Inteligenta Artificiala

Universitatea Politehnica Bucuresti
Anul universitar 2010-2011

Adina Magda Florea

http://turing.cs.pub.ro/ia_10 si curs.cs.pub.ro

Curs nr. 11

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 învãta
- Învatarea automatã este domeniul cel mai provocator al inteligentei artificiale si, în acelasi timp, cel mai rezistent încercãrilor de automatizare completã

Reguli de inferenta utilizate in invatare

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

Inferenta inductiva

- O proprietate adevãratã pentru o submultime de obiecte dintr-o clasã este adevãratã pentru toate obiectele din acea clasã

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

Inferenta inductiva

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

$$\frac{\begin{array}{c} P(a_1) \rightarrow Q(b_1) \\ P(a_2) \rightarrow Q(b_2) \\ \vdots \\ P(a_n) \rightarrow Q(b_n) \end{array}}{(\forall x)(\forall y)\,(P(x) \rightarrow Q(y))}$$

Inferenta abductiva

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

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

Inferenta abductiva

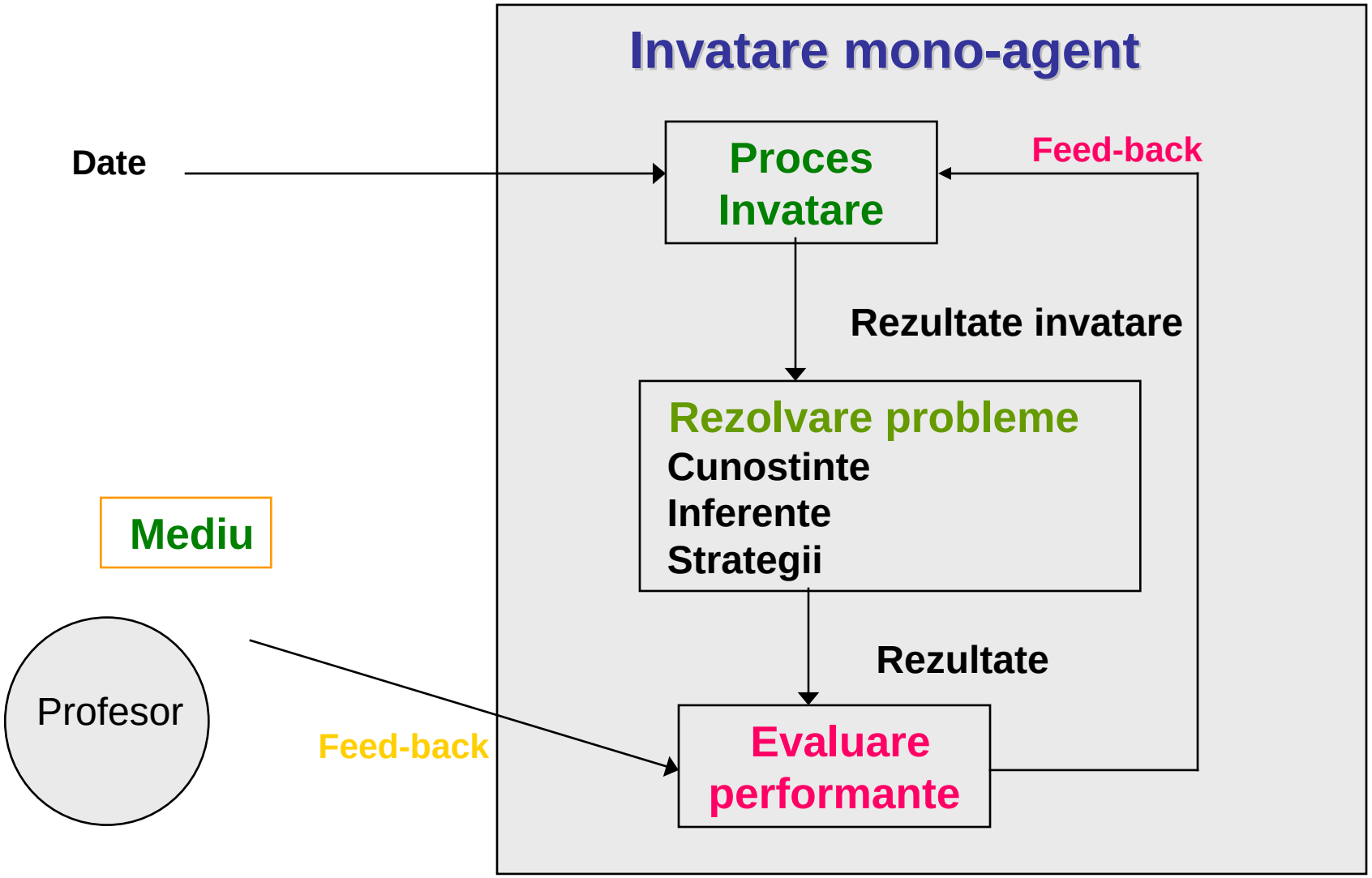
- Exemplul 1
  - **Udã(iarba)**
  - **$(\forall x)$ (PlouaPeste(x) $\rightarrow$ Udã(x))**
  - Se poate infera abductiv cã a plouat
  - Cu toate acestea, abductia nu poate fi aplicatã consistent în oricare caz

Inferenta analogica

- Situatii sau entitãti care tind sã fie asemãnãtoare sub anumite aspecte sunt asemãnãtoare în general
- Este o combinatie a celorlalte forme de inferentã: 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 invatare automata

- În functie de diferenta între nivelul informatiei oferite de mediu si cel al informatiei din baza de cunostinte, se pot identifica patru tipuri de învãtare
  - **invatarea prin memorare**
  - **invatarea prin instruire**
  - **invatarea prin inductie (din exemple)**
  - **invatarea prin analogie**

2. Arbori de decizie. Algoritmul ID3

- Invatare inductiva
- Algoritmul ID3 **învatã inductiv** concepte din exemple
- Conceptele se reprezintã ca un **arbore de decizie**, ceea ce permite clasificarea unui obiect prin teste asupra valorii anumitor proprietãti (atribute) ale sale
- **Arbore de decizie -** arbore care contine în noduri câte un test pentru o anumitã proprietate, fiecare arc fiind etichetat cu o valoare a proprietãtii testate în nodul din care pleacã arcul respectiv, iar în fiecare frunzã o clasã

Prezentarea algoritmului ID3

- Algoritmul ID3 urmeazã principiul conform cãruia explicatia cea mai simplã (arborele de decizie cel mai simplu) este si cea adevãratã – Ockham’s razor
- Ordinea testelor este importantã, punându-se accent pe criteriul alegerii testului din rãdãcina arborelui de decizie

Construirea arborelui de decizie

- Mai intai, se construieste arborele de decizie
- Dupa aceea, se foloseste arborele de decizie pentru a clasifica exemple necunoscute
- Exemplele necunoscute pot fi clasificate astfel:
  - apartin unei clase (YES sau C_i)
  - nu apartin unei clase (NO)

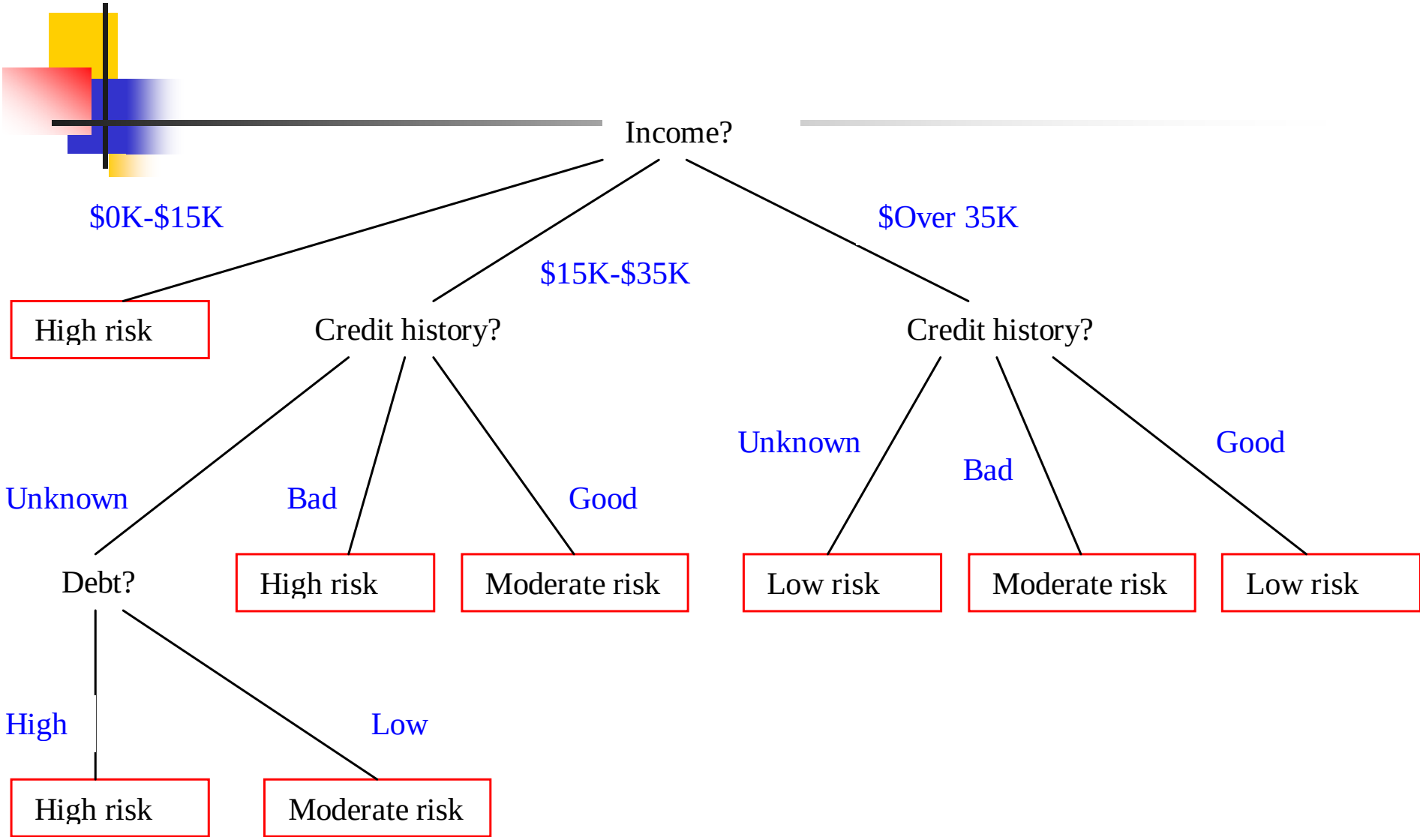
Exemplu simplu de clasificare

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 estimãrii riscului acordãrii unui credit unei anumite persoane, bazat pe anumite proprietãti: comportamentul anterior al persoanei atunci când i-au fost acordate credite (istoria creditului), datoria curentã, garantii si venit

No.	*Risk (Classification)*	*Credit History*	*Debt*	*Collateral*	*Income*
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


Income?
$0K-$15K
$15K-$35K
$Over 35K
High risk
Credit history?
Credit history?
Unknown
Bad
Good
Unknown
Bad
Good
Debt?
High risk
Moderate risk
Low risk
Moderate risk
Low risk
High
Low
High risk
Moderate risk

Algoritm pentru construirea arborelui de decizie

functie ind-arbore (set-exemple, atribute, default)

1. **daca** set-exemple = vid **atunci intoarce** frunza etichetata cu default
2. **dacã** toate exemplele din set-exemple sunt în aceeasi clasã
   atunci întoarce o frunzã etichetatã cu acea clasã
3. **dacã** atribute este vidã
   atunci întoarce o frunzã etichetatã cu disjunctia tuturor claselor din set-exemple

4. - selectează un atribut A, creaza nod pt A si eticheteaza nodul cu A

- sterge A din atribute –> atribute1
- m = majoritate (set-exemple)
- **pentru** fiecare valoare V a lui A **repeta**
  - fie $partitie_V$ multimea exemplelor din set-exemple, cu valorea V pentru A
  - creaza nod_V = ind-arbore ($partitie_V$, atribute1,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 atributele 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 anumitã contributie de informatie la clasificarea respectivului exemplu
- Euristica algoritmului ID3 mãsoarã câstigul informational pe care îl aduce fiecare atribut si alege ca test acel atribut care maximizeazã acest câstig

Notiuni despre teoria informatiei

- Teoria informatiei furnizeazã fundamentul matematic pentru mãsurarea continutului de informatie dintr-un mesaj
- Un mesaj este privit ca o instantã dintr-un univers al tuturor mesajelor posibile
- Transmiterea mesajului este echivalentã cu selectia unui anumit mesaj din acest univers

Notiuni despre teoria informatiei

- Continutul informational al unui mesaj depinde de mãrimea universului si de frecventa fiecãrui mesaj
- Continutul informational al unui mesaj se defineste ca fiind probabilitatea de aparitie a oricãrui mesaj posibil

Notiuni despre teoria informatiei

- Având un univers de mesaje
- $M = \{m_1, m_2, ..., m_n\}$
- si o probabilitate $p(m_i)$ de aparitie a fiecãrui 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 mãsoarã in biti
- Algoritmul ID3 foloseste teoria informatiei pentru a selecta atributul care ofera cel mai mare câstig informational în clasificarea exemplelor de învãtare
- Consideram un arbore de decizie ca având informatie despre clasificarea exemplelor din multimea de învãtare
- Continutul informational al arborelui este calculat cu ajutorul probabilitãtilor diferitelor clasificãri

Continutul de informatie I(T)

- ***p(risk is high) = 6/14***
- ***p(risk is moderate) = 3/14***
- ***p(risk is low) = 5/14***

- Continutul de informatie al arborelui de decizie este:

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

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

Castigul informational G(A)

- Pentru un anumit atribut **A**, câstigul informational produs de selectarea acestuia ca rãdãcinã a arborelui de decizie este egal cu continutul total de infomatie 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 informatie necesarã pentru a termina constructia arborelui este media ponderatã a continutului de informatie din toti subarborii
- Presupunem cã avem o multime de exemple de învãtare **C**
- Dacã punem atributul **A** cu **n** valori în rãdãcina arborelui de decizie, acesta va determina partitionarea multimii **C** în submultimile $\{C_1, C_2, ..., C_n\}$

Cum calculam E(A)

- Estimarea cantitãtii de informatie necesarã pentru a construi arborele de decizie, dupã ce atributul **A** a fost ales ca rãdãcinã, 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(income) = I(Arb) - E(Income) = 1,531 - 0,564 = 0,967 bits

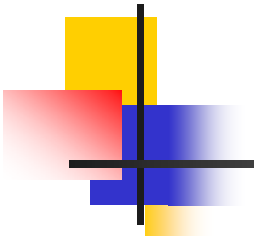
G(credit history) = 0,266 bits

G(debt) = 0,581 bits

G(collateral) = 0,756 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
- Atribute cu valori multiple si castig mare
- Atribute 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

supragenerализare

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 – 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

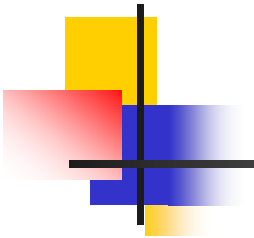
  shape(X, round) $\wedge$ size(X, small) $\wedge$ color(X, red)

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

- Inlocuirea unei proprietati cu parintele din ierarhie

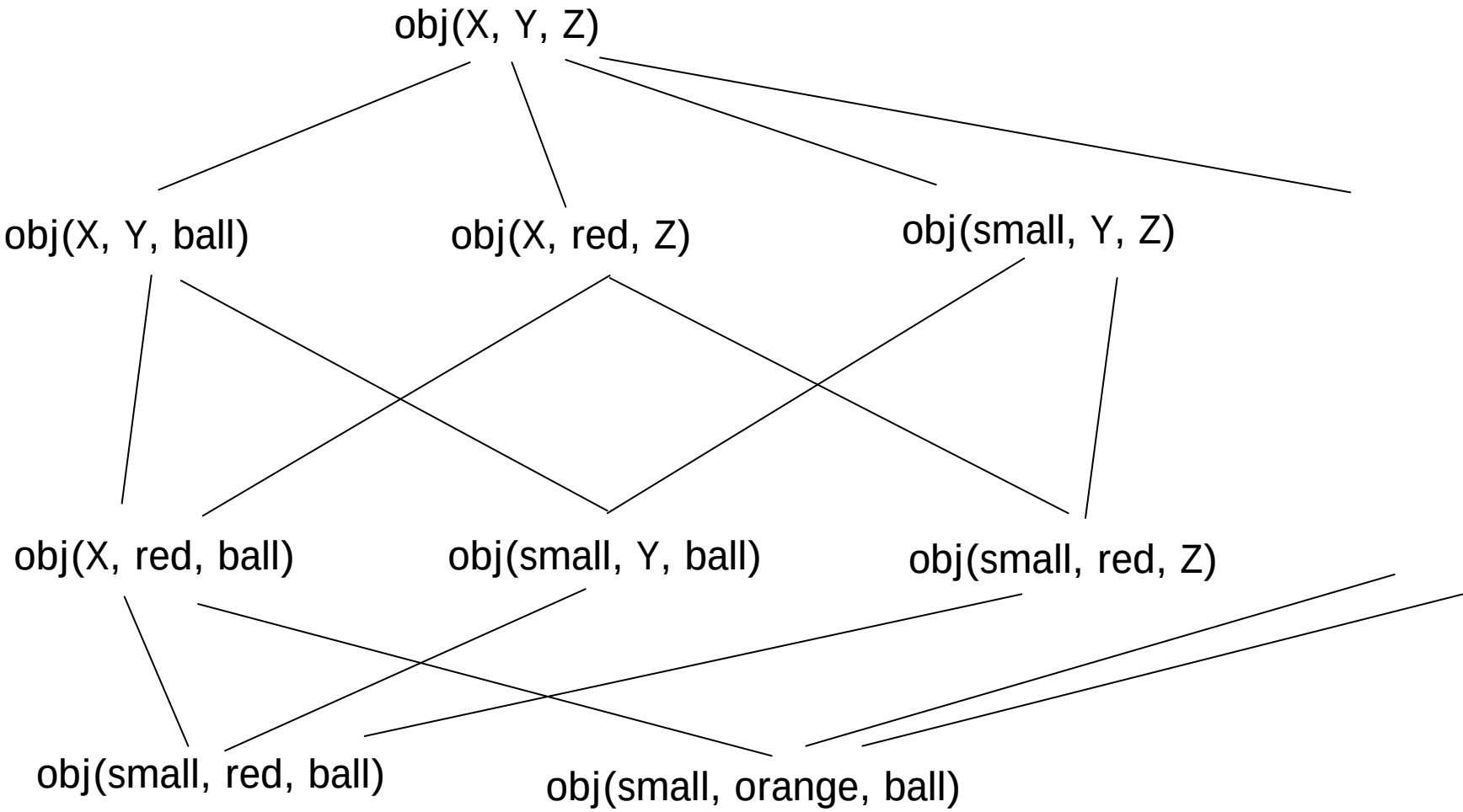
is-a(tom, cat) is-a(tom, animal)

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**

$$P \supseteq Q$$

$$color(X,red) \supseteq color(ball,red)$$

- p mai general decat q **- p ≥ q**

  $\forall x\ p(x) \rightarrow pozitiv(x)$ $\forall x\ q(x) \rightarrow pozitiv(x)$

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

  q(x) → pozitiv(x) este o consecinat logica a p(x) → pozitiv(x)

- **Spatiul conceptelor** 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**

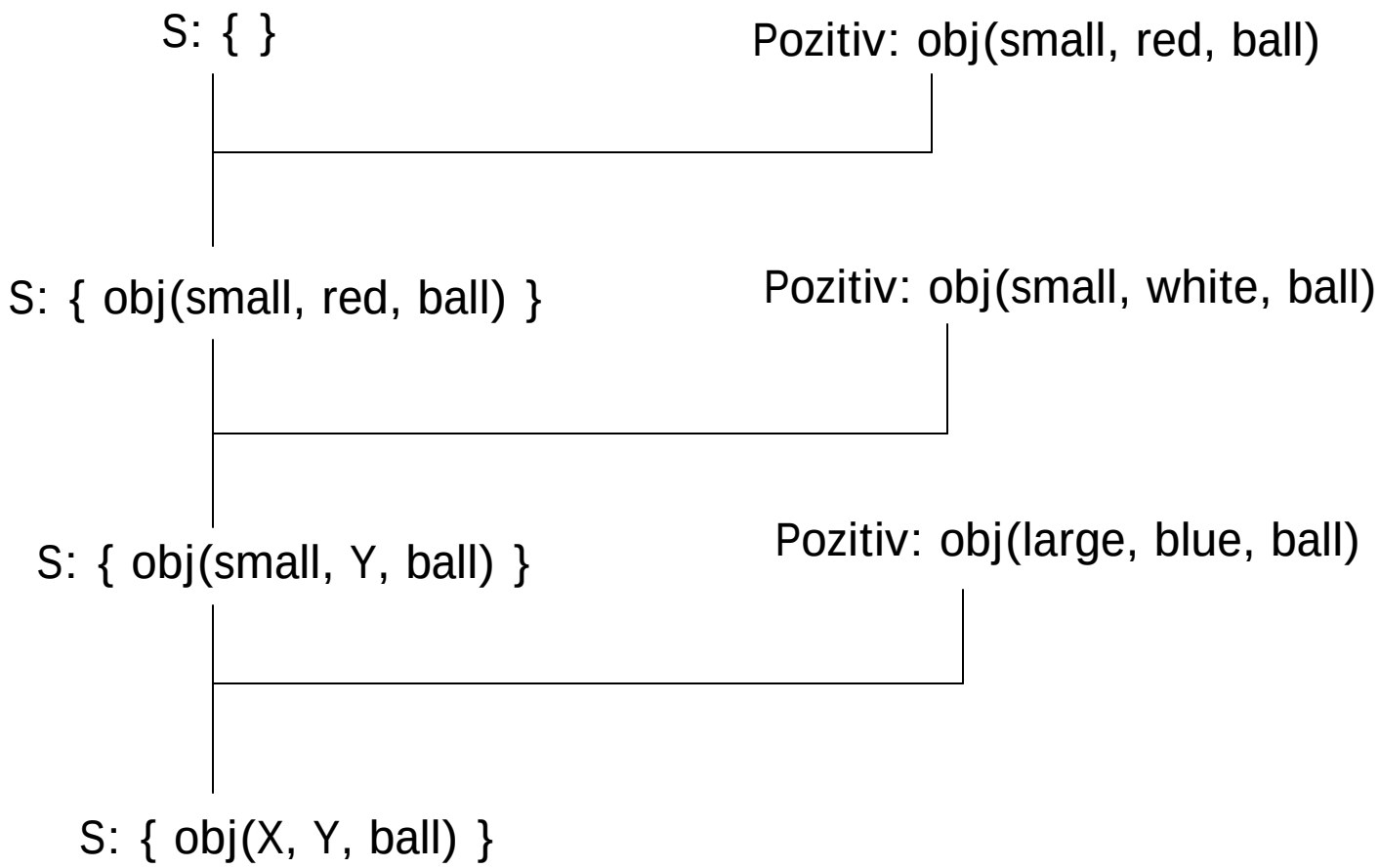
   3.2 **daca** ex inv este exemplu negativ, ***n***, **atunci**

   - 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 **repeta**

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

   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 care nu acopera exemple pozitive din **P**

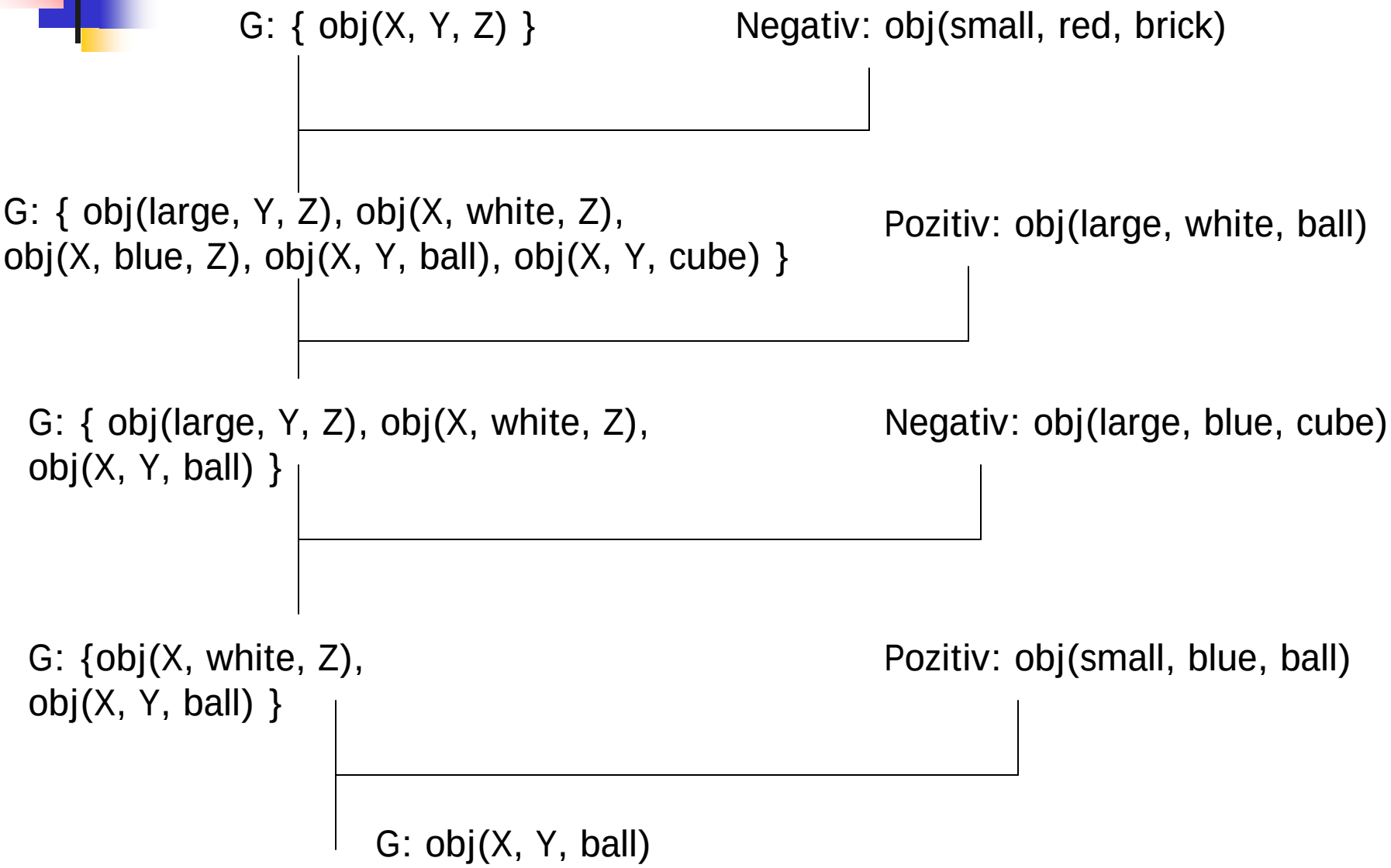
   3.2 **daca** ex inv este exemplu pozitiv, ***p***, **atunci**

   - 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



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 $s \in$ **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**

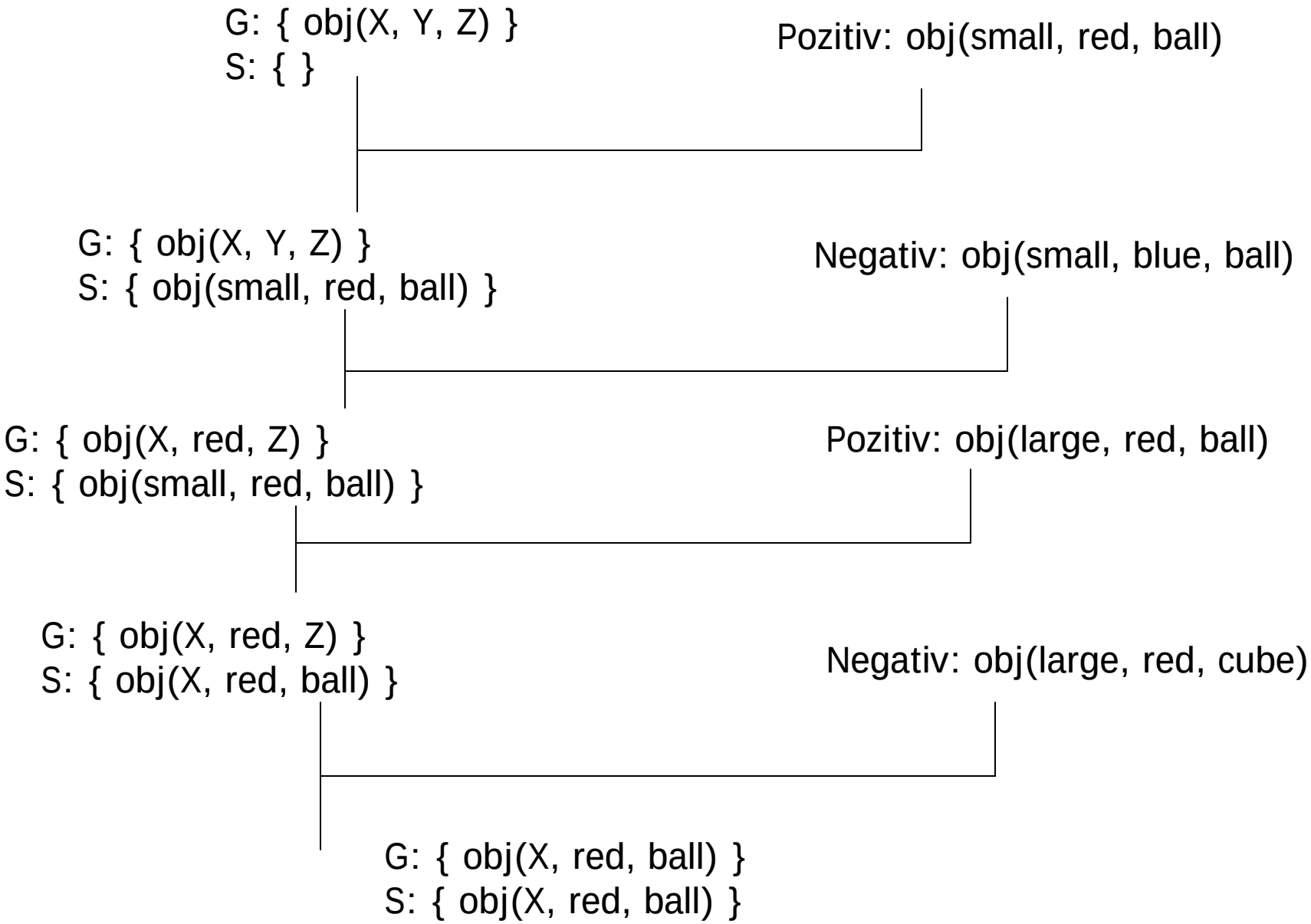
- Elimina din **G** toate ipotezele mai specifice decat alte ipoteze din **S**

4. daca G = **S** si 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 algoritm specific-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 algoritm 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) :-
          generalizset(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]]
```