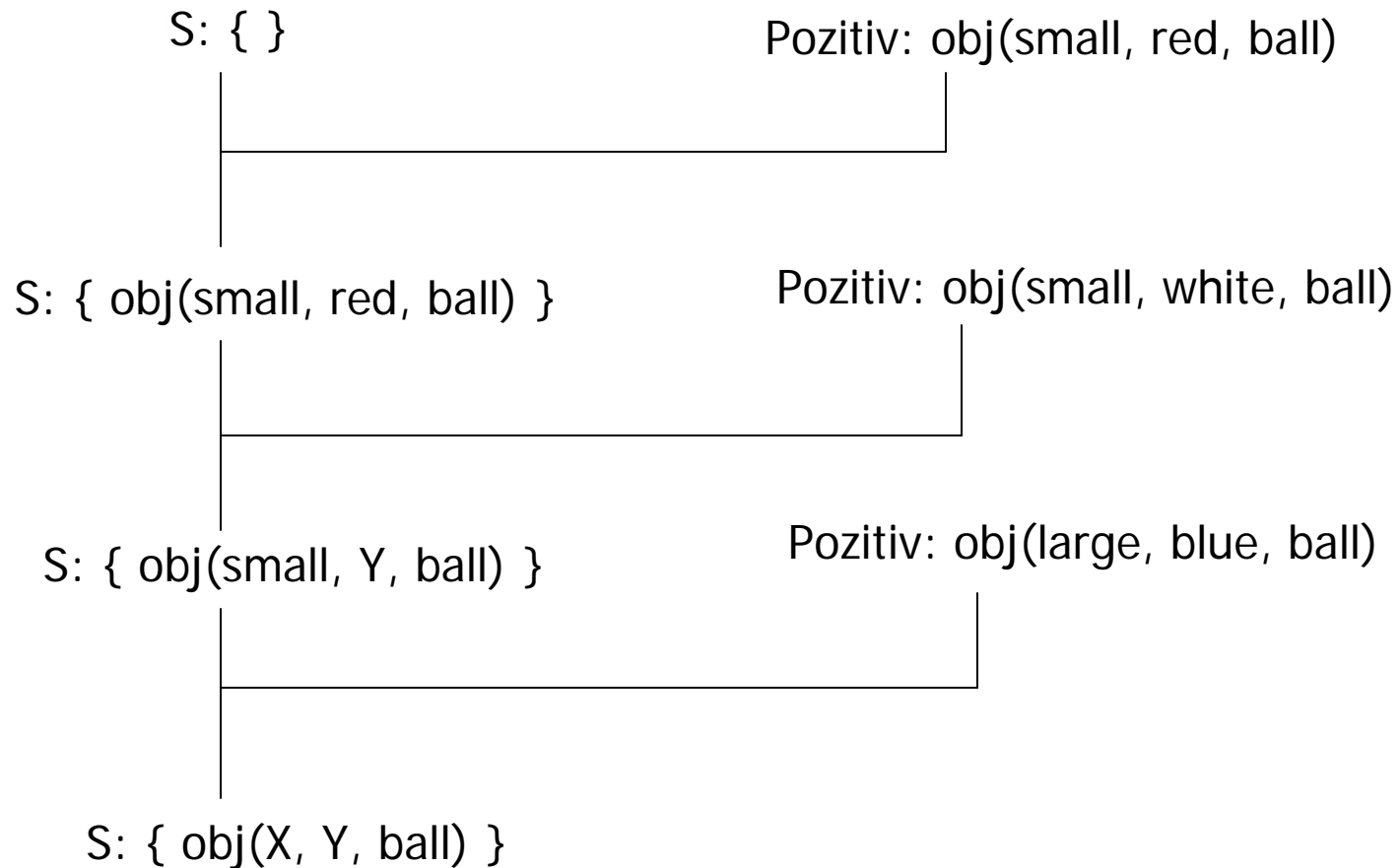
Algoritmul de cautare de la specific la general

1. Initializeaza **S** cu primul exemplu pozitiv
2. Initializeaza **N** la multimea vida
3. **pentru** fiecare exemplu de invatare **repeta**
 - 3.1 **daca** ex inv este exemplu pozitiv, ***p***, **atunci**
pentru fiecare $s \in S$ **repeta**
 - **daca** s nu acopera ***p*** **atunci** inlocuieste s cu cea mai specifica generalizare care acopera ***p***
 - Elimina din **S** toate ipotezele mai generale decat alte ipoteze din **S**
 - Elimina din **S** toate ipotezele care acopera un exemplu negativ din **N**
 - 3.2 **daca** ex inv este exemplu negativ, ***n***, **atunci**
 - Elimina din **S** toate ipotezele care acopera ***n***
 - Adauga ***n*** la **N** (pentru a verifica suprageneralizarea)

sfarsit



Algoritmul de cautare de la specific la general





Algoritmul de cautare de la general la specific

1. Initializeaza **G** cu cea mai generala descriere
2. Initializeaza **P** la multimea vida
3. **pentru** fiecare exemplu de invatare **repete**
 - 3.1 **daca** ex inv este exemplu negativ, **n**, **atunci**
pentru fiecare $g \in \mathbf{G}$ **repete**
 - **daca** g acopera **n** **atunci** inlocuieste g cu cea mai generala specializare care nu acopera **n**
 - Elimina din **G** toate ipotezele mai specifice decat alte ipoteze din **G**
 - Elimina din **G** toate ipotezele care nu acopera exemple pozitive din **P**
 - 3.2 **daca** ex inv este exemplu pozitiv, **p**, **atunci**
 - Elimina din **G** toate ipotezele care nu acopera **p**
 - Adauga **p** la **P** (pentru a verifica supraspecializarea)

sfarsit



Algoritmul de cautare de la general la specific

G: { obj(X, Y, Z) }

Negativ: obj(small, red, brick)

G: { obj(large, Y, Z), obj(X, white, Z),
obj(X, blue, Z), obj(X, Y, ball), obj(X, Y, cube) }

Pozitiv: obj(large, white, ball)

G: { obj(large, Y, Z), obj(X, white, Z),
obj(X, Y, ball) }

Negativ: obj(large, blue, cube)

G: { obj(X, white, Z),
obj(X, Y, ball) }

Pozitiv: obj(small, blue, ball)

G: obj(X, Y, ball)



Algoritmul de cautare in spatiul versiunilor

1. Initializeaza **G** cu cea mai generala descriere
2. Initializeaza **S** cu primul exemplu pozitiv
3. **pentru** fiecare exemplu de invatare **repeta**
 - 3.1 **daca** ex inv este exemplu pozitiv, ***p***, **atunci**
 - 3.1.1 Elimina din **G** toate elementele care nu acopera ***p***
 - 3.1.2 **pentru** fiecare $s \in \mathbf{S}$ **repeta**
 - **daca** s nu acopera ***p*** **atunci** inlocuieste s cu cea mai specifica generalizare care acopera ***p***
 - Elimina din **S** toate ipotezele mai generale decat alte ipoteze din **S**
 - Elimina din **S** toate ipotezele mai generale decat alte ipoteze din **G**



Algoritmul de cautare in spatiul versiunilor - cont

3.2 **daca** ex inv este exemplu negativ, n , **atunci**

3.2.1 Elimina din S toate ipotezele care acopera n

3.2.2 **pentru** fiecare $g \in G$ **repeta**

- **daca** g acopera n **atunci** inlocuieste g cu cea mai generala specializare care nu acopera n

- Elimina din G toate ipotezele mai specifice decat alte ipoteze din G

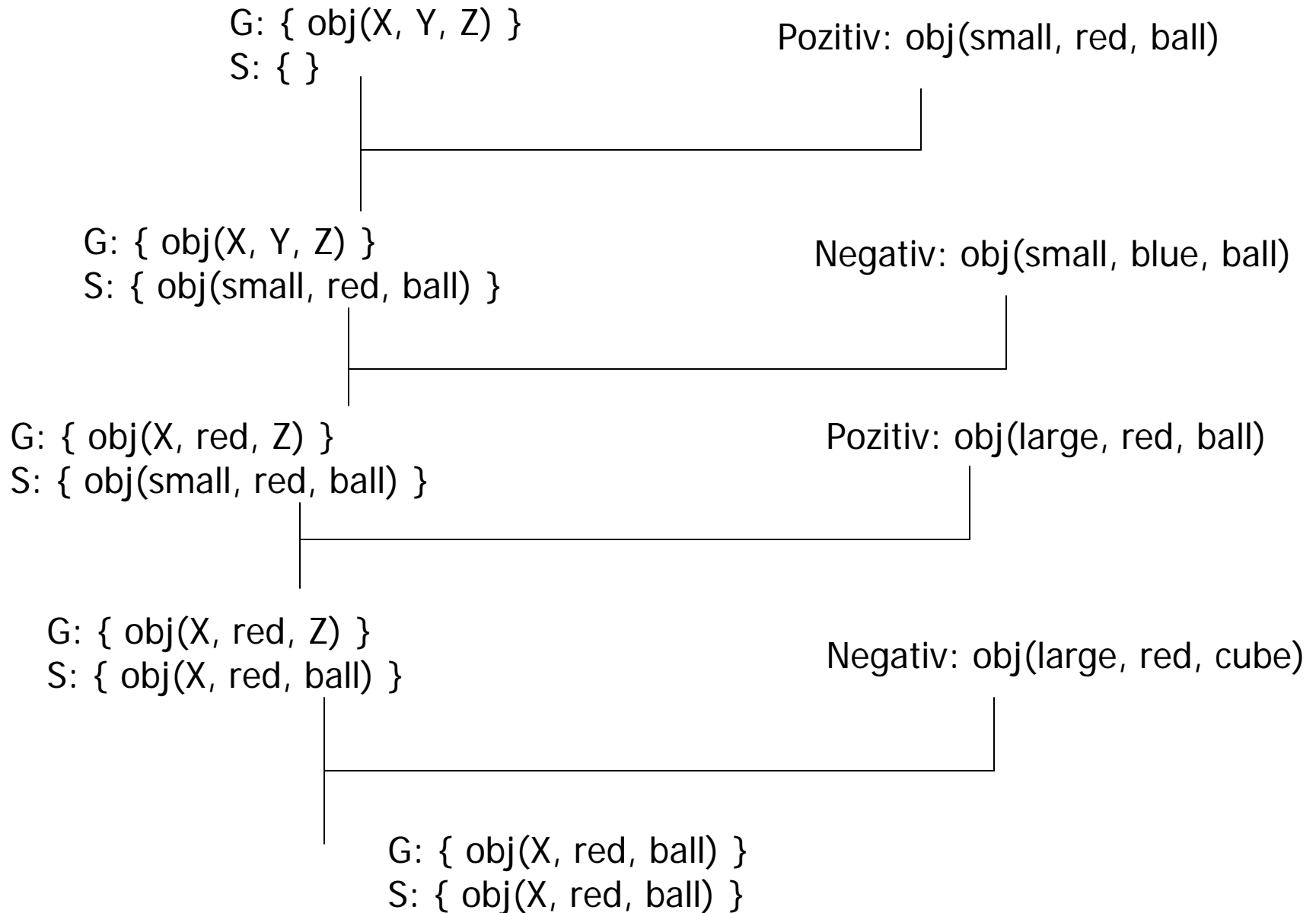
- Elimina din G toate ipotezele mai specifice decat alte ipoteze din S

4. daca $G = S$ si $\text{card}(S) = 1$ **atunci** s-a gasit un concept

5. daca $G = S = \{ \}$ **atunci** nu exista un concept consistent cu toate exemplele

sfarsit

Algoritmul de cautare in spatiul versiunilor



Implementare algoritmi specifici-general

```
exemple([pos([large,white,ball]),neg([small,red,brick]),  
        pos([small,blue,ball]),neg([large,blue,cube]))).
```

```
acopera([],[]).
```

```
acopera([H1|T1], [H2|T2]) :- var(H1), var(H2), acopera(T1,T2).
```

```
acopera([H1|T1], [H2|T2]) :- var(H1), atom(H2), acopera(T1,T2).
```

```
acopera([H1|T1], [H2|T2]) :- atom(H1), atom(H2), H1=H2,  
acopera(T1,T2).
```

```
maigeneral(X,Y) :- not(acopera(Y,X)), acopera(X,Y).
```

```
generaliz([], [], []).
```

```
generaliz([Atrib|Rest], [Inst|RestInst], [Atrib|RestGen]) :-
```

```
    Atrib==Inst, generaliz(Rest,RestInst,RestGen).
```

```
generaliz([Atrib|Rest], [Inst|RestInst], [_|RestGen]) :-
```

```
    Atrib\=Inst, generaliz(Rest,RestInst,RestGen).
```

Implementare algoritmi specific-general

```
specgen :- exemple( [pos(H)|Rest] ), speclagen([H], [], Rest).
```

```
speclagen(H, N, []) :- print('H='), print(H), nl,  
                        print('N='), print(N), nl.
```

```
speclagen(H, N, [Ex|RestEx]) :- process(Ex, H, N, H1, N1),  
                                speclagen(H1, N1, RestEx).
```

```
process(pos(Ex), H, N, H1, N) :-  
    generalizet(H, HGen, Ex),  
    elim(X, HGen, (member(Y,HGen), maigeneral(X,Y)), H2),  
    elim(X, H2, (member(Y,N),acopera(X,Y)), H1).
```

```
process(neg(Ex), H, N, H1, [Ex|N]) :-  
    elim(X, H, acopera(X,Ex), H1).
```

```
elim(X,L,Goal,L1):- (bagof(X, (member(X,L), not(Goal)), L1);  
                    L1=[]).
```

Implementare algoritm specific-general

```
generalizset([], [], _).
```

```
generalizset([Ipot|Rest], IpotNoua, Ex) :-  
    not(acopera(Ipot,Ex)),  
    (bagof(X, generaliz(Ipot,Ex,X), ListIpot);  
     ListIpot=[]),  
    generalizset(Rest,RestNou,Ex),  
    append(ListIpot,RestNou,IpotNoua).
```

```
generalizset([Ipot|Rest], [Ipot|RestNou], Ex):-  
    acopera(Ipot,Ex),  
    generalizset(Rest,RestNou,Ex).
```

?- specgen.

```
H=[[_G390, _G393, ball]]
```

```
N=[[large, blue, cube], [small, red, brick]]
```