

Reprezentarea cunoștințelor

Un bun sistem de reprezentare a cunoștințelor într-un anumit domeniu ar trebui să posede următoarele patru proprietăți:

- Adecvare în reprezentare - abilitatea de a reprezenta toate tipurile de cunoștințe care sunt necesare într-un anumit domeniu.
- Adecvare inferențială - abilitatea de a manipula structurile reprezentationale într-un asemenea mod încât să poată fi derivate structuri noi, corespunzătoare cunoștințelor noi deduse din cele vechi.
- Eficiența inferențială - abilitatea de a încorpora în structura de cunoștințe informații adiționale care să poată fi folosite pentru a canaliza atenția mecanismului de inferență în direcțiile cele mai promițătoare.

- **Eficiența în achiziție** - abilitatea de a achiziționa ușor informații noi. În mod ideal, însuși programul ar trebui să poată controla achiziția de cunoștințe.

Nici un sistem unic nu poate optimiza toate aceste caracteristici, nu în ultimul rând datorită tipurilor extrem de diverse de cunoștințe care există, majoritatea programelor bazându-se pe tehnici multiple.

Tipuri de cunoștințe

Cunoștințe relaționale simple

Cea mai simplă modalitate de reprezentare a faptelor declarative constă în folosirea unei mulțimi de relații de același tip cu cele utilizate în sistemele de baze de date. Un exemplu de sistem relațional este cel din Fig. 4.1. Cunoștințele relaționale din acest tabel corespund unei mulțimi de atribute și de valori asociate, care împreună descriu obiectele bazei de cunoștințe.

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

Fig. 4.1

Această reprezentare este simplă deoarece, în sine, nu posedă și nu furnizează capacitatea de inferență. Dar cunoștințele reprezentate în acest mod pot constitui input-ul adecvat pentru motoare de inferență mult mai puternice. Sistemele de baze de date sunt proiectate tocmai cu scopul de a furniza suportul necesar cunoștințelor relaționale.

Cunoștințe care se moștenesc

Este posibil ca reprezentarea de bază să fie îmbogățită cu mecanisme de inferență care operează asupra structurii reprezentării. Pentru ca această modalitate de reprezentare să fie eficientă, structura trebuie proiectată în așa fel încât ea să corespundă mecanismelor de inferență dorite. Una dintre cele mai utilizate forme de inferență este moștenirea proprietăților, prin care elemente aparținând anumitor clase moștenesc attribute și valori provenite de la clase mai generale, în care sunt incluse. Pentru a admite moștenirea proprietăților, obiectele trebuie să fie organizate în clase, iar clasele trebuie să fie aranjate în cadrul unei ierarhii. În Fig. 4.2 sunt reprezentate cunoștințe legate de jocul de fotbal, cunoștințe organizate într-o structură de acest tip. În această reprezentare, liniile desemnează

atribute. Nodurile figurate prin dreptunghiuri reprezintă obiecte și valori ale atributelor obiectelor. Aceste valori pot fi, la rândul lor, privite ca obiecte având atribute și valori ș.a.m.d.. Săgețile corespunzătoare liniilor sunt orientate de la un obiect la valoarea lui (de-a lungul liniei desemnând atributul corespunzător).

În exemplul din Fig. 4.2., toate obiectele și majoritatea atributelor care intervin corespund domeniului sportiv al jocului de fotbal și nu au o semnificație generală. Singurele două excepții sunt atributul *isa*, utilizat pentru a desemna *incluziunea între clase* și atributul *instantiere*, folosit pentru a arăta *apartenența la o clasă*. Aceste două atribute, extrem de folosite, se află la baza *moștenirii proprietăților* ca tehnică de inferență.

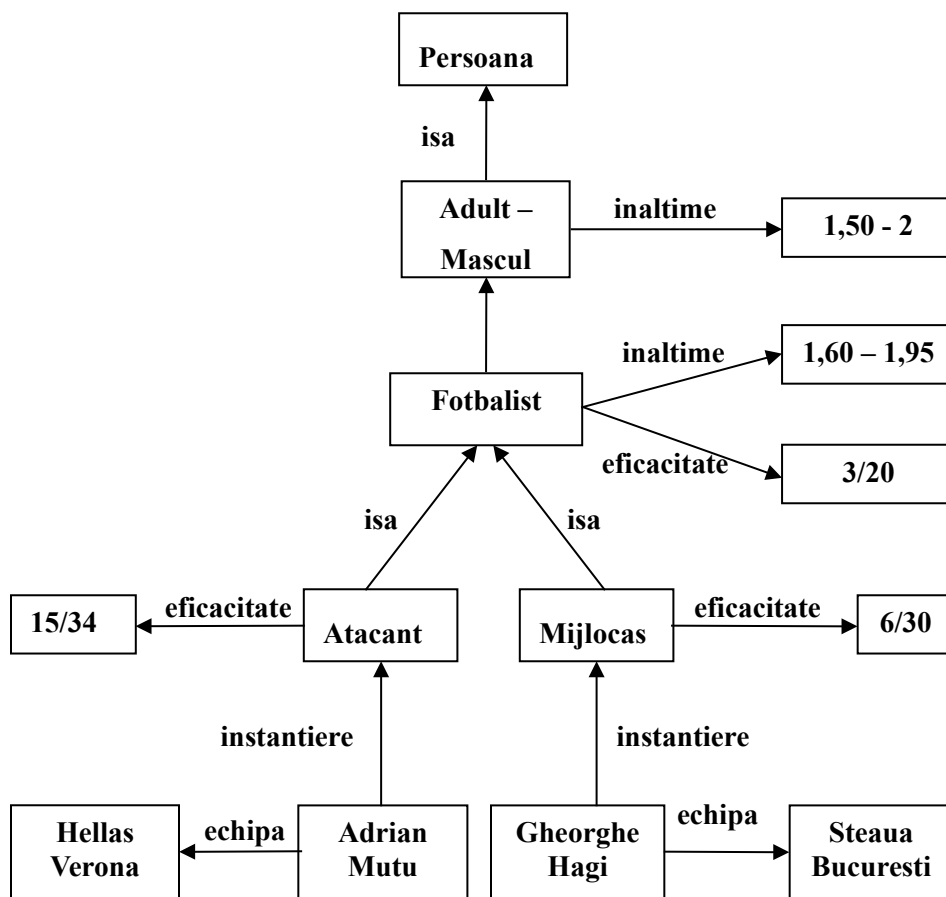


Fig. 4.2

Utilizând această tehnică de inferență, baza de cunoștințe poate asigura atât regăsirea faptelor care au fost memorate în mod explicit, precum și a faptelor care derivă din cele memorate în mod explicit, ca în următorul exemplu:

$$\text{eficacitate(Adrian_Mutu)} = 15/34$$

Pentru a găsi răspunsul la această interogare, întrucât nu există nici o valoare pentru eficacitate memorată în mod explicit corespunzător lui Adrian_Mutu, a fost urmat atributul *instanțiere* până la *Atacant* și a fost extrasă valoarea memorată acolo. Se poate acum observa una dintre caracteristicile negative ale moștenirii proprietăților, și anume aceea că această tehnică de inferență poate produce valori implicite care nu sunt garantate a fi corecte, dar care reprezintă cele mai bune aproximări în lipsa unor informații exacte. Această tehnică de inferență continuă să fie printre cele mai folosite.

În acest exemplu, structura corespunzătoare reprezintă o *structură de tip “slot-and-filler”*. Ea mai poate fi privită și ca o *rețea semantică* sau ca o *colecție de cadre*. În cel din urmă caz, fiecare cadru individual reprezintă colecția atributelor și a valorilor asociate cu un anumit nod. O diferențiere exactă a acestor tipuri de reprezentări este greu de făcut datorită mării flexibilități care există în reprezentarea cunoștințelor. În general, termenul de sistem de cadre implică existența unei mai mari structurări a atributelor și a mecanismelor de inferență care le pot fi aplicate decât în cazul rețelelor semantice.

Moștenirea proprietăților este o formă de inferență puternică, dar nu este suficientă pentru a construi un sistem de reprezentare complet, care, de cele mai multe ori, combină mai multe tehnici de reprezentare a cunoștințelor.

Cunoștințe inferențiale

Puterea logicii tradiționale este adesea utilă pentru a se descrie toate inferențele necesare. Astfel de cunoștințe nu sunt utile decât în prezența unei proceduri de inferență care să le poată exploata. Procedura de inferență necesară în acest caz este una care implementează regulile logice de inferență standard. Există multe asemenea proceduri, dintre care unele fac raționamente de tipul “*înainte*”, de la fapte date către concluzii, iar altele raționează “*înapoi*”, de la concluziile dorite la faptele date. Una dintre procedurile cele mai folosite de acest tip este rezoluția, care folosește strategia contradicției.

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.**

Cunoștințe procedurale

Reprezentarea cunoștințelor descrisă până în prezent s-a concentrat asupra faptelor statice, declarative. Un alt tip de cunoștințe extrem de utile sunt cunoștințele procedurale sau operaționale, care specifică ce anume trebuie făcut și când.

Cunoștințele procedurale pot fi reprezentate în programe în diverse moduri. Cea mai simplă modalitate este cea sub formă de cod, într-un anumit limbaj de programare. În acest caz, mașina folosește cunoștințele atunci când execută codul pentru a efectua o anumită sarcină. Acest mod de reprezentare a cunoștințelor procedurale nu este însă cel mai fericit din punctul de vedere al adecvării inferențiale, precum și al eficienței în achiziție. Din această cauză s-au căutat alte modalități, diferite, de reprezentare

a cunoștințelor procedurale, astfel încât acestea să poată fi manipulate relativ ușor atât de către oameni, cât și de către alte programe.

Cea mai folosită tehnică de reprezentare a cunoștințelor procedurale în programele de inteligență artificială este aceea a utilizării regulilor de producție. Atunci când sunt îmbogățite cu informații asupra felului în care trebuie să fie folosite, regulile de producție sunt mai procedurale decât alte metode existente de reprezentare a cunoștințelor.

Regulile de producție, numite și reguli de tip if-then, sunt instrucțiuni condiționale, care pot avea diverse interpretări, cum ar fi:

- if precondiție P then concluzie C
- if situație S then acțiune A
- if condițiile C1 și C2 sunt verificate then condiția C nu este verificată.

Regulile de producție sunt foarte utilizate în proiectarea sistemelor expert.

A face o distincție clară între cunoștințele declarative și cele procedurale este foarte dificil. Diferența esențială între ele este dată de modul în care cunoștințele sunt folosite de către procedurile care le manipulează.

Clase de metode pentru reprezentarea cunoștințelor

Principalele tipuri de reprezentări ale cunoștințelor sunt reprezentările *bazate pe logică* și cele de tip “*slot-filler*” (“deschizătură-umplutură”).

Reprezentările bazate pe logică aparțin unor două mari categorii, în funcție de instrumentele folosite în reprezentare, și anume:

- **Logica** - mecanismul principal îl constituie inferența logică.
- **Regulile** (folosite, de pildă, în sistemele expert)
 - principalele mecanisme sunt “înlănțuirea înainte” și “înlănțuirea înapoi”. O regulă este similară unei implicații logice, dar nu are o valoare proprie (regulile sunt aplicate, ele nu au una dintre valorile „true” sau „false”).

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

- Rețele semantice și grafuri conceptuale - o reprezentare “distribuită” (concepte legate între ele prin diverse relații). Principalul mecanism folosit este *căutarea*.
- Cadre și scripturi - o reprezentare structurată (grupuri de concepte și relații); sunt foarte utile în reprezentarea tipicității. Principalul mecanism folosit este împerecherea (potrivirea) șabloanelor (tiparelor).

Reprezentarea cunoștințelor și sistemele expert

Un sistem expert este un program care se comportă ca un expert într-un domeniu relativ restrâns. Caracteristica majoră a sistemelor expert, numite și sisteme bazate pe cunoștințe, este aceea că ele se bazează pe cunoștințele unui expert uman în domeniul care este studiat. Mai exact, ele se sprijină pe cunoștințele expertului uman asupra strategiilor de rezolvare a problemelor specifice domeniului. Astfel, la baza sistemelor expert se află utilizarea în rezolvarea problemelor a unor mari cantități de cunoștințe specifice domeniului.

Sistemele expert trebuie să poată rezolva probleme care necesită cunoștințe într-un anumit domeniu. Prin urmare, ele trebuie să posedă aceste cunoștințe într-o anumită formă. Din această cauză ele se mai numesc și sisteme bazate pe cunoștințe. Nu toate sistemele bazate pe cunoștințe constituie însă sisteme expert. Un sistem expert trebuie, în plus, să fie capabil să explice utilizatorului comportamentul său și deciziile luate, la fel cum o fac experții umani. Această caracteristică referitoare la generarea explicațiilor este necesară în special în domeniile în care intervine incertitudinea, cum ar fi diagnosticarea medicală. Numai în acest fel utilizatorul poate detecta o posibilă fisură în raționamentul sistemului.

O caracteristică suplimentară care este adesea cerută sistemelor expert este, prin urmare, abilitatea de a trata incertitudinea sau starea de a fi incomplet. Astfel, informațiile asupra problemei care trebuie rezolvată pot fi incomplete sau de natură a nu inspira încredere, iar relațiile din domeniul problemei pot fi aproximative. Spre exemplu, un anumit medicament poate genera anumite probleme, dar *cel mai adesea* nu o face.

Pentru a construi un sistem expert este necesară, în general, definirea următoarelor două funcții:

- **o funcție de rezolvare a problemei**, funcție capabilă să folosească cunoștințele specifice domeniului și care trebuie să poată trata incertitudinea.
- **o funcție de interacțiune cu utilizatorul**, care asigură și generarea de explicații asupra intențiilor sistemului și a deciziilor luate de acesta în timpul și după procesul de rezolvare a problemei.

Fiecare dintre aceste funcții poate fi foarte complicată și depinde în mod nemijlocit de domeniul de aplicație și de cerințele practice care pot să apară.

Structura de bază a unui sistem expert

Un sistem expert conține trei module principale, și anume:

- **o bază de cunoștințe;**
- **un motor de inferență;**
- **o interfață cu utilizatorul.**

- Baza de cunoștințe cuprinde cunoștințele specifice unui domeniu, inclusiv fapte simple referitoare la domeniul de aplicație, reguli care descriu relațiile și fenomenele proprii domeniului și, eventual, metode, euristici și idei pentru rezolvarea problemelor domeniului.
- Motorul de inferență utilizează în mod activ cunoștințele din baza de cunoștințe.
- Interfața cu utilizatorul asigură comunicarea dintre utilizator și sistem, oferind utilizatorului o privire asupra procesului de rezolvare a problemei executat de către motorul de inferență. De cele mai multe ori motorul de inferență și interfața sunt privite ca un unic modul, numit shell (înveliș).

Structura de bază a unui sistem expert este prezentată în Fig. 4.3:

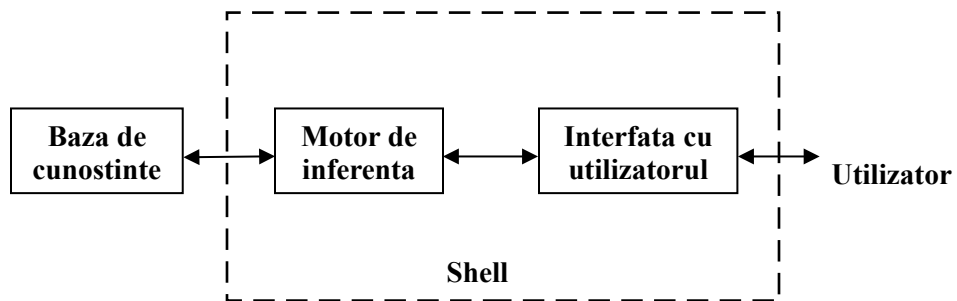


Fig. 4.3

Schema din Fig. 4.3 separă cunoștințele de algoritmi care folosesc aceste cunoștințe. Această separare este convenabilă din următoarele motive: baza de cunoștințe depinde în mod clar de aplicație. Pe de altă parte, învelișul este, în principiu, independent de domeniu. O modalitate de a dezvolta sisteme expert pentru diverse aplicații, în aceste condiții, constă din dezvoltarea unui *shell* care poate fi folosit în mod universal, cu utilizarea unei baze de cunoștințe diferite corespunzător fiecărei aplicații. Desigur, toate bazele de cunoștințe utilizate vor trebui să se conformeze unui același formalism care este înțeles de către shell. În practică, această abordare nu va avea succes decât în cazul în care domeniile de aplicație sunt extrem de similare, dar, chiar dacă modificări ale învelișului vor fi necesare, atunci când se trece de la un domeniu la altul, principiile de bază pot fi reținute.

