Tipuri de cunoştinţe

1. Cunoştinţe relaţionale simple

- Cea mai simplă modalitate de reprezentare a faptelor declarative constă în folosirea unei mulţimi de relaţii de acelaşi tip cu cele utilizate în sistemele de baze de date.
- Cunoştinţele relaţionale din acest tabel corespund unei mulţimi de atribute şi de valori asociate, care împreună descriu obiectele bazei de cunoştinţe.

Student	Vârstă	An de studiu	Note la informatică
Popescu Andrei	18	I	8-9
Ionescu Maria	18	I	9-10
Hristea Oana	20	I	7-8
Pârvu Ana	19	II	8-9
Savu Andrei	19	II	7-8
Popescu Ana	20	III	9-10

2. Cunoştinţe care se moştenesc

- Este posibil ca reprezentarea de bază să fie îmbogăţită cu mecanisme de inferenţă care operează asupra structurii reprezentării.
- Pentru ca această modalitate de reprezentare să fie eficientă, structura trebuie proiectată în aşa fel încât ea să corespundă mecanismelor de inferenţă dorite.
- Una dintre cele mai utilizate forme de inferenţă este *moştenirea proprietăţilor*, prin care elemente aparţinând anumitor *clase moştenesc atribute şi valori* provenite de la clase mai generale, în care sunt incluse.
- Pentru a admite moştenirea proprietăţilor, *obiectele* trebuie să fie organizate în *clase*, iar *clasele* trebuie să fie aranjate în cadrul unei *ierarhii*.

3. Cunoştinţe inferenţiale

- Puterea logicii tradiţionale este adesea utilă pentru a se descrie toate inferenţele necesare.
- Astfel de cunoştinţe nu sunt utile decât în prezenţa unei proceduri de inferenţă care să le poată exploata.
- Există multe asemenea proceduri, dintre care unele fac raţionamente de tipul "*înainte*", de la fapte date către concluzii, iar altele raţionează "*înapoi*", de la concluziile dorite la faptele date. Una dintre procedurile cele mai folosite de acest tip este *rezoluţia*, care foloseşte strategia contradicţiei.
- În general, logica furnizează o structură puternică în cadrul căreia sunt descrise legăturile dintre valori. Ea se combină adesea cu un alt limbaj puternic de descriere, cum ar fi o ierarhie de tip *isa*.

4. Cunoştinţe procedurale

- Reprezentarea cunoştinţelor descrisă până în prezent s-a concentrat asupra faptelor statice, declarative.
- Un alt tip de cunoştinţe extrem de utile sunt *cunoştinţele procedurale sau operaţionale*, care specifică ce anume trebuie făcut şi când.
- Cea mai simplă modalitate de reprezentare a cunoştinţelor procedurale este cea sub formă de cod, într-un anumit limbaj de programare.
- În acest caz, maşina foloseşte cunoştinţele atunci când execută codul pentru a efectua o anumită sarcină.
- Acest mod de reprezentare a cunoştinţelor procedurale nu este însă cel mai fericit din punctul de vedere al adecvării inferenţiale, precum şi al eficienţei în achiziţie.

Clase de metode pentru reprezentarea cunoştinţelor

Principalele tipuri de reprezentări ale cunoştinţelor sunt:

1. reprezentările *bazate pe logică*
2. reprezentările de tip "*slot-filler*" ("deschizătură-umplutură")

1. Reprezentările bazate pe logică aparţin unor două mari categorii, în funcţie de instrumentele folosite în reprezentare, şi anume:

   - Logica - mecanismul principal îl constituie inferenţa logică.

   - Regulile (folosite, de pildă, în sistemele expert) - principalele mecanisme sunt "*înlănţuirea înainte*" şi "*înlănţuirea înapoi*". O regulă este similară unei implicaţii logice, dar nu are o valoare proprie (regulile sunt *aplicate*, ele nu au una dintre valorile „true" sau „false").

Limbajul logicii de ordinul I (first order logic - FOL)

Trei lucruri definesc un limbaj declarativ:

- sintaxa – ce grupuri de simboluri sunt valide și în ce ordine

  „mașina pe care o conduc” „conduc mașina o pe care”

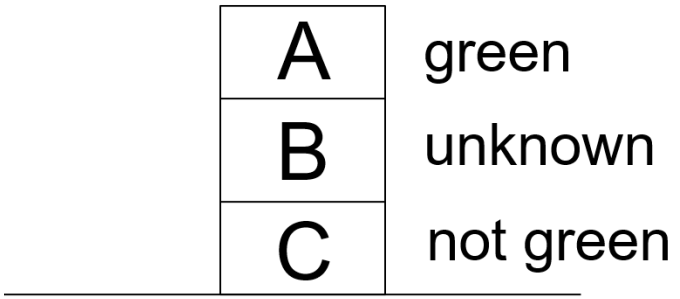
- semantica – ce înseamnă propozițiile bine formate d.p.v. sintactic – unele expresii pot să nu însemne nimic

  „sărbătorile albastre aleargă”

- componenta pragmatică – care este sensul dorit al expresiilor

  „este cineva în spatele tău”

Exemplu de problemă



Is there a green block directly on top of a non-green one?

Formalizarea în FOL

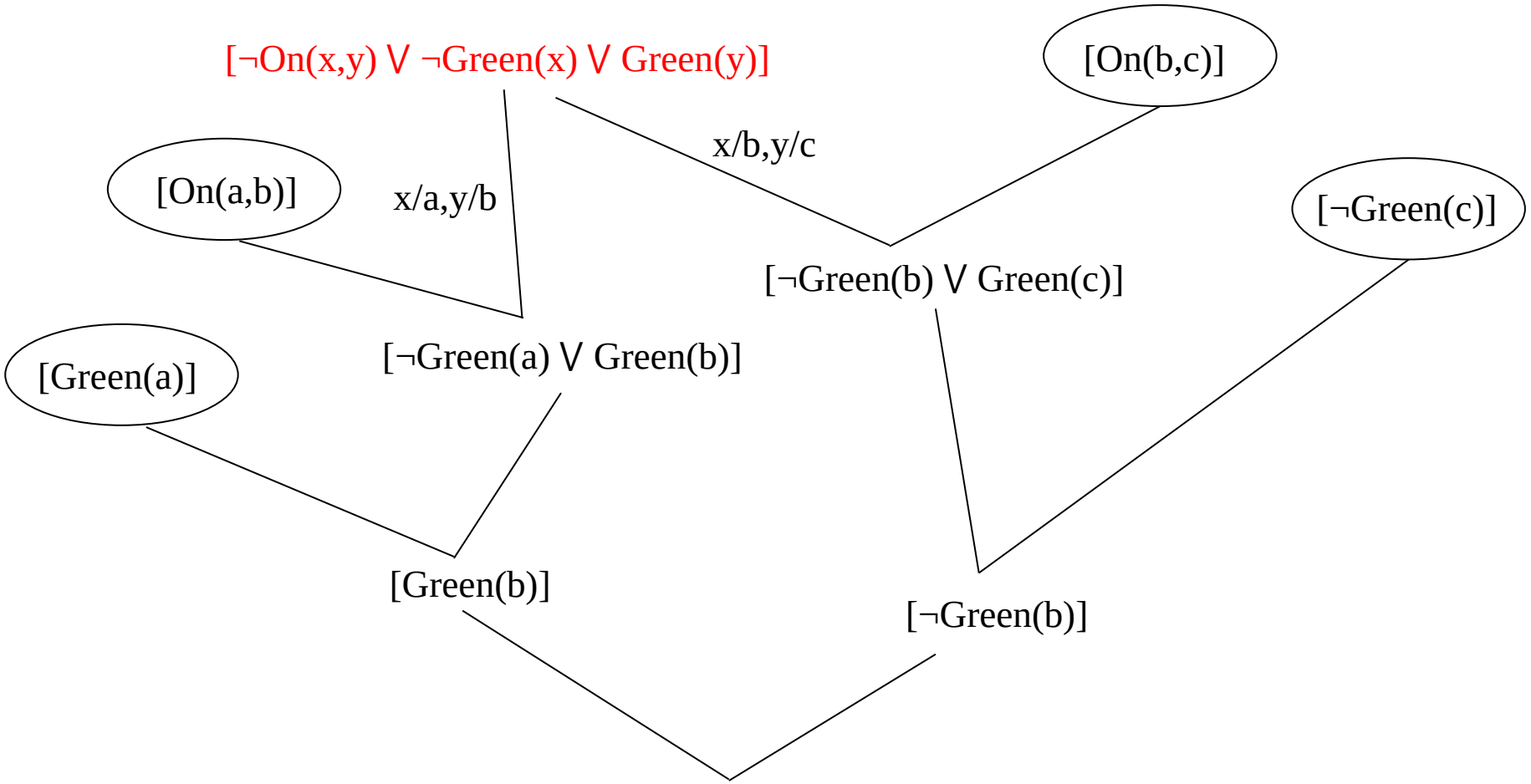
- a, b, c numele blocurilor
- G simbolul predicatului unar ce reprezintă “green”
- O simbolul predicatului binar ce reprezintă “on”

BC (baza de cunoştinţe) = {O(a,b), O(b,c), G(a), ¬G(c)}}

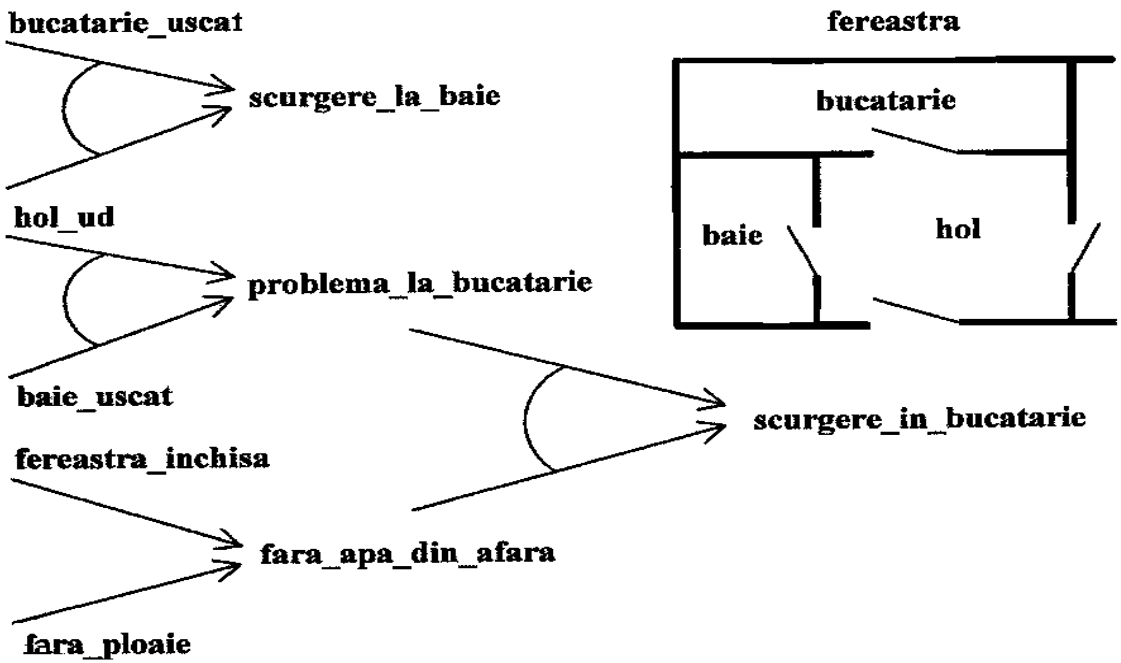
Propoziţia pe care dorim să o deducem logic din BC:

$$\exists x \exists y.\ G(x) \wedge \neg G(y) \wedge O(x,y).$$

Rezoluţia


[¬On(x,y) ∨ ¬Green(x) ∨ Green(y)]
[On(b,c)]
x/b,y/c
[On(a,b)]
x/a,y/b
[¬Green(c)]
[¬Green(b) ∨ Green(c)]
[Green(a)]
[¬Green(a) ∨ Green(b)]
[Green(b)]
[¬Green(b)]
[ ] (disjuncţie de nicio posibilitate –
reprezentare pentru FALS)

Înlănţuirea înapoi/înainte



Faptele observate sunt: hol_ud, baie_uscat şi fereastra_inchisa

Întrebare (scop): scurgere_in_bucatarie

Înlănţuirea înapoi/înainte

IF bucatarie_uscat AND hol_ud THEN scurgere_la_baie
IF hol_ud AND baie_uscat THEN problema_la_bucatarie
IF fereastra_inchisa OR fara_ploaie THEN fara_apa_din_afara
IF problema_la_bucatarie AND fara_apa_din_afara THEN scurgere_in_bucatarie.

Înlănţuirea înapoi: se pleacă de la Întrebare spre Faptele observate, folosind regulile

Înlănţuirea înainte: se pleacă de la Faptele observate spre Întrebare, folosind regulile

Cunoştinţe procedurale

- Specifică ce anume trebuie făcut şi când.
- Cea mai folosită tehnică de reprezentare a cunoştinţelor procedurale în programele de inteligenţă artificială este aceea a utilizării *regulilor de producţie*.
- Atunci când sunt îmbogăţite cu informaţii asupra felului în care trebuie să fie folosite, regulile de producţie sunt mai procedurale decât alte metode existente de reprezentare a cunoştinţelor.
- Regulile de producţie, numite şi reguli de tip if-then, sunt instrucţiuni condiţionale, care pot avea diverse interpretări, cum ar fi:
  - if precondiţie P then concluzie C
  - if situaţie S then acţiune A
  - if condiţiile C1 şi C2 sunt verificate then condiţia C nu este verificată
- Regulile de producţie sunt foarte utilizate în proiectarea *sistemelor expert*.

- Regulile de tip if-then adesea definesc relaţii logice între conceptele aparţinând domeniului problemei. Relaţiile pur logice pot fi caracterizate ca aparţinând aşa-numitelor *cunoştinţe categorice*, adică acelor cunoştinţe care vor fi întotdeauna adevărate.

- În unele domenii, cum ar fi diagnosticarea în medicină, predomină cunoştinţele "*moi*" sau *probabiliste*. În cazul acestui tip de cunoştinţe, regularităţile empirice sunt valide numai până la un anumit punct (adesea, dar nu întotdeauna). În astfel de cazuri, regulile de producţie sunt modificate prin adăugarea la interpretarea lor logică a unei calificări de verosimilitate, obţinându-se reguli de forma:

if conditie A then concluzie B cu certitudinea F

unde:

F = *factor de certitudine*, *măsură a încrederii* sau *certitudine subiectivă*

Pentru calculul lui F: statistica Bayesiana

Reguli în sisteme de producţie

Sistemele de producție formalizează cunoștințele prin reguli de producție și folosesc înlănțuirea înainte pentru a obține cunoștințe noi.

Un sistem de producție menține o memorie de lucru (WM) care conține aserțiuni ce se modifică în timpul funcționării sistemului.

Memorie de lucru WM constă dintr-o mulțime de elemente ale memoriei de lucru (WME).

Un WME are forma (tip atribut1:val1...atributn:valn)

Exemple

(persoana varsta:21 localitate:București)

(student nume:daniel dept:informatica)

O regulă de producție constă dintr-un set de condiții și un set de acțiuni:

IF condiții THEN acțiuni

Condițiile unei reguli sunt legate prin conjuncții.

Setul de acțiuni ale regulilor de producție au o interpretare procedurală. Toate acțiunile sunt executate secvențial și pot fi de următoarele tipuri:

- ADD – adaugă un nou WME la WM
- REMOVE *i* – elimină din WM WME-ul care se potrivește cu a *i*-a condiție a regulii; nu se aplică dacă condiția este negativă
- MODIFY *i* (atribut specificație) – modifică WME-ul care se potrivește cu a *i*-a condiție, prin înlocuirea valorii curente a atributului cu specificația; nu se aplică dacă condiția este negativă.

Sistemul de reguli de producție funcționează într-un ciclu în trei etape, care se repetă până când nu mai sunt reguli aplicabile pentru WM:

1. Recunoaștere – se identifică regulile aplicabile, adică regulile ale căror condiții sunt îndeplinite de actualul WM
2. Rezolvare conflicte – dintre regulile găsite la primul pas, se aleg cele care se vor executa, conform unui criteriu prestabilit (de ex., prima regula)
3. Acțiune – se modifică WM-ul efectuând acțiunile tuturor regulilor selectate la pasul 2

Exemplu 1

IF (student nume:x) THEN ADD (persoana nume:x)

(echivalentul implicaţiei din FOL $\forall x.\ \text{Student}(x) \longrightarrow \text{Persoana}(x)$)

Exemplu 2 – presupunând ca ‘cineva’ adaugă un WME de tipul zi_nastere

IF (persoana varsta:x nume:n) (zi_nastere cine:n)

THEN MODIFY 1 (varsta [x+1])

REMOVE 2

Reprezentarea cunoştinţelor în sistemele expert

- Un *sistem expert* este un program care se comportă ca un expert într-un domeniu relativ restrans.
- Caracteristica majoră a sistemelor expert, numite şi *sisteme bazate pe cunoştinţe*, este aceea că ele se bazează pe cunoştinţele unui expert uman în domeniul care este studiat.
- La baza sistemelor expert se află utilizarea în rezolvarea problemelor a unor mari cantităţi de cunoştinţe specifice domeniului.

Alte caracteristici ale sistemului expert:

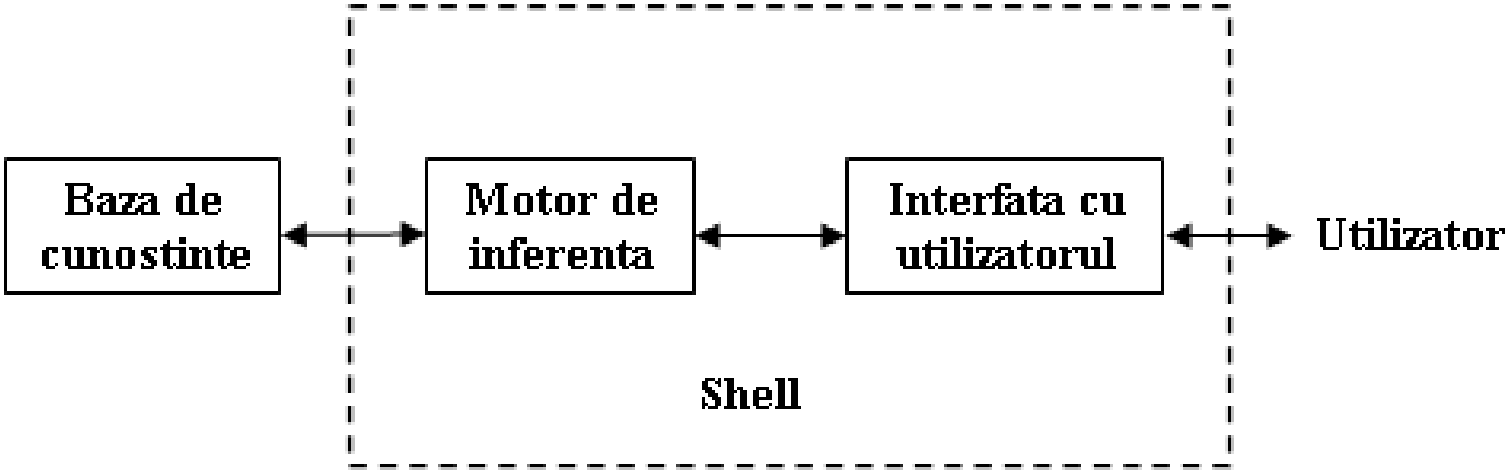
- să fie capabil *să explice* comportamentul său şi deciziile luate - la fel cum o fac experţii umani - prin generarea răspunsului pentru două tipuri de întrebări ale utilizatorului:
  - ✓ întrebare de tipul "*cum*": *Cum* ai ajuns la această concluzie?
  - ✓ întrebare de tipul "*de ce*": *De ce* te interesează această informaţie?

- să lucreze cu *incertitudinea* sau *starea de a fi incomplet* - informaţii incerte, incomplete sau care lipsesc; relaţii aproximative în domeniul problemei (de ex., efectul unui medicament asupra stării pacientului).

Structura de bază a unui sistem expert

Un sistem expert conţine trei module principale, şi anume:

- o bază de cunoştinţe;
- un motor de inferenţă;
- o interfaţă cu utilizatorul.



Concluzii

- Regulile if-then formează lanţuri de forma

  informatie input →...→ informatie dedusa

- Informaţia de tip input mai poartă denumirea de *date* sau *manifestări*.
- Informaţia dedusă constituie *ipotezele* care trebuie demonstrate sau *cauzele manifestărilor* sau *diagnostice* sau *explicaţii*.
- Atât înlănţuirea înainte, cât şi cea înapoi (ca metode de inferenţă) presupun *căutare*, dar direcţia de căutare este diferită pentru fiecare în parte.
- Înlănţuirea înapoi execută o căutare de la scopuri înspre date, din care cauză se spune despre ea că este *orientată către scop*.
- Înlănţuirea înainte caută pornind de la date înspre scopuri, fiind *orientată către date*.

Exemple de sisteme expert

MYCIN – sistem expert de diagnosticare a infecţiilor bacteriene, dezvoltat la Universitatea Stanford în ‘70

– 500 de reguli de producție pentru recunoașterea a 100 de cauze ale infecțiilor

– cea mai semnificativă contribuție a fost introducerea unui nivel de incertitudine a faptelor

XCON – sistem bazat pe reguli pentru configurarea computerelor, dezvoltat la Universitatea Carnegie Mellon în 1978

– 10000 de reguli pentru a descrie sute de tipuri de componente

– a contribuit la creșterea interesului comercial pentru sistemele expert bazate pe reguli.

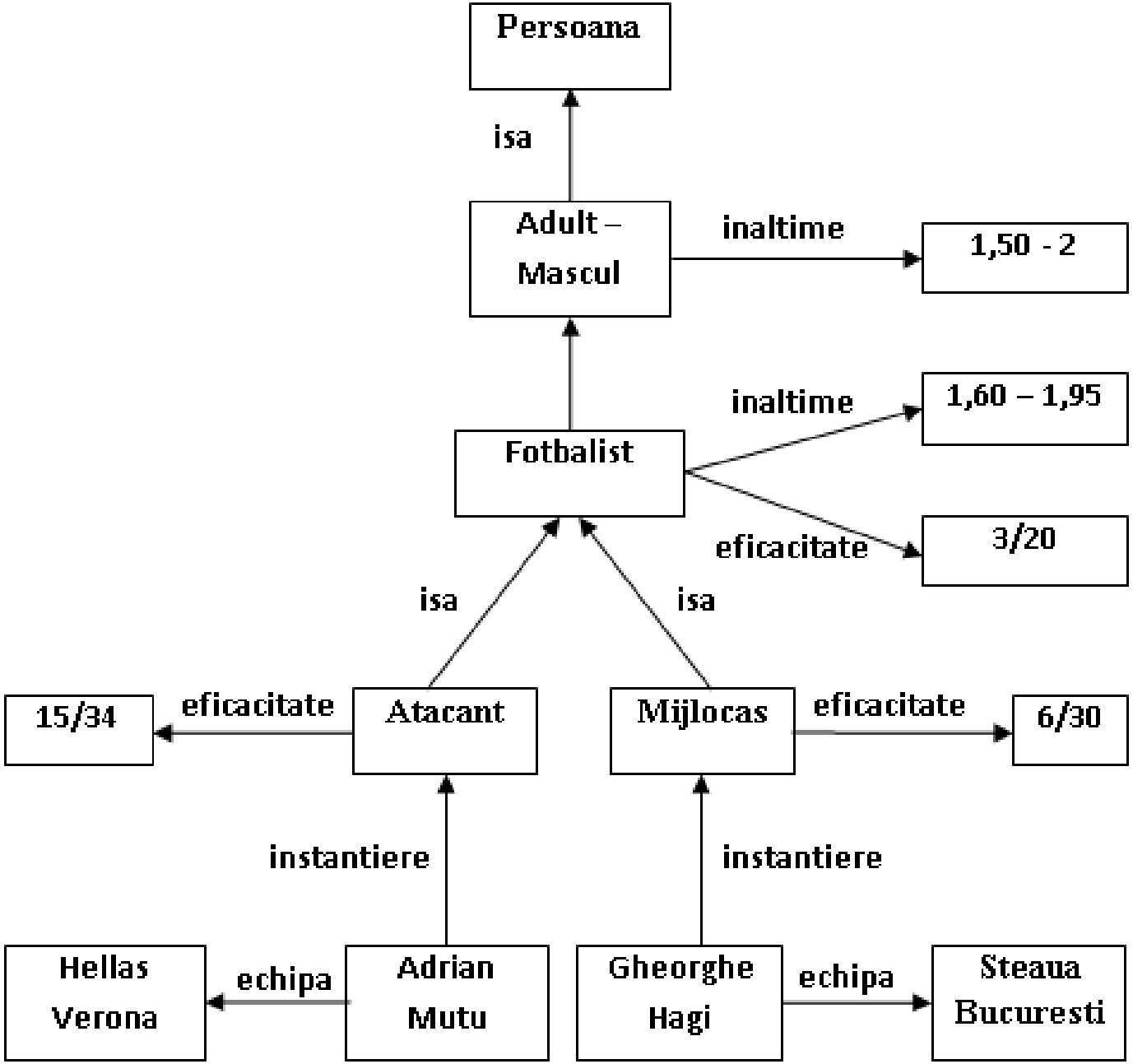
2. Reprezentările de tip slot-filler folosesc două categorii diferite de structuri:

- Reţele semantice şi grafuri conceptuale - o reprezentare distribuită (concepte legate între ele prin diverse relaţii). Principalul mecanism folosit este *căutarea*.

- Cadre şi scripturi - o reprezentare structurată (grupuri de concepte şi relaţii); sunt foarte utile în reprezentarea tipicităţii. Principalul mecanism folosit este *împerecherea* (potrivirea) *şabloanelor* (tiparelor) – „pattern matching”.

În figura care urmează, sunt reprezentate cunoştinţe legate de jocul de fotbal, cunoştinţe organizate într-o structură de acest tip. În această reprezentare, liniile desemnează atribute. Nodurile figurate prin dreptunghiuri reprezintă obiecte şi valori ale atributelor obiectelor.

- ✓ Aceste valori pot fi, la rândul lor, privite ca obiecte având atribute şi valori ş.a.m.d..
- ✓ Săgeţile corespunzătoare liniilor sunt orientate de la un obiect la valoarea lui (de-a lungul liniei desemnând atributul corespunzător).
- ✓ Toate obiectele şi majoritatea atributelor care intervin corespund domeniului sportiv al jocului de fotbal şi nu au o semnificaţie generală. Singurele două excepţii sunt atributul *isa*, utilizat pentru a desemna *incluziunea între clase* şi atributul *instanţiere*, folosit pentru a arăta *apartenenţa la o clasă*. Aceste două atribute, extrem de folosite, se află la baza *moştenirii proprietăţilor* ca tehnică de inferenţă.


Persoana
isa
Adult – Mascul
inaltime
1,50 - 2
inaltime
1,60 – 1,95
Fotbalist
eficacitate
3/20
isa
isa
15/34
eficacitate
Atacant
Mijlocas
eficacitate
6/30
instantiere
instantiere
Hellas Verona
echipa
Adrian Mutu
Gheorghe Hagi
echipa
Steaua Bucuresti

Utilizând această tehnică de inferenţă, *baza de cunoştinţe* poate asigura atât regăsirea faptelor care au fost memorate în mod explicit, precum şi a faptelor care derivă din cele memorate în mod explicit, ca în următorul exemplu:

eficacitate(Adrian_Mutu) = 15/34

Este una dintre cele mai folosite tehnici de inferenţă!

- În acest exemplu, structura corespunzătoare reprezintă o *structură de tip* "*slot-and-filler*". Ea mai poate fi privită şi ca o *reţea semantică* sau ca o *colecţie de cadre*.

- În cazul *colecţiei de cadre*, fiecare cadru individual reprezintă colecţia atributelor şi a valorilor asociate cu un anumit nod.

- O diferenţiere exactă a acestor tipuri de reprezentări este greu de făcut. În general, termenul de sistem de cadre implică existenţa unei mai mari structurări a atributelor şi a mecanismelor de inferenţă care le pot fi aplicate decât în cazul reţelelor semantice.

Cadre (*eng. Frame*s) - pot fi generice sau instanţe

(Nume_cadru
<slot1 filler1>
<slot2 filler2>
…)

Exemple

(Fotbalist
<:ISA Adult-Mascul>
<:Inaltime InaltimePosibila>
<:Eficacitate 3/20> …)

(Atacant
<:ISA Fotbalist>
<:Eficacitate 15/34>…)

(adrianmutu
<:INSTANCEOF Atacant>
<:Inaltime h180>
<:Echipa hellas_verona> …)

Slot-urile cadrelor generice pot avea ataşate proceduri de tip **IF-ADDED** sau **IF-NEEDED**.

(Fotbalist
<:TotalCost [IF-NEEDED ComputeTotalCost]>…)

(Atacant
<:Sponsor >
<:Echipa [IF-ADDED GetSponsor]>…)

Sistemul de cadre funcționează într-o ciclu în trei pași:

1. Cineva (un utilizator, o procedură, un sistem extern) creează un obiect, prin instanțierea un cadru generic.
2. Orice filler care nu este furnizat în mod explicit, dar care poate fi moștenit de noua instanță, este moștenit.
3. Pentru fiecare slot cu un filler, dacă o procedură IF-ADDED poate fi moștenită, atunci aceasta este executată. Prin executarea acesteia, se pot completa noi slot-uri sau pot fi instanțiate noi cadre; apoi treci la 1.

Dacă un utilizator, un sistem extern sau o procedură atașată solicită un filler, atunci:

1. Dacă filler-ul există, atunci valoarea este returnată;
2. În caz contrar, orice procedură IF-NEEDED care poate fi moștenită este executată, calculând filler-ul. Execuția procedurii poate, de asemenea, calcula si alte filler-e sau instanția cadre noi.

   Dacă pasul de mai sus nu produce niciun rezultat, atunci valoarea slot-ului ramâne necunoscută.