UNIVERSITATEA "POLITEHNICA" DIN TIMISOARA

ing. NICOLAE MIRCEA DEHELEAN

- teză de doctorat -

Conducător ştiinţific :

Prof. dr. ing. NICOLAE GHEORGHIU

1998

CUPRINS

SCOPUL LUCRARII SI IMPORTANTA TEMEI

Activitățile inginereşti sunt subordonate creației tehnice. Ele necesită, atât capacitate novatoare, cât şi rigoare ştiințifică. Creația tehnică are ca scop, atât obiectul tehnic, cât şi tehnologia de fabricație.

In secolul al XX-lea, importanța tehnologiei a devenit majoră, pentru că performanțele tehnologice au ajuns să determine în mod esențial, performanțele produselor tehnice, succesul economic al întreprinderii şi progresul general al tehnicii.

Performanțele activităților inginereşti sunt legate şi de folosirea intensivă în acest domeniu, a produselor software şi a tehnicii de calcul. Sintagma "tehnică de calcul" este depăşită, deoarece computerul, calculatorul electronic, a depăşit etapa în care era capabil să efectueze numai calcule şi a pătruns într-o etapă nouă, în care efectuează judecăți şi ia decizii, operațiile de calcul devenind auxiliare.

Evoluția produselor software conferă calculatorului un comportament inteligent. Această inteligență pe "purtătoare artificială" devine un instrument capabil să organizeze şi să dea performanță, oricărui domeniu în care este folosită.

Lucrarea îşi propune să găsească o metodă adecvată de organizare a cunoştințelor şi de dezvoltare a Sistemelor Expert, dedicate activităților de proiectare, în general şi a proiectării transmisiilor mecanice, în particular.

Inteligența Artificială a devenit cel mai perfecționat instrument, pe care omul şi l-a construit spre a-i fi de mare folos. Privind înapoi spre drumul parcurs, începând cu prima piatră cioplită, nu avem decât să fim uimiți. Privind înainte, putem fi încrezători.

1. STADIUL ACTUAL AL ACTIVITATILOR DE PROIECTARE IN CONSTRUCTIA DE MASINI

1.1. CONTINUTUL SI CONEXIUNILE ACTIVITATILOR DE PROIECTARE CONSTRUCTIVA

Analiza titlului lucrării, **Sistem Expert dedicat proiectării Transmisiilor Mecanice,** cere dintr-un început stabilirea conexiunilor necesare acelor activităţi de proiectare, care se înscriu în domeniul producţiei şi exploatării maşinilor. Transmisiile Mecanice sunt subansamble, care intră în componenţa maşinilor şi instalaţiilor, cu importanţă funcţională majoră. Importanţa majoră rezultă din observaţia că, în stadiul actual al tehnicii, transmisiile mecanice nu pot fi eliminate din componenţa maşinilor şi instalaţiilor, în condiţiile păstrării performanţelor acestora.

Creşterea explozivă a complexităţii structurale şi funcţionale a maşinilor cere ca acţiunea de realizare fizică a lor, de producţie, să fie precedată de o amplă activitate de anticipare conceptuală, a ceea ce se va pune în operă. Activitatea aceasta de *planificare* şi *prognozare*, pe care o numim generic proiectare, trebuie să aibă temei. Temeiurile cele mai acceptate sunt rezultatele cercetărilor ştiinţifice şi experienţa în domeniu. Conexiunile activităţii de proiectare pot fi decelate prin: luarea în considerare a apartenenţei la un grup mai larg de activităţi, prospectarea conţinutului noţiunii, detalierea algoritmului general de funcţionare şi prin cercetarea instrumentelor folosite (Fig.1.1).

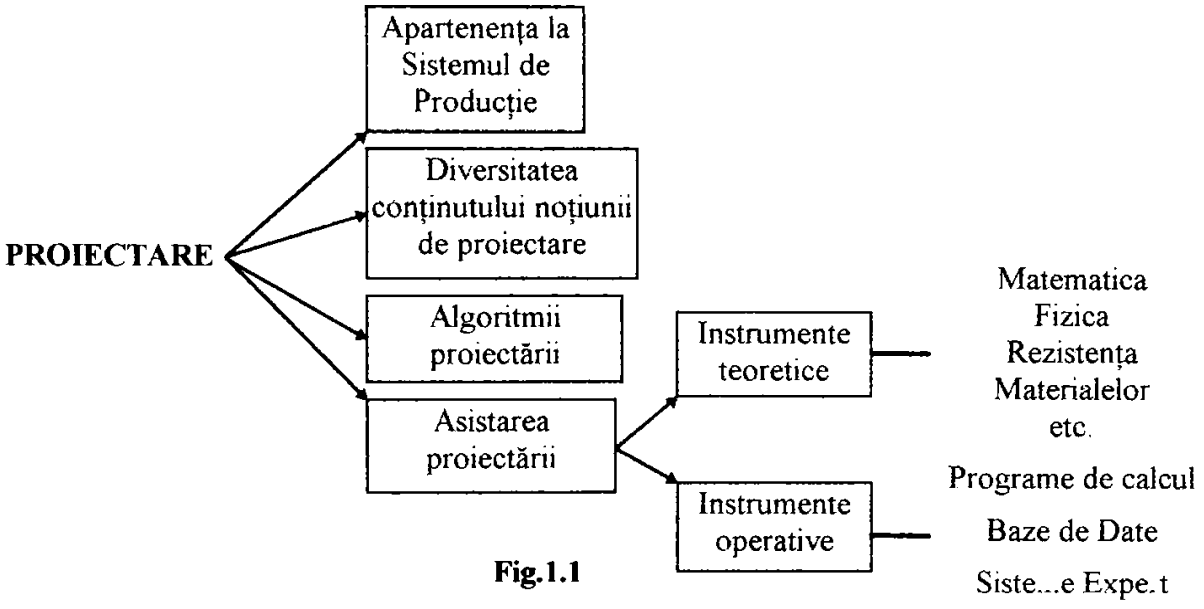


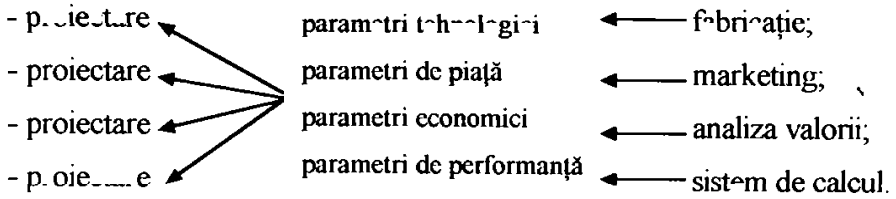
Fig.1.1

Apartenenţa la Sistemul de Producţie. Din punct de vedere al interconexiunilor, proiectarea constructivă a maşinilor este parte integrantă a grupului de discipline, care astăzi poartă numele generic de Stiinţa Producţiei sau Productică. Scopul acestei activităţi este de a obţine, în condiţii de performanţă, un obiect tehnic, care vândut pe piaţă, să conducă la succes economic. Scopul particular al proiectării este subordonat scopului global al producticii. Proiectarea se preocupă mai mult de 2 grupe de parametri, care se referă la:

- performanţele structurale şi funcţionale ale produsului;
- performanţele tehnologiei de fabricaţie.

Productica se ocupă de performanţele globale ale producţiei, prin corelarea informaţională a tuturor activităţilor componente. Productica, ca metodă de conducere a producţiei, corelează parametri iniţiali ai procesului de proiectare, cu parametri celorlalte compartimente ale Sistemului Productiv.

Sistemul de Producţie realizează o conexiune informaţională strânsă, între toate compartimentele, care participă la realizarea obiectului tehnic. Scopul funcţionării integrate a tuturor disciplinelor tehnice şi organizatorice este Fabricaţia în regim de competiţie. Principalele conexiuni, realizate de metoda de conducere promovată de disciplina productică, sunt redate în Fig.1.2. Corelaţiile care privesc proiectarea sunt:



Din punct de vedere al procedurii, proiectarea constructivă este un proces algoritmic, în care se regăsesc sistematic aceleaşi faze, indiferent de domeniul tehnicii, căreia îi aparţine obiectul proiectat.

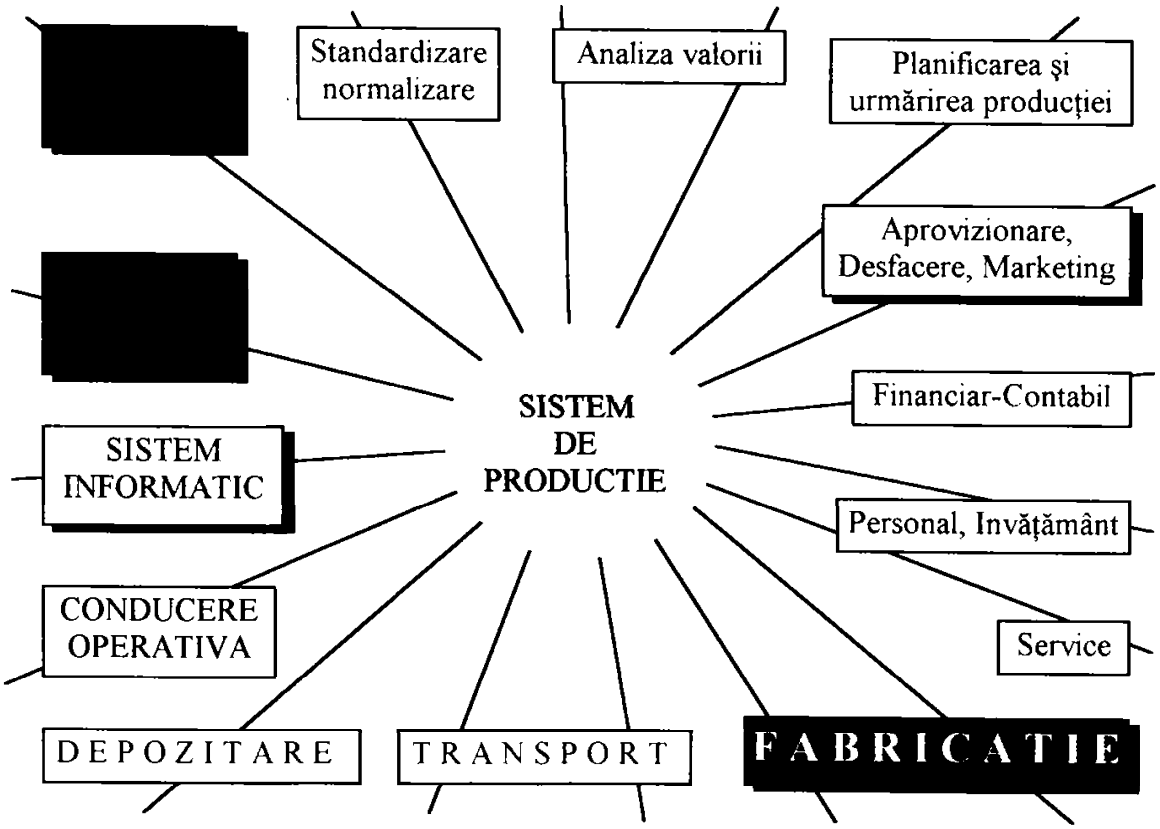


Fig.1.2

Din această perspectivă, proiectarea constructivă a maşinilor este un proces implementat într-un compartiment de proiectare, puternic corelat cu alte compartimente, şi în acelaşi timp, inundat de o mare cantitate de informaţii. Proiectarea poate, şi este constrânsă, să apeleze la baze de date cu

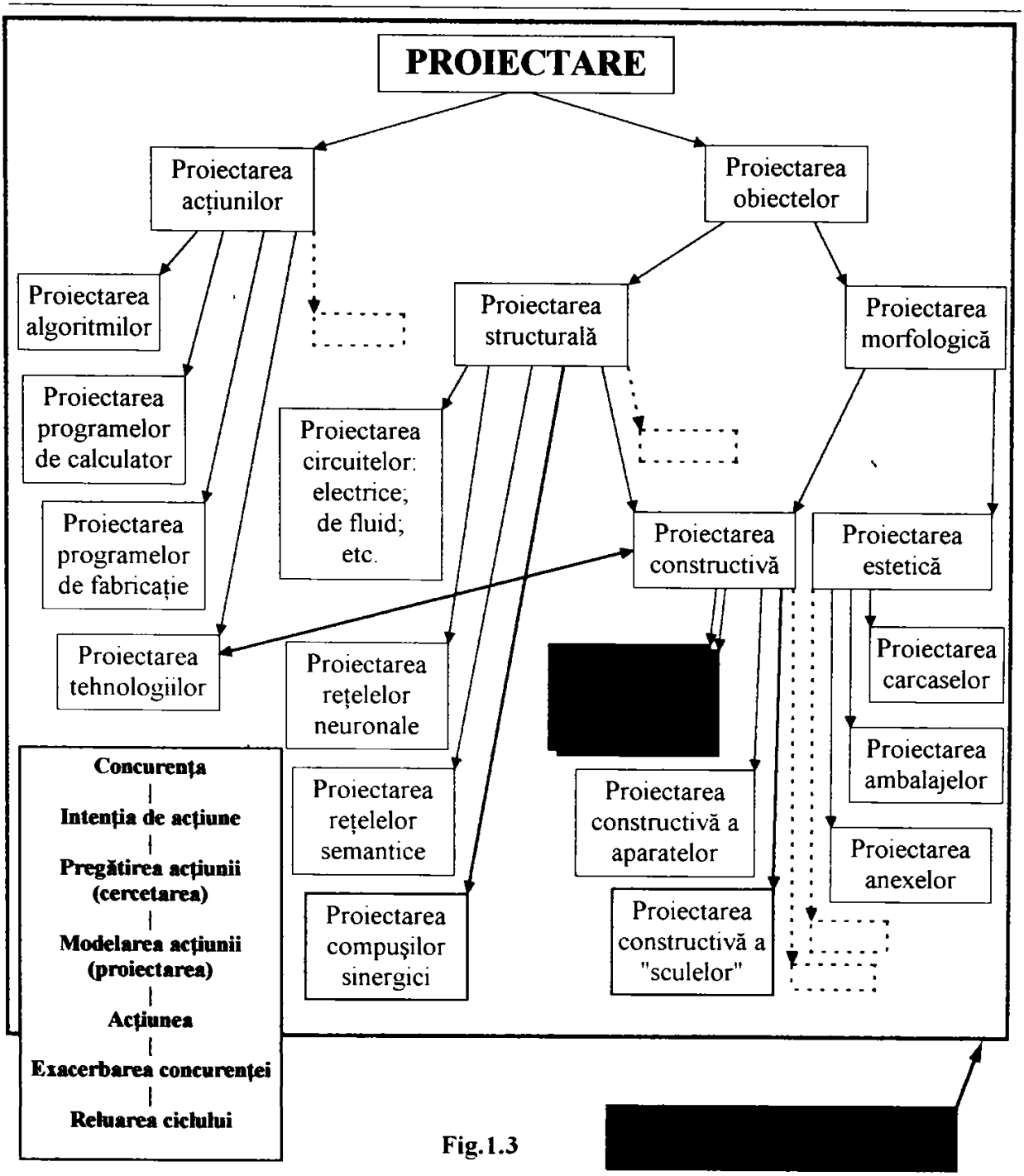


Fig.1.3

conţinut vast şi divers. Această posibilitate tehnică satisface, în primul rând, fazele incipiente ale procesului de proiectare, care sunt îndreptate spre informare.

Diversitatea conţinutului noţiunii de proiectare. Separând majoritatea activităţilor de planificare, după criteriul acţiune-obiect, am construit un mediu deschis al proiectării conceptuale, care are abilitatea de a deveni complet. Organizarea clasificării este făcută după obiectul proiectării şi este prezentată în Fig.1.3. Avantajul acestei organizări este caracterul sistemic. Această organizare permite completarea structurii cu activităţi, care ulterior devin activităţi de proiectare.

Schema din Fig.1.3, prin invocarea concurenţei, sugerează necesitatea conectării activităţii de proiectare la piaţa mondială. Concurenţa este un fenomen continuu pe piaţă, dar în activitatea unei întreprinderi acţionează ciclic, prin ciclul de fabricaţie propriu întreprinderii.

1.2. ALGORITMII PROIECTARII

Descoperirea detaliilor conexiunii proiectării cu alte activităţi inginereşti şi de altă specialitate, precum şi stabilirea căilor de comunicare, este facilitată de analiza algoritmului proiectării constructive al maşinilor. Algoritmul tradiţional de proiectare constructivă a maşinilor este prezentat simplificat în Fig.1.4, este detaliat în continuare şi cuprinde următoarele faze:

O. Analiza temei de proiectare

O.1. Sublinierea parametrilor dificil, respectiv imposibil de atins

O.2. Sublinierea restricţiilor dificil, respectiv imposibil de îndeplinit

O.3. Reformularea temei de proiectare:

A. Documentare asupra soluţiilor existente

A.1. Documentarea bibliografică prin consultare de:

- monografii;
- tratate de specialitate, manuale.
- articole de specialitate;
- reviste, referate, recenzii;
- standarde, norme;
- prospecte, cataloage de produse, reclame;
- brevete de invenţie;

Baze de date Internet

Transmisii prin curele

Organe de maşini; Vibraţii mecanice; Transmisii mecanice

Clasificarea transmisiilor mecanice cu element flexibil, etc.

ISO 9000 ISO 9001 ISO 9002 ISO 9003 ISO 9004

Optibelt Contitech Pirrelli

Patent nr. 12344.34.45/96. etc.

A.2. Informarea pe teren:

- participarea la târguri şi expoziţii;
- releveul;
- spionajul tehnic.

Dotare materială specializată

A.3. Sistematizarea informaţiilor:

- stabilirea criteriilor de clasificare a soluţiilor, efectuarea clasificării;
- elaborarea diagramelor ideilor;

A.4. Analiza inginerească a soluţiilor existente:

- stabilirea funcţiilor şi ordonarea acestora după metoda deciziei impuse;
- compararea soluţiilor şi stabilirea soluţiei optime existente;
- stabilirea lipsurilor generale ale întregii familii de soluţii existente;
- formularea temei de proiectare dezvoltată.

B. Elaborarea soluţiei tehnice

B.1. Creaţia originală:

- elaborarea soluţiilor tehnice originale;

Program de analiza combinatorică: Brainstorming, s.a.

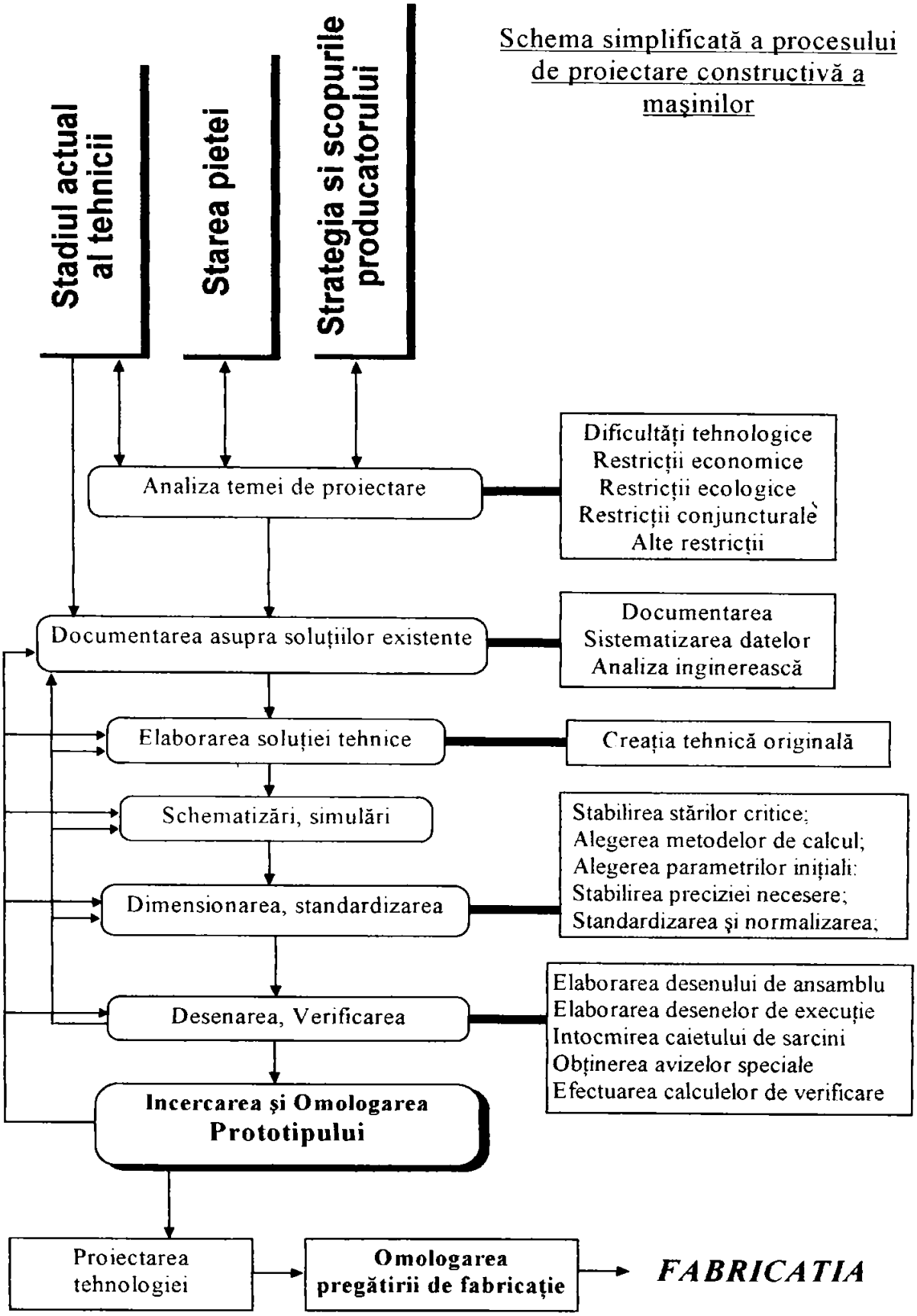

Schema simplificată a procesului de proiectare constructivă a maşinilor
Stadiul actual al tehnicii
Starea pietei
Strategia si scopurile producatorului
Analiza temei de proiectare
Dificultăţi tehnologice
Restricţii economice
Restricţii ecologice
Restricţii conjuncturale
Alte restricţii
Documentarea asupra soluţiilor existente
Documentarea
Sistematizarea datelor
Analiza inginerească
Elaborarea soluţiei tehnice
Creaţia tehnică originală
Schematizări, simulări
Stabilirea stărilor critice;
Alegerea metodelor de calcul;
Alegerea parametrilor iniţiali;
Stabilirea preciziei necesere;
Standardizarea şi normalizarea;
Dimensionarea, standardizarea
Desenarea, Verificarea
Elaborarea desenului de ansamblu
Elaborarea desenelor de execuţie
Intocmirea caietului de sarcini
Obţinerea avizelor speciale
Efectuarea calculelor de verificare
Incercarea şi Omologarea Prototipului
Proiectarea tehnologiei
Omologarea pregătirii de fabricaţie
FABRICATIA

Fig.1.4

- analiza inginerească a noilor soluţii; ← Analiza valorii, Programe de analiză.
- adoptarea soluţiei tehnice. ← Aprobarea temei de proiectare.

B.2. Cumpărarea de licenţă
B.3. Releveul

Baze de date; Internet; Relaţii internaţionale; Embargo; Resurse financiare

C. Schematizările, simulările
- C.1. Intocmirea schemelor de principiu
- C.2. Intocmirea schemelor de structură
- C.3. Intocmirea s h lor in matice
- C.5. Alte scheme

IDE pent u des n

D. Calculele de dimensionare ← Algoritmi de calcul; Programe procedurale de calcul
- D.1. Stabilirea stărilor critice
- D.2. Alegerea metodelor de calcul ← Programe de analiză structurală
- D.3. Alegerea parametrilor iniţiali:
  - alegerea materialelor; ← Programe expert de alegere a materialelor
  - stabilirea factorilor de sarcină, de mediu, etc.
- D.4. Dimensionarea: ← Programe Expert de analiză
  - stabilirea preciziei necesare;
  - standardizarea şi normalizarea; ← Baze de date de tip relaţional
  - analiza rezultatelor cu decizia de reluare a calculelor sau de stop.

E. Desenarea şi efectuarea verificărilor ← IDE (Integrated Develop Enviroment) pentru desenare
- E.1. Elaborarea desenului de ansamblu
- E.2. Efectuarea simultană a calculelor de verificare ← Programe de simulare şi verificare
- E.3. Obţinerea avizelor speciale (antiex., ecologice, etc.)
- E.4. Reluarea unor calcule de dimensionare
- E.5. Elaborarea desenelor de execuţie ← IDE pentru desenare
- E.6. Intocmirea caietului de sarcini ← Editoare de text
- E.7. Verificarea desenelor

F. Reluarea unor etape pentru creşterea performanţelor proiectului

G. Incercarea şi omologarea ← Programe specializate pentru controlul calităţii
- G.1. Măsurători efectuate asupra prototipului

G.2. Modificarea proiectului tehnic

G.3. Modificarea proiectului de execuţie

G.4. Modificarea caietului de sarcini

Intregul soft folosit la proiectare

H. Imbunătăţirea proiectării

Acest algoritm urmăreşte obţinerea unui proiect cu calităţi deosebite, şi în general, este urmat în cazul unui proiect cu mare grad de noutate. Din acest motiv este bogat în faze de informare **A1**, **A2** şi în faze de prelucrare şi interpretare a informaţiilor **A3**, **A4**.

In cazul unui proiect de rutină (de SDV de exemplu) conţinutul algoritmului este sensibil diferit. El poate fi următorul:

O. Analiza temei de proiectare

O.1. Sublinierea parametrilor dificil, respectiv imposibil de atins

O.2. Sublinierea restricţiilor dificil, respectiv imposibil de îndeplinit

O.3. Reformularea temei de proiectare

A. Documentarea asupra soluţiilor existente

Baze de date
Internet
Biblioteci

A.1. Documentarea bibliografică prin consultare de:

- arhivă proprie;
- standarde, norme;
- prospecte, cataloage de produse, reclame;
- manuale.

A.2. Informarea pe teren:

Registru de casă

- observarea funcţionării obiectelor similare.

A.3. Analiza inginerească a soluţiilor existente:

- compararea soluţiilor şi stabilirea soluţiei optime existente;
- stabilirea lipsurilor generale ale întregii familii de soluţii existente;
- formularea temei de proiectare dezvoltată.

B. Elaborarea soluţiei tehnice

B.1. Creaţia originală

- elaborarea soluţiilor tehnice originale;
- analiza inginerească a noilor soluţii;
- adoptarea soluţiei tehnice.

B.2. Cumpărarea de licenţă

B.3. Releveul

IDE specializat pe desenare

C. Schematizările, simulările

C.1. Intocmirea schemelor de principiu

C.2. Intocmirea schemelor de structură

C.3. Intocmirea schemelor cinematice

C.5. Alte scheme

D. Calculele de dimensionare ← Programe de calcul scrise în limbaje procedurale

D.1. Stabilirea stărilor critice

D.2. Alegerea metodelor de calcul

D.3. Alegerea parametrilor iniţiali: ← Programe expert pentru alegerea materialelor

- alegerea materialelor;

- stabilirea factorilor de sarcină, de mediu, etc.

D.4. Dimensionarea:

- stabilirea preciziei necesere;

- standardizarea şi normalizarea; ← Baze de date LAN (Local Arrav Network)

- analiza rezultatelor cu decizia de reluare a calculelor sau de stop.

E. Desenarea şi efectuarea verificărilor ← IDE specializate pe desenare

E.1. Elaborarea desenului de ansamblu

E.2. Efectuarea simultană a calculelor de verificare

E.3. Obţinerea aprobărilor speciale (antiex, ecologice, etc.)

E.4. Reluarea unor calcule de dimensionare

E.5. Elaborarea desenelor de execuţie

E.6. Intocmirea caietului de sarcini ← Editoare de texte

E.7. Verificarea desenelor

F. Reluarea unor etape, pentru creşterea performanţelor proiectului

G. Incercarea şi omologarea ← Intregul soft folosit la proiectare

G.1. Măsurători efectuate asupra prototipului

G.2. Modificarea proiectului tehnic

G.3. Modificarea proiectului de execuţie

G.4. Modificarea caietului de sarcini ← Soft specializat pentru controlul calităţii

H. Imbunătăţirea proiectării

1.3. PROIECTAREA ASISTATA DE CALCULATOR

Se înţelege prin proiectare asistată, acel proces de proiectare în care, un număr de operaţii sunt efectuate de către proiectant, cu ajutorul calculatorului electronic. Expresia a apărut odată cu folosirea calculatorului electronic în proiectare. Operaţiile efectuate de calculator au evoluat de la simple calcule iterative spre operaţiile de decizie, caracteristice Sistemelor Expert actuale. Gama temelor abordate de proiectarea asistată s-a extins, iar preocupările din acest domeniu, s-au integrat grupului de preocupări care se numeşte "Inteligenţa Artificială" (IA).

Metodologia de lucru cu IA funcţionează bine în anumite condiţii şi este adecvată unor anumite situaţii. De exemplu:

* problemele adresate sistemului bază de cunoştinţe sunt prost structurate şi implică cunoştinţe vaste sau de tipuri diferite.
* acea cunoaştere poate fi încorporată într-un algoritm eficient de căutare, care poate fi privit ca o configuraţie de blocuri standard, pentru algoritmii de căutare.
* algoritmul de căutare poate fi implementat ca un program eficient, folosind limbajele curente.

Definirea dimensiunilor problemei proiectării se face prin desfăşurarea lor după două dimensiuni:

- complexitatea problemei;
- cunoaşterea disponibilă.

Complexitatea problemei:

- complexitate pe verticală;
- complexitate pe orizontală.

Complexitatea pe verticală:

- numărul de nivele de abstractizare;
- posibilitatea de a lua decizii la nivel înalt, astfel încât, implementarea să se facă corect, cu respectarea condiţiilor de la nivelul fizic.

Complexitatea pe orizontală constă în complexitatea interacţiunilor dintre subprobleme:

- subprobleme independente;
- subprobleme în interacţiune puternică.

Prin cunoaştere disponibilă, se înţelege acea cunoaştere uşor accesibilă de la experţi sau din manuale.

Multe probleme de proiectare pot fi formulate ca procese de căutare într-un spaţiu al obiectelor-soluţie cu aceeaşi funcţionalitate. Pot exista mai multe feluri de obiecte tehnice obţinute prin tehnologii diferite şi care să aibă aceeaşi funcţionalitate. Procesul de proiectare este privit, în mod uzual, ca o căutare dirijată, într-un spaţiu multidimensional al soluţiilor posibile.

1.4. ANALIZA TIPOLOGICA A ASISTARII PROCESULUI DE PROIECTARE

Faza de analiză a temei de proiectare este extrem de rafinată şi necesită o bună cunoaştere a domeniului, o vastă experienţă şi o intuiţie dezvoltată. Luând în considerare cerinţele şi faptul că, această fază este determinantă în obţinerea succesului sau a eşecului, ea rămâne în continuare, apanajul exclusiv al proiectantului uman.

Faza **A**, de informare asupra soluţiilor existente, este deosebit de adecvată pentru a fi asistată de calculator. Cercetarea bibliografică **A1**, cu ajutorul bazelor de date, este deosebit de eficientă, atât ca

timp necesar, cât şi ca dimensiune a spaţiului cercetat. Sistematizarea informaţiilor şi analiza inginerească a soluţiilor **A3 şi A4** sunt etape care pot fi parţial asistate de calculator.

Faza **B**, de elaborare a soluţiei tehnice, este caracterizată printr-o activitate creativă şi ar putea fi asistată de calculator prin utilizarea de jocuri combinative implementate în Sisteme Expert. Decizia de alegere finală a soluţiei rămâne atribuţia proiectantului uman.

Faza **C**, de schematizări, este deja puternic asistată de calculator, prin programele de desenare dotate cu biblioteci adecvate AUTOCAD, ORCAD, etc.

Faza **D** conţine, atât etape de calcul, care sunt asistate de calculator, prin programele scrise în limbajele procedurale cunoscute, cât şi etape care se pretează la alegeri euristice, cum ar fi alegerea materialelor. Dimensionarea şi standardizarea pot face parte din programele de calcul.

Faza **E**, de desenare, este asistată la diferite nivele, foarte răspândite fiind atât mediul, cât şi conceptul "AUTOCAD". Calculele de verificare se efectuează, fie cu programe separate şi independente, fie cu programe care îşi extrag datele din mediul AUTOCAD. Tendinţa actuală este de a folosi pentru proiectarea de rutină Sisteme Expert, care să efectueze automat atât alegerile necesare, cât şi calculele şi desenele.

Faza **G**, de efectuare a probelor, este asistată de calculator, în măsura existenţei standurilor pe care măsurătorile, şi respectiv prelucrarea rezultatelor se efectuează cu calculatorul, în timp real sau "a posteiori".

Procesul de proiectare constructivă a maşinilor foloseşte programe algoritmice de calcul, baze de date şi Sisteme Expert. Intre ultimele două, se constată, în ultimul timp, o foarte puternică interferenţă, în sensul că, Sistemele Expert sunt capabile să exploreze baze de date, cu scopul de a lua decizii de nivel expert iar, pe de altă parte, sistemele de gestionare ale bazelor de date evoluate folosesc accesări neprocedurale. Se remarcă conceptul **SQL,** care are asociat un limbaj neprocedural, aşa cum sunt limbajele folosite la dezvoltarea Sistemelor Expert.

1.5. CONCLUZII

- Proiectarea este un proces multiplu conectat, din punct de vedere informaţional, la Sistemul de Producţie.
- Noţiunea de Proiectare are un conţinut larg de activităţi şi o circumscrie pe cea de Proiectare Constructivă a Maşinilor.
- Detalierea conexiunilor informatice ale proiectării constructive se poate face pe baza analizei algoritmului tradiţional al proiectării.
- Nivelul de asistare, cu ajutorul calculatorului, a fazelor procesului de proiectare constructivă rezultă din analiza tipologică a procesului de proiectare.
- Programele de calcul, bazele de date, mediile de desenare şi Sistemele Expert au devenit instrumente operative în Procesul de Proiectare.

2. INTELIGENTA ARTIFICIALA SI SISTEMUL EXPERT

(stadiul actual - continuare)

2.1. DEFINITII

Cuvântul *inteligență* provine din limba latină şi este preluat de majoritatea limbilor moderne, cu aceeaşi semantică. Câteva definiţii, redate mai jos, au menirea să contribuie la conturarea conţinutul noţiunii:

1. **Intelligentia** lat. - pricepere, înţelegere, cunoaştere.

2. **Inteligență** - este înţelegerea profundă, uşoară a unor lucruri, mai ales în domeniul culturii şi al ştiinţei; facultatea de a înţelege, de a pricepe fenomenele, lucrurile, etc. [1]

3. **Intelligence** fr. - este facultatea de a cunoaşte, de a înţelege; ansamblul funcţiilor mentale, având ca obiect, cunoaşterea conceptuală şi raţională (opusă cunoaşterii prin senzaţii şi intuiţii)

4. **Inteligență** - facultatea de a înţelege. [2].

5. **Inteligență** - Capacitatea de a înţelege uşor şi bine, de a sesiza ceea ce este esenţial, de a rezolva situaţii sau probleme noi, pe baza experienţei acumulate anterior; deşteptăciune. [3].

6. **Înţelegere** - Este activitatea gândirii prin care se descoperă legăturile dintre obiecte şi fenomene. In formă elementară, înţelegerea este cuprinsă chiar în procesul perceperii. In formă mai complexă, înţelegerea este implicată în descoperirea legăturilor de cauză şi efect, a semnificaţiei unei lucrări artistice sau ştiinţifice, a motivelor conduitei oamenilor, etc. Înţelegerea este implicată, mai ales, în procesul de rezolvare al problemelor. Înţelegerea se bazează pe experienţa trecută şi pe utilizarea acesteia, într-o situaţie nouă. [4]

7. **Inteligența Artificiala** - IA [5] este un domeniu de cercetare al cărui scop constă în, creearea sistemelor capabile să îndeplinească activităţi inteligente; studiul şi modelarea inteligenţei. Sistemele realizate nu trebuie să copieze în mod necesar, metodele şi tehnicile utilizate de om pentru îndeplinirea activităţilor, importantă fiind numai efectuarea ieftină, sigură şi eficientă a activităţilor propuse.

8. **Inteligența Artificială** - IA[6] cuprinde eforturile depuse pentru dotarea calculatoarelor cu capacităţi, care în mod obişnuit, sunt atributele inteligenţei umane: achiziţia de cunoştinţe, percepţia (vizuală, auditivă), raţionamentul, luarea deciziei, etc.

Aceasta constă în a reproduce comportamentul inteligent, fără a reproduce funcţionarea creierului uman. Inteligența Artificială apare ca o branşă avansată a informaticii, întrucât ea foloseşte tehnicile informaticii, asimilându-i progresiv experienţa. Este o disciplină pluridisciplinară.

9. **Inteligența Artificială** poate fi definită [10] ca o ştiinţă, care capacitează calculatoarele şi maşinile să înveţe, să gândească şi să emită judecăţi.

Tehnologia IA oferă instrumente, care ne ajută să efectuăm următoarele operaţii:

- captarea şi reţinerea experienţei acumulate de inginerie de-a lungul anilor;

- amplificarea expertizei, care este necesară, pentru a aborda aplicaţii şi metode noi;

- proiectarea sistemelor care operează inteligent şi în timp real, pentru asistarea personalului destinat să supravegheze funcţionarea instalaţiilor complexe.

2.2. STADIUL ACTUAL AL APLICATIILOR DE INTELIGENTA ARTIFICIALA

Două dintre preocupările fundamentale ale cercetătorilor din domeniul IA sunt: reprezentarea cunoştinţelor şi problema căutării. Prima, îşi pune problema cuprinderii întregului domeniu de cunoştinţe cerute de comportamentul inteligent, într-un limbaj formal; accesibil pentru calculator. Căutarea este o problemă de explorare sistematică a spaţiului cunoştinţelor, parcurgând succesiv şi alternativ mai multe nivele. Aceasta constituie acţiunea esenţiala a rezolvitorului de probleme. In continuare se prezintă cele mai importante domenii de aplicaţie ale Inteligenţei Artificiale [7].

2.2.1. Teoria jocurilor.

Multe jocuri folosesc un set bine definit de reguli, ceea ce face să fie uşor de generat un spaţiu de căutare şi să se elibereze cercetătorul de multe ambiguităţi şi complexe, inerente în cele mai multe probleme structurate. Masa de joc este uşor de reprezentat în calculator, fără să ceară nici un sistem complex de achiziţie al datelor.

Jocurile pot genera spaţii de căutare foarte extinse, atât de extinse, încât să pretindă tehnici performante de căutare, numite tehnici euristice. Ele reprezintă un vast domeniu de cercetare în IA. Programele de jocuri, în ciuda simplicităţii lor, îşi oferă parteneri ale căror mişcări sunt imprevizibile.

In 1981 programul american "Belle" a devenit campionul mondial de şah al calculatoarelor. El conţine un fişier cu 350.000 de poziţii. Pe baza rezultatelor obţinute, în 1983 i s-a conferit titlul de maestru al SUA.[8]

In Mai 1997 calculatorul "Deep Blue", capabil să proceseze miliarde de variante, l-a învins, în mod spectaculos, pe campionul mondial de şah Garri Casparov. Acest rezultat demonstrează, nu ascendentul maşinii asupra omului, ci măiestria cu care omul a reuşit să perfecţioneze maşina.

2.2.2. Raţionamentul automat şi demonstrarea teoremelor.

Demonstrarea automată a teoremelor este cea mai veche preocupare din domeniul IA. Multe probleme importante, precum, proiectarea şi verificarea circuitelor logice, verificarea corectitudinii programelor pentru calculator şi controlul sistemelor complexe, sunt adecvate acestui mod de abordare. Demonstrarea teoremelor s-a bucurat de un succes sporit, prin născocirea unei euristici puternice, care să se bazeze numai pe evaluarea sintactică a expresiilor logice, reducând complexitatea spaţiului de căutare, fără să recurgă la tehnicile folosite în acest scop de rezolvitorii umani. In cazul problemelor de complexitate extremă factorului uman îi rămâne sarcina să le descompună în sarcini mai mici şi să conceapă euristica, pentru căutarea spaţiului posibil al soluţiilor.

2.2.3. Sistemele Expert.

Sistemul Expert (SE) este o combinaţie între cunoaşterea teoretica şi o colecţie de reguli euristice de rezolvare, provenite din experienţa practică. SE sunt construite folosind cunoştinţele provenite de

la un expert uman (sau mai mulți), care cunoștințe sunt codificate, astfel încât, calculatorul să le poată aplica la rezolvarea problemelor similare.

Caracteristica majoră a SE constă în, legătura între cunoștințele unui expert uman și strategiile de rezolvare ale problemelor ce aparțin sistemului. Aceste cunoștințe sunt implementate în program de către un inginer specialist în IA. Apoi sistemul este adus la rafinament, prin rezolvarea de probleme și supunerea lui criticilor experților. Procesul este reluat până se atinge nivelul de performanță dorit.

S-au realizat SE pentru chimia organică, capabile să deducă formulele structurale ale substanțelor, pornind de la formulele de bază și de la analiza spectrografică. SE Mycin este unul dintre primele, care rezolvă problemele de raționament, pornind de la informații incerte și incomplete; acesta a fost realizat pentru aplicații în medicină. Alt Sistem Expert, Prospector, a fost dezvoltat pentru prospectări geologice. Domeniile cele mai bine acoperite cu SE sunt: medicina, educația, afacerile, proiectarea și cercetarea științifică.

Majoritatea SE au fost scrise pentru domenii de specialitate relativ restrânse. Acestea sunt, în general, bine studiate și au clar definte strategiile de rezolvare ale problemelor. SE sunt lipsite de robustețe și flexibilitate și sunt incapabile de judecăți profunde. SE sunt greu de verificat, ceea ce constituie o problemă critică în aplicații, cum sunt: controlul traficului aerian, conducerea centralelor nucleare, sistemele de arme. In ciuda acestor deficiente SE își dovedesc valoarea în numeroase aplicații importante.

2.2.4. Înțelegerea limbajului natural și modelarea semantică.

Un scop de lungă durată al IA este crearea de programe capabile să înțeleagă limbajul uman. Nu atât abilitatea de a înțelege limbajul, pare să fie esențială, cât mai ales succesele acesteia în automatizarea înțelegerii.

Înțelegerea limbajului natural implică mult mai mult decât analiza gramaticală: descompunerea propozițiilor în părți componente și căutarea cuvintelor în dicționar. Adevarata înțelegere se bazează pe cunoașterea extensivă a contextului despre discurs și a idiomurilor folosite în acest domeniu, precum și abilitatea de a aplica cunoștințele contextuale generale, la rezolvarea omisiunilor și ambiguităților, ce constituie manifestări normale în vorbirea umană.

2.2.5. Modelarea performanțelor umane.

Chiar dacă SE împrumută o mare parte dintre cunoștințe de la experții umani, în realitate nu încearcă să simuleze procesele mentale umane, aceasta pentru că, în aprecierea unui SE contează exclusiv, criteriile de performanță. De fapt, programele care adoptă metode de rezolvare, care nu provin din gândirea umană, sunt adesea mai performante decât cele proprii umane. Cu toate acestea, proiectarea sistemelor, care explicitează modul propriu uman de gândire, este un domeniu fertil de cercetare, atât în ce privește IA, cât și psihologia.

Metodologiile de rezolvare ale problemelor dezvoltate pe calculatoarele științifice au oferit psihologilor o nouă metaforă pentru explorarea minții umane. Mulți psihologi au adoptat limbajul și teoria științei calculatoarelor pentru formularea modelelor inteligenței umane.

2.2.6. Simularea activității gândirii [4].

Au fost elaborate mai multe programe de simulare, dintre care cel mai cunoscut este "General Problem Solver", elaborat în 1960. Autorii au încercat să încorporeze în el diferite componente, care sunt comune unor modele variate de prelucrare a informației, un număr variat de concepte, tactici,

strategii, abordări euristice etc., care se presupune că sunt caracteristice gândirii umane. Modelul nu procedează la o căutare exhaustivă, ci abordează o strategie euristică de selectare a variantelor şi alegerea alternativei celei mai promiţătoare. In acest mod, se reduce drastic timpul de căutare al soluţiei, fără să se garanteze că soluţia va fi găsită.

2.2.7. Programarea roboţilor.

Proiectarea roboţilor inteligenţi, îşi propune să creeze maşini capabile să interacţioneze cu lumea înconjurătoare să ia, în mod independent, decizii pentru îndeplinirea unor sarcini dificile. Funcţia unui robot inteligent este îndeplinirea unei sarcini complexe. Se defineşte ca fiind complexă, sarcina pentru atingerea căreia, se pune problema unei acţiuni optimizate şi restricţionate; în plus există un număr extrem de mare de posibilităţi de atingere a scopului.

Un robot care execută orbeşte o secvenţă de mişcări, care nu răspunde la schimbările de mediu, care nu-şi detectează şi corectează erorile, nu este un robot inteligent şi nu poate îndeplini sarcini complexe. Ori tocmai sarcinile complexe, rezultate din dificultatea atingerii obiectivelor, constituie motive temeinice pentru utilizarea roboţilor.

2.2.8. Limbaje şi medii pentru Inteligenţa Artificială.

Mediile de programare includ tehnici de cunoştinţe structurate, programări orientate pe obiecte şi facilităţi de lucru pentru SE. Limbajele de nivel înalt utilizate sunt: LISP, C++ şi PROLOG. In afară de cîteva instrumente şi tehnici noi, este îndoielnic că se vor mai putea creea multe alte sisteme semnificative de IA. Multe dintre tehnicile actuale au devenit standarde de programare software şi sunt strâns legate de teoria IA.

2.2.9. Maşina de învăţare.

Invăţarea a rămas o problemă dificilă pentru programele IA, în ciuda succesului lor, ca rezolvitori de probleme. Mai multe programe au fost scrise pentru a demonstra că învăţarea este un scop, posibil de atins. Cel mai impresionant este programul "Automatic Mathematician" (AM), proiectat să descopere noi legi matematice.

AM presupune teoreme noi, îşi modifică cunoştinţele curente de bază şi utilizând legi euristice, urmăreşte cele mai interesante dintre alternativele posibile. Succesul programelor de tip maşină de învăţare, presupune existenţa unui set de principii generale de învăţare, care să fie cuprinse în structura programelor şi să fie corelate cu abilitatea de a învăţa, specifică fiecărui domeniu. Deşi există o mare varietate de probleme abordate de IA, acestea au câteva trăsături comune:

- concentrarea asupra problemelor pentru a căror rezolvare nu există algoritmi. Pentru aceste probleme căutarea euristică este o tehnică de rezolvare în IA;
- problemele cu omisiuni sau incomplet definite, pentru care se folosesc formalisme compensatoare;
- aptitudinea de a capta şi manipula manifestările calitative semnificative ale situaţiilor, care devine prioritară faţă de punerea problemei în formă numerică;
- efortul de a face legătura între sensurile înţelesului semantic şi forma sintactică;
- răspunsurile nu pot fi niciodată exacte sau optimale, ci numai "suficiente";
- folosirea unei mari cantităţi de cunoştinţe specifice domeniului problemei;

- folosirea unui meta-nivel de cunoştinţe, cu efect de sofisticare al controlului strategiei de rezolvare.

Reprezentarea şi raţiunea sunt acum sub cercetare, iar cercetătorii în domeniul calculatoarelor doresc să le înţeleagă operaţional, sau chiar algoritmic. In acelaşi timp, motivaţii politice, economice şi etice ne obligă să ne asumăm răspunderea pentru consecinţele artificiilor noastre.

2.3. APARITIA CALCULATOARELOR ELECTRONICE.

Primul calculator electronic s-a numit **ENIAC "Electronic Numerical Integrator And Calculator"** şi a fost construit în 1946 de către **John Mauchley şi Presper Eckert** de la Univesity of Pennsylvania. Acesta avea în componenţă 19000 de tuburi şi era capabil de 5000 de operaţii pe secundă. Progamarea ENIAC-ului se efectua cu ajutorul a 6000 de comutatoare, amplasate pe plăci externe. Toată construcţia cântarea 30t şi ocupa 145 m^2.

In acelaşi an, al apariţiei primului calculator electronic, la Princeton University **John von Neumann** a introdus conceptul de program memorat. Până la apariţia microprocesorului, evoluţia calculatoarelor a fost exprimată prin succesiunea a trei, sau patru generaţii, care în esenţă, nu reflectă evoluţia calculatoarelor, ci a electronicii.

Generaţia 1-a, până în 1959, este constituită din calculatoare construite cu tuburi electronice. Acestea aveau volum mare, viteză de operare redusă, memorie de capacitate mică şi erau programate în limbaj cod-maşină.

Generaţia a 2-a cuprinde calculatoare realizate între anii 1959-1964. Acestea au fost construite cu tranzistoare şi ferite. In această perioadă a apărut informatica de gestiune; au apărut limbajele **FORTRAN şi COBOL**.

Generaţia a 3-a apare în 1964 şi cuprinde calculatoarele realizate cu circuite integrate. Generaţia a 4-a este formată din calculatoare cu circuite VLSI, iar pentru generaţia a 5-a s-a prevăzut o structură paralelă, diferită de structura "von Neumann". Această clasificare se referă la hardware şi poate exprima în cifre, saltul tehnic spectaculos al construcţiei calculatoarelor. In paralel, putem constata saltul, cel puţin la fel de spectaculos, al programării. Realizarea Inteligenţei Artificiale este performanţa software a sfârşitului de mileniu. In linii generale, evoluţia calculatoarelor din ultimele decenii poate fi arătată prin:

- apariţia **PC**-urilor (1982), care a transformat calculatorul într-un produs de larg consum;
- apariţia reţelelor de calculatoare, a softului distribuit, a bazelor de date distribuite;

apariţia calculatoarelor paralele.

2.4. SISTEME EXPERT. ENUNTURI SI DEFINITII

Un prim enunţ definitoriu este dat în [2] de către Ivan Bratko, un specialist în limbajul PROLOG. Ideile centrale ale acestui enunţ le regăsim în mai toate enunţurile din lucrările care abordează această temă.

Un sistem bază de cunoştinţe este un program, care rezolvă problemele dintr-un domeniu îngust de aplicaţie, asemeni unui expert uman.

Sistemul Expert (SE), pentru a putea rezolva un număr mare de probleme, trebuie să posede un volum adecvat de cunoştinţe în domeniu. Din acest motiv, se mai numeşte sistem bază de cunoştinţe. De altfel, fiecare sistem bază de cunoştinţe poate fi considerat un SE. Sistemul trebuie, într-o anume măsură, să fie capabil să explice utilizatorului, atât comportamentul, cât şi deciziile sale, la fel ca un expert uman. O asemenea atidudine explicativă este necesară, în special, în domenii incerte, cum este diagosticarea medicală, cu scopul de a creşte încrederea în sistem şi de a oferi posibilitatea utilizatorului să descopere posibilele scăpări în raţionamentul sistemului. In acest mod se creează o interacţiune prietenoasă între utilizator şi sistem.

O altă trăsătură, care adesea i se pretinde sistemului, este aceea de a fi capabil să acţioneze în condiţii de incertitudine şi date incomplete. Informaţiile despre problemă pot fi incomplete sau nereale, iar relaţiile în domeniul problemei, pot fi aproximative.

Pentru a construi un SE trebuie, în general, să se dezvolte următoarele funcţii:

- funcţia rezolvitorului de probleme, capabilă să folosească cunoştinţele specifice domeniului;
- funcţia de interacţiune cu utilizatorul, care include explicaţiile asupra intenţiilor şi deciziilor;
- funcţia sistemului, în timpul şi după procesul de rezolvare al problemei.

Fiecare dintre aceste funcţii poate să fie foarte complicată, dependent de domeniul de aplicare şi de cerinţele impuse.

Instumentele IA includ aşa numitele SE bază de cunoştinţe, care sunt adecvate rezolvării sarcinilor specializate. Elementele caracteristice ale SE după [10] sunt următoarele:

- reprezentarea simbolică a cunoaşterii (predicate logice, reţele semantice, cadre, obiecte, reguli de producţie, etc.);
- raţionamentul simbolic, care foloseşte reguli şi metode capabile să deducă, să examineze, să se determine pe sine, etc.;
- existenţa explicaţiilor grafice, atât pentru managerul de sistem, cât şi pentru utilizatorul obişnuit.

In IA, [10] pentru luarea deciziilor, conducerea şi automatizarea sistemelor, se aplică trei metodologii importante:

- raţionamentul cu incertitudine;
- raţionamentul calitativ;
- raţionamentul cu reţele neuronale.

Structura generală a unui SE, după [7] este arătată în Fig.2.1

Este convenabil să se dezvolte un SE în trei module principale:

- o bază de cunoştinţe.
- o maşină de inferenţă.
- o interfaţă cu utilizatorul.

Baza de cunoştinţe cuprinde cunoaşterea specifică domeniului şi constă din fapte, reguli şi alte metode care conduc la rezolvarea problemelor în domeniu.

Maşina de inferenţă este cea care activează cunoştinţele din baza de cunoştinţe.

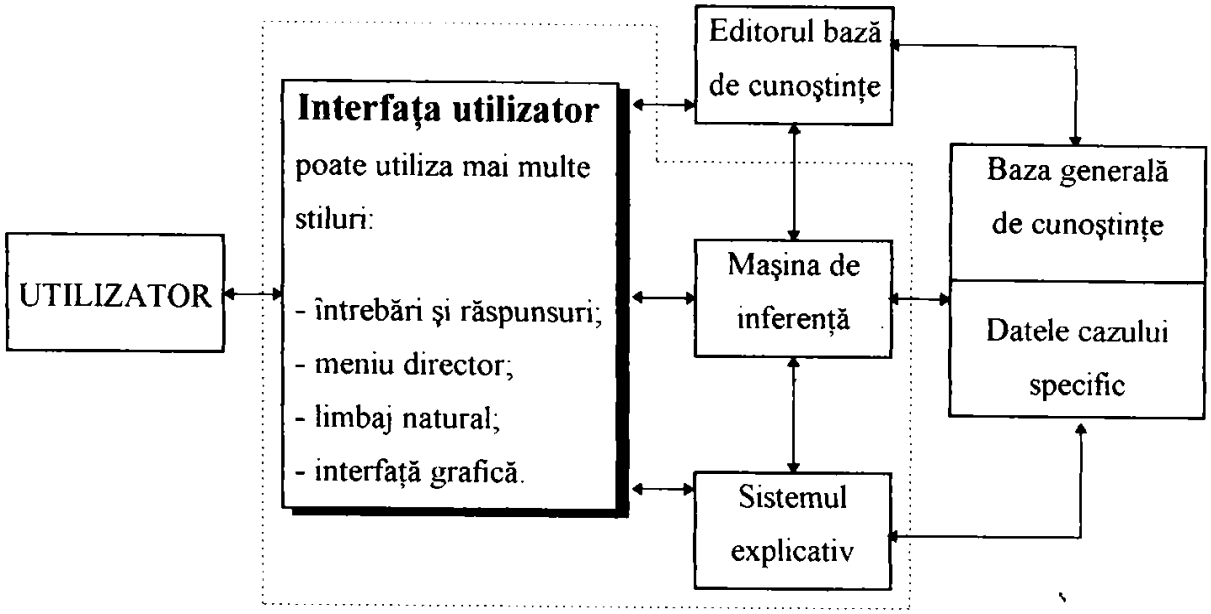


Fig.2.1.

Interfața utilizator realizează o legătură facilă, între sistem și utilizator, oferindu-i totdată acestuia, o imagine asupra procesului de rezolvare, realizat de mașina de inferență. Schema de alcătuire a SE separă cunoașterea de mașina de inferență. Această separare este necesară deoarece, cunoașterea este legată de domeniul de aplicație.

Dezvoltarea SE se face pentru mai multe aplicații, prin învățarea unui sistem Shell, cu folosință universală. Acestuia i se adaugă o nouă bază de cunoștințe pentru o nouă aplicație. Bineînțeles, toate bazele de cunoștințe vor respecta același formalism, care este recunoscut de sistem.

Planul de dezvoltare al unui SE [7] poate fi următorul:

1. Selectarea unui formalism pentru reprezentarea cunoașterii.
2. Proiectarea unei mașini de inferență, care să corespundă acestui formalism.
3. Adăugarea facilităților de interacțiune cu utilizatorul.
4. Adăugarea unei facilități de manevrare a incertitudinii.

In principiu, oricare formalism solid, în care putem exprima cunoașterea, poate fi folosit într-un SE. Cel mai cunoscut formalism este cel al silogismului "dacă-atunci". Folosirea regulii "dacă-atunci" adaugă următoarele caracteristici:

- **Modularitate;** fiecare regulă definește o parte relativ mică și independentă din cunoaștere.
- **Incrementabilitate;** regulile noi pot fi adăugate bazei de cunoștințe, relativ independent de alte reguli.
- **Flexibilitate;** regulile vechi pot fi schimbate relativ independent cu altele.
- Admite **transparența sistemului.**

Prin transpareanță se înțelege abilitatea sistemului de a explica deciziile și soluțiile. Regula adoptată facilitează răspunsul la următoarele întrebări fundamentale ale utilizatorului:

"CUM" întrebare care înseamnă: Cum ai ajuns la această concluzie?

"DE CE" întrebare care înseamnă: De ce te interesează această informaţie?

Un alt enunţ definitoriu este dat de George Luger în [7]. Un SE este un program bază de cunoştinţe, care oferă soluţii de calitate expert, într-un domeniu specific. In general, ştiinţa sa este preluată de la experţii umani, iar sistemul încearcă să le imite metodele şi mai ales performanţele. La fel ca oamenii pricepuţi, SE tind să devină specialişti, concentrându-se asupra unui set îngust de probleme.

La fel ca şi în cazul oamenilor, cunoştinţele SE, atât cele teoretice, cât şi cele practice, sunt perfecţionate pornind de la experienţa existentă în domeniu. Programele nu pot învăţa din propria lor experienţă, aceasta trebuie preluată din cea umană şi apoi codată în limbaj formal. Aceasta este sarcina majora a creatorilor de SE. Ele nu trebuie confundate cu programele de modelare a cunoaşterii, care se străduiesc să simuleze, în detaliu, arhitectura mentală umană. SE, nici nu copiază structura mentală umană, nici nu sunt mecanisme de inteligenţă generală.

Sistemele Expert sunt programe practice, care utilizează strategii euristice de căutare, dezvoltate pentru a rezolva clase speciale de probleme.

SE au fost creeate să rezolve o serie de probleme din domenii ca : medicină, matematică, inginerie, chimie, geologie, ştiinţa calculatoarelor, afaceri, drept, apărare şi educaţie. O lista restrânsă a categoriilor de probleme [7] rezolvabile de către SE este următoarea :

1. **Interpretare** - formularea concluziilor sau descrierilor de nivel înalt, despre colecţii de date native (primare).
2. **Predicţie** - predeterminarea consecinţelor probabile pentru situaţii date.
3. **Diagnoză** - stabilirea stării de funcţionare a unui sistem, prin măsurarea şi evaluarea unui set accesibil de parametri.
4. **Proiectare** - stabilirea configuraţiei unui sistem, care să atingă sigur performanţele cerute şi în acelaşi timp, să satisfacă un set de restricţii.
5. **Planificare** - planificarea unei secvenţe de acţiuni, care va completa un set de scopuri date, ca şi condiţii sigure de start.
6. **Supraveghere** - compararea comportamentului efectiv al unui sistem, cu cel prognozat.
7. **Depanare** - prescrierea şi implementarea remediilor, în caz de proastă funcţionare.
8. **Instruire** - detectarea şi corectarea deficienţelor de înţelegere a unui subiect, de către utilizatori.
9. **Control** - conducerea comportamentului mediilor complexe.

Trăsăturile SE conform [7] sunt :

- Deschise pentru inspecţie, atât în prezentarea paşilor intermediari, cât şi în ce priveşte răspunsurile la întrebările legate de procesul de rezolvare.
- Uşor de modificat, atât în ce priveşte adăugarea, cât şi ştergerea deprinderilor din baza de cunoştinţe.
- Obţinerea soluţiilor prin metode euristice.

Sunt mai multe motive pentru care este de dorit, ca un Sistem Expert, să fie deschis pentru inspecţie. In primul rând, dacă un expert uman, precum un medic sau un inginer, acceptă recomandarea făcută de un computer, el trebuie să fie convins că aceasta este corectă. Puţini experţi umani vor accepta sfaturi de la alţi confraţi, fără să înţeleagă raţionamentul, din care a rezultat sfatul. Această

necesitate provine din suspiciunea firească a utilizatorilor, iar explicaţiile trebuie să lege sfaturile oferite, de modul lor de înţelegere al domeniului.

In al doilea rând, când o soluţie este deschisă inspecţiei, se poate evalua fiecare aspect al deciziilor luate pe parcursul procesului de soluţionare al problemei, adăugându-se noi informaţii şi reguli, pentru creşterea performanţei sistemului. Aceasta joacă un rol esenţial în rafinarea bazei de cunoştinţe.

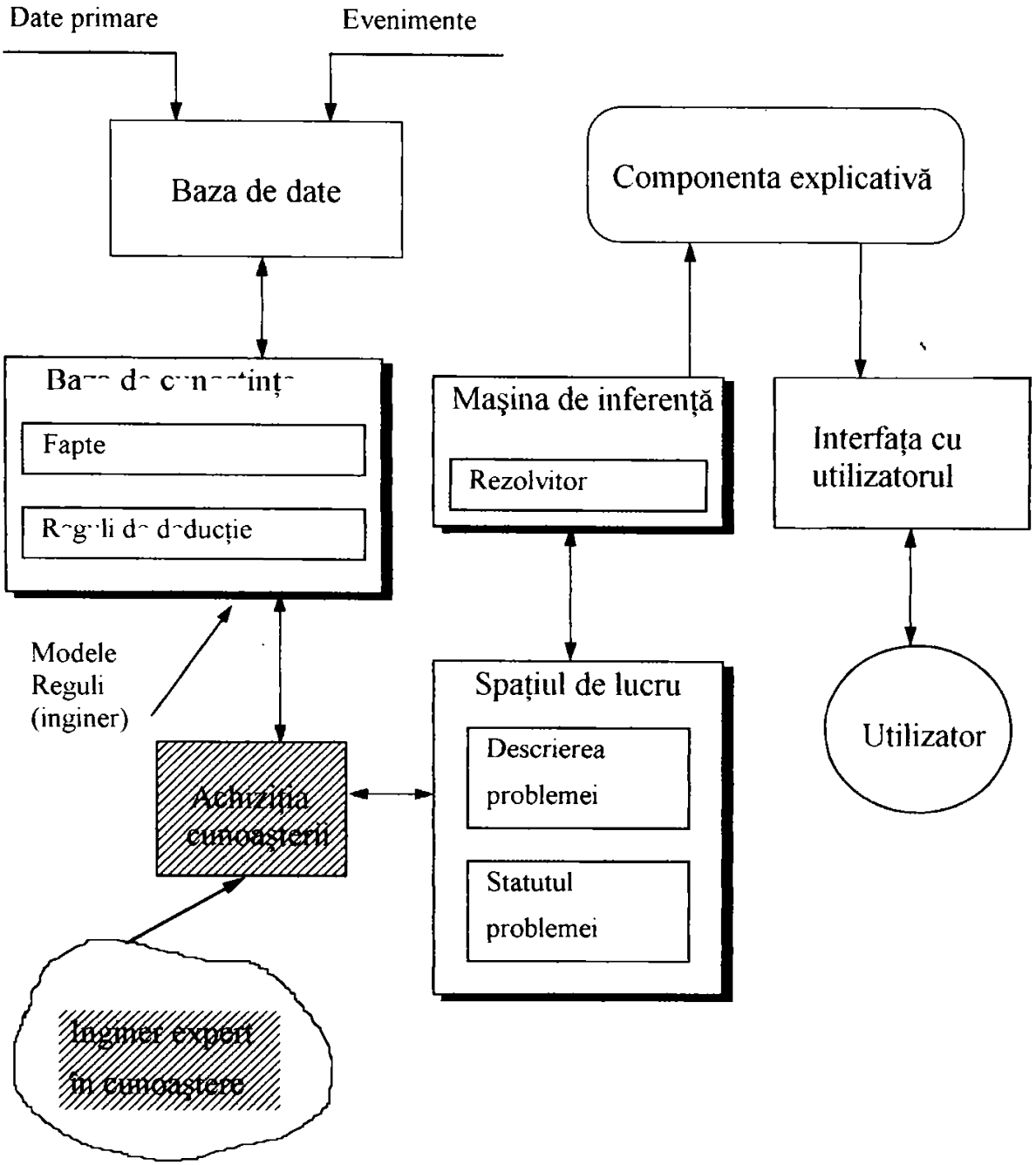


Fig.2.2.

A doua caracteristică pretinde ca sistemul să fie uşor de construit, testat şi schimbat, iar modificarea unei reguli să nu conducă la schimbări globale în sistem. Uşurinţa modificării bazei de cunoştinţe este un factor necesar realizării programelor de succes.

SE funcţionează pe baza regulilor euristice. De multe ori acestea contribuie la progresul cunoaşterii teoretice, iar alteori sunt simple scurtcircuitări, care nu sunt legate de teorie, dar care funcţionează în practică. Natura euristică a SE creează probleme în evaluarea performanţelor programului. Se ştie că metodele euristice au şi un procent de eşec. Probabil că evaluarea cea mai corectă constă în, compararea rezolvării date de SE, cu cea a unui expert uman în domeniu. Această metodă este sugerată de testul Turing[1].

O altă definire a structurii SE din [10] este redată în Fig.2.2. Elementele care o compun sunt următoarele:

- baza de cunoştinţe (cunoştinţe generale despre problemă, fapte şi reguli);
- baza de date (informaţii despre problema curentă);
- maşina de inferenţă;
- componenta explicativă (este dispozitivul, care poate informa utilizatorul despre modul în care s-au obţinut concluziile);
- interfaţa cu utilizatorul şi componenta de achiziţie a cunoaşterii;
- spaţiul de lucru (este o zonă de memorie în care se păstrează o descriere a problemei alcătuită din fapte furnizate de utilizator, sau deduse din baza de cunoştinţe).

2.5. CLASIFICAREA PE CATEGORII A SISTEMELOR EXPERT

Intenţia unui potenţial utilizator de a construi integral un SE nu poate să fie decât păguboasă, atât prin consumul de timp şi bani, cât şi prin calitatea soft-ului obţinut. Un drum raţional a fost deschis odată cu realizarea şi comercializarea SE Shell. Folosirea Shell-urilor presupune o bună cunoaştere a lor. Se prezintă în continuare câteva demersuri în acest sens.

O clasificare empirică a SE după aplicaţii tipice[11] este dată în tabelul T2.1. In condiţiile în care mediul de tip SE "Shell" se găseşte pe piaţă, într-o mare diversitate, alegerea tipului cel mai adecvat unei aplicaţii specifice, devine o decizie importantă. Această decizie poate fi abordată sistematic atunci când, aceste medii sunt evaluate şi clasificate. Alegerea unui Shell de către o organizaţie trebuie să ţină seama de următorii factori:

- caracteristicile aplicaţiei;
- caracteristicile proiectului;
- capacităţile SE Shell;
- sofisticarea utilizatorilor;
- uşurinţa de integrare cu soft-ul existent;
- susţinerea vânzării.

[1] Testul a fost propus de către matematicianul britanic Alan Turing, în anul 1950, cu scopul măsurării performanţelor maşinilor inteligente. Testul se bazează pe aprecierea subiectivă, de către un examinator, a răspunsurilor date de un subiect uman şi o maşină inteligentă. Maşina trece testul şi este considerată inteligentă atunci când examinatorul nu poate face distincţie între răspunsurile subiectului uman şi răspunsurile maşinii.

Categorie generică	Aplicaţii tip
1	Judecată inductivă
	Depanare
2	Analiză
	Proiectare
3	Analogie conceptuală
	Monitorizare
	Simulare
4	Configurare
	Invăţare
5	Conducere
	Planificare
6	Intermediere baze de date
	Interfaţă inteligentă
	Evaluare de situaţie
7	Diagnoză
	Selecţie
8	Diagnoză de eroare şi de defecţiune
	Instruire
	Interpretare
	Predicţie
9	Clasificare ierarhică
	Analogia modelelor
	Robotică şi vedere
	Validare
10	Deducţie aupra cunoaşterii directe
	Planificare

T2.1

Aplicaţiile SE, percepute ca distincte, sunt definite în tabelul T2.2. Acestea fiind prea numeroase pentru a putea fi manevrate la o anchetă, au fost grupate şi clasificate pe categorii generice. Categoriile rezultate sunt prezentate în tabelul T2.1.

Categoriile generice au uşurat efectuarea anchetei, privind importanţa care se acordă de către utilizatorii de SE Shell, diferitelor atribute şi capabilităţi ale acestora.

Grupul chestionat cuprinde programatori, ingineri de cunoaştere, proiectanţi de baze de date şi de SE, utilizatori de SE.

Notarea s-a efectuat astfel: fără importanţă - 1P; importanţă minoră - 2P; importanţă moderată - 3P; importanţă mare - 4P; importanţă critică - 5P.

Această anchetă stabileşte importanţa diferitelor criterii şi capabilităţi ale SE în funcţie de categoria aplicaţiei.

Rezultatele pot fi folosite la alegerea SE Shell ce urmează a fi achiziţionat.

2.6. EVALUAREA PRINCIPALELOR ATRIBUTE ALE SISTEMELOR EXPERT

Evaluarea Sistemelor Expert existente în exploatare s-a efectuat pe baza unui chestionar, care a fost distribuit specialiştilor şi utilizatorilor. Chestionarul a fost conceput astfel încât, să conţină cele mai semnificative criterii, factori şi surse, care se regăsesc în studiile de evaluare anterioare. Ancheta [11] a anticipat faptul că, mulţi dintre subiecţi nu vor fi capabili să răspundă, din cauza unei insuficiente experienţe în utilizarea SE.

S-au luat în considerare 87 criterii de evaluare şi o scară de evaluare de 5 nivele. Am extras numai atributele, care au reuşit să obţină punctaje medii, peste 3.9 şi le-am grupat în tabelele T2.3, T2.4, T2.5, T2.6 şi T2.7. In tabelul T2.8 am extras şi ordonat atributele cel mai favorabil punctate, pentru SE utilizate în proiectare (categoria a 2-a).

Se observă că, cele mai importante facilităţi, pentru această categorie, se referă la: legăturile cu limbajele de generaţia a 3-a şi a 4-a, legăturile cu bazele de date consacrate şi cu soft-ul special de proiectere.

Cel mai puţin cotat a fost atributul care se referă la adaptarea explicaţiilor. Aceasta nu înseamnă că facilitatea de explicare nu este apreciată, dimpotrivă ea este foarte bine punctată, şi deci, solicitată pentru SE din categoriile 9 şi 7.

T2.2.

Tip	Definiţie
Judecată inductivă	Sursă de ipoteze simple, plauzibile.
Analiză	Analizează structura unei entităţi date.
Formare conceptuală	Descoperă concepte, ordini sau relaţii.
Confruntarea conceptelor	Decide asupra conceptului cel mai adecvat situaţiei.
Configurare	Selectarea şi aranjarea componentelor.
Control	Interpretează, prezice, repară şi monitorizează comportamentul sistemelor.
Intermediar bază de date	Interfaţă pentru una sau mai multe baze de date. Interceptează toate cererile provenite de la utilizatori sau programe şi generează strategii inteligente de căutare.
Depanare	Acţionează sau emite recomandările necesare corectării situaţiilor nefavorabile.
Proiectare	Configurează un sistem cu restricţii, pe baza unui set de posibilităţi alternative.
Diagnoză medicală	Deduce disfuncţiile sistemice din observaţii.
Diagnoza defectelor şi a proastei funcţionări	Descoperă un defect, îl repară şi încearcă să descopere dacă nu mai sunt probleme remanente.
Clasificare ierarhică	Identifică locaţia unui obiect sau situaţie, în categoria de care aparţine, precum şi relaţia cu alte categorii.
Analogia ipotezelor	Determină când o aserţiune este conformă logic cu faptele cunoscute.
Instruire	Diagnostichează şi corectează comportamentul studenţilor.
Filtru inteligent	Interfaţă care îi ajută pe utilizatori să rezolve dificultăţile de utilizare hard, soft sau de procedură.
Interpretare	Pe baza unor date de măsură şi a unei informaţii auxiliare, selectează o ipoteză.
Deducţie direcţionată	Deduce ceva care nu provine neapărat din baza de date.
Învăţare	Acumulează cunoaştere nouă prin analogie, din exemple, prin relatare; leagă cunoaşterea nouă de ceea ce este deja cunoscut.
Monitorizare	Observă evoluţia sistemului şi alertează utilizatorul de îndepărtarea de la o situaţie previzibilă sau uzuală.
Deducţia limbajului natural	Acceptă solicitările utilizatorului, formulate în limbaj natural şi le transformă în limbaj formal.
Sinteză de obiect	Identifică şi evaluează pe baza unei descrieri structurale, componentele posibile ale unui obiect.
Organizare	Indeplineşte funcţia de conducere a proceselor
Echivalarea asemănărilor	Identifică obiectele aflate în relaţie.
Planificare	Activităţi de proiectare.
Predicţie	Deduce consecinţele probabile ale unor situaţii date.
Management	Conducerea sistemelor economice
Reparaţii	Execută planuri pentru reparaţii.
Roboţi şi vedere	Manipulează dispozitivele robotului; planifică mişcările; procesează informaţia video.
Programare	Intocmeşte un program orar pentru un grup de sarcini care pot fi îndeplinite cu resursele disponibile, fără să interfereze.
Procesor de imagine şi semnal	Transformă o imagine de intrare într-o altă imagine.
Simulare	Simulează procesele din sistemele mari.
Selecţie	Selecţia ca proces de căutare.
Evaluare de situaţie	Evaluează situaţia pe baza datelor de intrare şi a informaţiilor auxiliare.
Demonstrarea teoremelor	Aplică pe un operator, axiome sau concluzii anterioare, producând deducţii noi.
Validare	Verifică, dacă o acţiune sau un plan sunt temeinice şi suficiente, pentru a atinge un scop propus.

T2.3.

Interfaţa utilizator		1	2	3	4	5	6	7	8	9	10
Salvare evenimente	3.5	4.1	3.5	3.9	3.5	3.1	3.3	3.5	3.3	3.8	3.0
Facilităţi de explicare	4.0	4.0	3.9	3.9	4.0	3.5	3.9	4.3	4.0	4.5	4.3
Documentare condensată	4.0	3.9	4.0	3.7	4.4	3.8	4.0	4.1	4.0	3.8	4.1
Instruire	3.4	3.4	3.5	3.5	4.0	3.0	3.4	3.5	3.2	3.1	3.3
Meniuri; Ferestre; Culori	3.6	3.4	3.8	3.8	4.0	3.2	3.8	3.5	3.5	3.6	3.2
Caracteristici adaptabile	3.7	2.9	4.0	3.4	3.6	3.8	3.9	3.6	3.7	4.3	3.6
Acceptă 'nu ştiu' ca răspuns	3.5	3.6	3.2	3.6	4.0	3.2	3.5	3.5	3.5	3.5	3.5
Rezultate grafice	3.2	2.1	4.0	3.4	3.4	3.1	3.1	2.8	3.2	3.6	2.8

T2.4.

Interfaţa de dezvoltare		1	2	3	4	5	6	7	8	9	10
Editare; depanare; instruire	4.0	3.9	4.0	3.7	4.4	3.8	4.0	4.1	4.0	3.8	4.1
Tracing	3.9	4.1	4.0	4.0	3.9	3.9	3.9	4.0	3.8	3.6	4.2
Facilitate de explicare	4.0	4.0	3.9	3.9	4.0	3.5	3.9	4.3	4.0	4.5	4.3
Adaptare explicaţii	4.0	4.0	3.8	3.6	4.1	3.4	3.8	4.4	4.2	4.4	4.0
Trăsături adaptabile	3.7	2.9	4.0	3.4	3.6	3.8	3.9	3.6	3.7	4.3	3.6
Prototip rapid	4.0	4.2	4.2	3.7	4.2	3.8	4.1	4.2	3.8	4.4	3.9

T2.5.

Interfaţa sistem		1	2	3	4	5	6	7	8	9	10
Portabilitate	4.0	3.9	4.1	3.8	4.2	3.5	4.4	3.9	3.7	4.1	3.9
Suport pt. microcomputere	3.9	3.7	3.7	3.6	4.0	3.6	3.8	4.2	4.0	4.3	3.6
Compatibilitate cu sist.existente	3.6	2.7	3.8	4.2	3.5	3.1	4.2	3.6	3.2	3.5	3.2
Implantabilitate	4.0	4.1	4.1	4.0	3.9	3.8	4.3	3.8	3.9	4.7	4.3

T2.6.

Maşina de inferenţă		1	2	3	4	5	6	7	8	9	10
Căutare înainte	3.8	3.0	4.0	4.1	4.1	3.6	3.7	3.6	3.9	4.3	4.0
Căutare înapoi	4.0	4.6	4.0	3.6	3.7	4.1	4.1	4.2	4.0	4.4	3.8
Găsire a tuturor răspunsurilor	3.5	4.4	3.7	3.6	3.3	3.3	3.1	4.1	3.5	3.7	3.4

T2.7.

Baza de cunoştinţe		1	2	3	4	5	6	7	8	9	10
Reguli de producţie	4.0	4.3	4.1	2.6	4.2	4.0	4.0	4.3	3.7	4.1	4.0
Set structurat de reguli	3.7	3.8	3.9	3.0	3.8	4.2	3.9	3.7	3.4	4.0	3.8

Legătură cu limbajele 3g şi 4g	3.9	3.8	4.3	3.8	4.1	3.1	4.0	3.6	3.9	4.3	3.7
Legătură cu bazele de date	4.0	4.0	4.3	4.0	4.1	3.5	4.1	4.0	3.9	4.0	4.0
Acces la limbajul de bază	3.3	3.3	4.1	2.7	3.7	3.4	3.4	2.8	3.0	4.3	3.6
Legătură cu soft-ul special	3.5	2.6	4.3	3.6	3.8	2.5	3.7	3.1	3.8	3.8	3.2

2.7. COMPLEXITATEA SISTEMELOR EXPERT

Analiza conceptului se poate face mai uşor printr-o împărţire convenţională [12] în: complexitate de cunoaştere şi complexitate tehnologică. Conceptul determină un plan al complexităţii sistemelor, împărţit în patru zone:

1. Sisteme de productivitate personală;
2. Sisteme de decizie intensivă;
3. Sisteme tehnologice intensive;
4. Sisteme strategice.

T2.8.

Atributele principale ale SE de proiectare	Punctaj
Legătura cu limbaje 3g, 4g	4.3
Legătura cu baze de date	4.3
Legătura cu soft-ul special ad hoc	4.3
Prototip rapid	4.2
Portabilitate	4.1
Implantabilitate	4.1
Reguli de producţie	4.1
Acces la limbajul de bază	4.1
Documentare condensată	4.0
Editare; depanare; instruire	4.0
Căutare înapoi	4.0

Evaluarea complexităţii, porneşte de la stabilirea elementelor definite ca şi componente esenţiale.

Utilitatea înţelegerii complexităţii tehnologice şi de cunoaştere depăşeşte aplicaţiile cu cunoştinţe formale de bază, dezvoltate cu SE Shell sau cu alte limbaje.

Rezultă că, unele aplicaţii pot fi dezvoltate cu ajutorul SE Shell şi al limbajelor speciale, iar altele, mai sofisticate, necesită un soft specializat scris în C++, de exemplu.

Complexitatea de cunoaştere

Această latură a complexităţii se referă numai la caracteristicile "informatice" ale SE. Atributele luate în considerare la evaluarea complexităţii de cunoaştere sunt următoarele:

- dimensiunea domeniului de luare al deciziilor **P1**;
- profunzimea domeniilor de luare ale deciziilor **P2**;
- rata de schimbare a domeniului de luare al deciziilor **P3**;
- gradul de penetrare al sistemului în domeniu **P4**;
- numărul categoriilor generice de ieşire **P5**;
- dimensiunea intrărilor **P6**;
- interpretarea informaţiilor de intrare **P7**.

Pentru aprecierea sintetică a complexităţii de cunoaştere se propune următoarea expresie ponderată:

$$C_c = (P1 * 20 + P2 * 20 + P3 * 20 + P4 * 12 + P5 * 15 + P6 * 20 + P7 * 20) / 4,2$$

Punctajul propus pentru aprecierea **dimensiunii domeniului de luare al deciziilor** este funcție de numărul domeniilor la care se referă sistemul şi este redat în tabelul T2.9.

Descriere	P1
Un singur domeniu	1
Două domenii distincte	2
Mai mult decât două domenii distincte	3

T2.9.

La evaluarea **profunzimii domeniilor** incluse, prin combinarea nivelului de instruire avansată, cu durata experienței aplicate, pot fi descrise cinci stadii de achiziție a priceperii, de la novice, până la expert.

Achiziția cunoaşterii presupune existența, atât a teoriei cu reguli precise, cât şi a unei abilități intuitive, câştigate prin experiență. Punctajul este redat în tabelul T2.10.

Rata de schimbare a domeniului (-lor) de luare al deciziilor constituie un alt criteriu de complexitate. Unele domenii sunt relativ fixe, altele se schimbă ocazional, iar altele sunt într-o continuă evoluție. Ratele înalte ale schimbărilor sporesc complexitatea cunoaşterii SE. Punctarea acestui criteriu este redată în tabelul T2.11.

Gradul de penetrare al sistemului în domeniul luării deciziilor este o măsură a gradului de cuprindere completă, a acelui domeniu. Punctajul este redat în tabelul T2.12.

Profunzime	Descriere	P2
Comună	Pentru a fi expert, individul nu a avut nevoie pentru acumularea de cunoştințe de un grad avansat de cunoaştere a domeniului, sau de o experiență aplicativă substanțială	1
Adâncă	Pentru a fi expert, individului i s-a cerut un grad înalt de cunoaştere a domeniului şi mai puțin de 10 ani de experiență în domeniu, sau mai mult de 10 ani de experiență, fără cunoaştere avansată	2
Profundă	Pentru a fi expert, individului i s-a cerut, atât un grad înalt de cunoaştere, cât şi o experiență în domeniu mai îndelungată de 10 ani	3

T2.10.

Unele dintre sisteme determină lipsa unei informații importante pentru luarea deciziilor, altele pot efectua corelații de date, altele pot elabora decizii, iar sistemele de planificare ale întreprinderilor pot elabora strategii. O caracteristică de evaluare este **numărul categoriilor generice de ieşire.**

Tipurile specifice de ieşiri ale SE sunt următoarele:

- diagnoze ale problemei;
- acțiuni recomandate;
- soluții;
- testarea ipotezelor.

Rata schimbării	Descriere	P3
Joasă	Domeniul în care cineva poate fi expert timp de 5 ani consecutivi fără să dobândească nici o cunoștință nouă.	1
Moderată	Domeniul în care cineva poate fi expert timp de 2 ani consecutivi fără să dobândească nici o cunoștință nouă.	2
Inaltă	Domeniul în care o persoană poate fi expert, numai dacă își actualizează în permanență cunoștințele.	3

T2.11.

Grad de penetrare	P4
Mic	1
Parțial	2
Moderat	3
Substanțial	4
Complet	5

T2.12.

Punctajul corespunzător acestui criteriu de complexitate este redat în tabelul T2.13.

Un alt atribut de apreciere al complexității este **dimensiunea intrărilor,** exprimată prin punctajul redat în tabelul T2.14.

Interpretarea informațiilor de intrare constituie un ultim criteriu de complexitate luat în considerare la evaluarea complexității de cunoaștere. Nivelul de ambiguitate exprimat prin punctaj este redat în tabelul T2.15.

Ieșiri soluție	P5
Numai un tip	1
Oricare două tipuri	2
Oricare trei tipuri	3
Toate patru tipurile	4

T2.13.

Informațiile de intrare	P6
Una sau două intrări	1
Trei sau patru intrări	2
Patru sau mai multe	3

T2.14

Complexitatea tehnologică

Complexitatea tehnologică exprimă efortul făcut de creatorii SE, în scopul adaptării produselor lor, la un număr cât mai mare de mașini. Complexitatea tehnologică nu exprimă complexitatea și diversitatea hardware, ci consecințele acesteia, asupra produselor soft. Criteriile luate în considerare la evaluarea complexității tehnologice, sunt următoarele:

Nivelul de ambiguitate	Descriere	P7
Scăzut	Nu se cere o interpretare suplimentară	1
Moderat	Se cere o oarecare interpretare asupra intrărilor	2
Inalt	Se cere un înalt grad de interpretare al informației de intrare	3

T2.15.

- diversitatea hardware și a sistemului de operare **P8**;
- fluxul de informații **P9**;

- conectarea la reţea **P10**;
- măsura efortului de programare **P11**;
- diversitatea surselor de informaţie **P12**;
- nivelul de difuzie **P13**;
- nivelul de integrare în sistemele existente **P14**.

Aprecierea sintetică a complexităţii tehnologice se face cu relaţia următoare:

$$C_t = (P8 * 10 + P9 * 5 + P10 * 10 + P11 * 10 + P12 * 10 + P13 * 6 + P14 * 6) / 2,4$$

Diversitatea hardware şi a sistemelor de operare este o măsură a gradului de portabilitate, pe diverse arhitecturi de calculator şi pe diferite sisteme de operare.

Diversitatea hardware	P8
Un singur hard şi un singur sistem de operare	1
Două hard şi două sisteme de operare	2
Trei sau mai multe hard şi sisteme de operare	3

T2.16.

In acelaşi context tehnologic se includ şi instrumentele soft folosite la dezvoltatea SE, cum ar fi:

* Shell-uri şi limbaje pentru dezvoltarea bazelor de cunoştinţe;
* reţele neuronale;
* biblioteci grafice;
* sisteme de gestionare a bazelor de date;
* programe;
* vedere artificială.

Punctajul pentru acest criteriu este redat în tabelul T2.16.

Fluxul de informaţii exprimă faptul că, bazele de date foarte mari conduc la un efort de dezvoltare suplimentar, solicitând un efort de proiectare logică, o implementare a bazei de date şi un proces de tranzacţii. Punctajul propus pentru acest criteriu este redat în tabelul T2.17.

Fluxul de informaţii	P9
Mai puţin de 1 megabyte	1
Intre 1 şi 10 megabyte	2
Peste 10 megabyte	3

T2.17.

Conectarea la reţea exprimă faptul că, un sistem operează singular, fără să fie conectat la vreo reţea, se conectează sporadic, sau este conectat permanent la o reţea de calculatoare. Punctajul propus este prezentat în tabelul T2.18. Sistemele avansate de luare a deciziilor procedează la un schimb continuu de informaţii, cu reţeaua.

Conectare la reţea	P10
Sisteme singulare	1
Sisteme sporadic conectate	2
Sisteme permanent conectate	3

T2.18.

Măsura efortului de programare trebuie să exprime efortul depus pentru dezvoltarea sistemului; convenţional munca se măsoară în om-ore de activitate. Deoarece, această exprimare este dificilă, se propune măsurarea efortului prin numărul de reguli, în corelaţie cu dimensiunea bazei de cunoştinţe. S-a efectuat această corelare, deoarece în unele sisteme apar "obiecte", care echivalează la număr cu regulile simple de forma "dacă-atunci", dar sunt mult mai complicate, ca structură.

De asemenea, trebuie făcută diferenţierea între o bază de cunoştinţe propriu-zisă şi o bază de date asociată, care este consultată de către SE. Punctajul propus pentru efortul de programare se găseşte în tabelul T2.19.

Efortul de programare	P11
Mai puțin de 500 de reguli sau mai puțin de 500 Kilobytes	1
Intre 500 și 1500 de reguli sau între 500 Kbytes și 1,5 Megabytes	2
Peste 1500 de reguli sau peste 1,5 Megabytes	3

T2.19.

Diversitatea surselor de informare marchează constatarea că, o creștere a numărului de surse de informare conduce la o creștere a complexității, căci numărul surselor de informare determină numărul modulelor de acces. Punctajul propus pentru acest criteriu este redat în tabelul T2.20.

Diversitatea surselor de informare	P12
1 la 3 surse computerizate de informare	1
3 sau 4 surse computerizate de informare	2
5 sau mai multe surse computerizate de informare	3

T2.20.

Nivelul de difuzie al unui sistem informațional poate mări complexitatea tehnică a efortului de dezvoltare, prin porționarea datelor distribuite și prin cerințele procesului programat. Actualizarea bazei de cunoștințe a unui sistem, dispersat în mai multe departamente ale unei întreprinderi, sau în mai multe locuri, poate deveni un subiect major de activitate tehnică. Aprecierea valorică a acestui criteriu este redată în tabelul T2.21.

Nivel de difuzie	P13
Un singur utilizator	1
Doi sau trei utilizatori într-un singur departament	2
Trei sau mai mulți utilizatori într-un singur departament	3
Mulți utilizatori în mai multe departamente	4
Utilizare de nivel companie	5

T2.21.

Nivelul de integrare	P14
Fără legături	1
Minore	2
Medii	3
Majore	4
Extrem de largi	5

T2.22.

Nivelul de integrare în sistemele existente este un criteriu care marchează efortul de programare cerut integrării SE, cu sistemele informatice existente. Sistemul "XCON", realizat de "Digital Corporation Company" este capabil de legături cu sisteme de vînzare, cu ingineria de producție, cu producția. Toate aceste posibilități de conectare au necesitat un mare efort de programare. Aprecierea valorică se găsește în tabelul T2.22.

Concluzii referitoare la atributele cerute SE și la evaluarea SE. [13]

Cele mai importante atribute ale SE destinate proiectării sunt:

-existența legăturilor cu limbajele de generația a 3-a și a 4-a;

-legătura cu bazele de date consacrate;

-legătura cu soft-ul special de proiectare;

-capacitatea de a scoate rapid un prototip;

-portabilitatea, implantabilitatea;

-accesul la limbajul de bază.

2.8. RASPANDIREA SISTEMELOR EXPERT

Folosind această metodă de evaluare se obţine o imagine asupra distribuţiei SE în lume, după complexitate. Această distribuţie demonstrează efortul depus pentru dezvoltarea soft-ului şi distribuţia acestuia pe continente. Distribuţia este ilustrată în tabelul T2.23.

In Fig.2.3 este arătată o reprezentare a celor patru zone de evaluare a sistemelor, după cele două complexităţi. Reprezentarea conţine costul mediu necesar dezvoltării, precum şi durata de dezvoltare medie.

Distribuţia în %	Mapamond	Europa	America	Asia
Cunoaştere j tehnologie j	32	35	23	36
Cunoaştere i tehnologie j	18	23	19	14
Cunoaştere j tehnologie i	19	15	27	18
Cunoaştere i tehnologie i	31	27	31	32

T2.23.

- Din Fig.2.3 rezultă că, din punct de vedere al costurilor, este avantajoasă dezvoltarea sistemelor tehnologice (cadranul 3).
- Din T2.23 rezultă că, sistemele tehnologice (3) se bucură de interes numai în America.
- Sistemele de complexitate scăzută sunt agreate în Europa şi Asia.
- Sistemele foarte complexe sunt dezvoltate cu predilecţie în Asia şi America.
- Creşterea complexităţii sistemului, conduce la multiplicarea cheltuielilor de dezvoltare.
- Creşterea complexităţii sistemului, conduce la dublarea duratei de dezvoltare.

Distribuţia SE pe continente, funcţie de domeniul de aplicaţie, este redată procentual în tabelul T2.24 şi este ilustrată prin graficul din Fig.2.4. Examinând cifrele se trag următoarele concluzii:

- Fabricaţia este domeniul cel mai bine dotat cu SE;
- Cele mai multe SE care funcţionează în domeniul fabricaţiei se află în Japonia;
- Domeniul cel mai puţin dotat cu SE este cel al birocraţiei guvernamentale;
- Al doilea domeniu de interes pentru dezvoltarea SE este cel al finanţelor;
- Cele mai multe SE care funcţionează în domeniul finanţelor se află în Japonia;
- Al treilea domeniu de interes pentru dezvoltarea SE este cel al ingineriei;
- Cele mai multe SE care funcţionează în domeniul ingineriei se află în Europa;

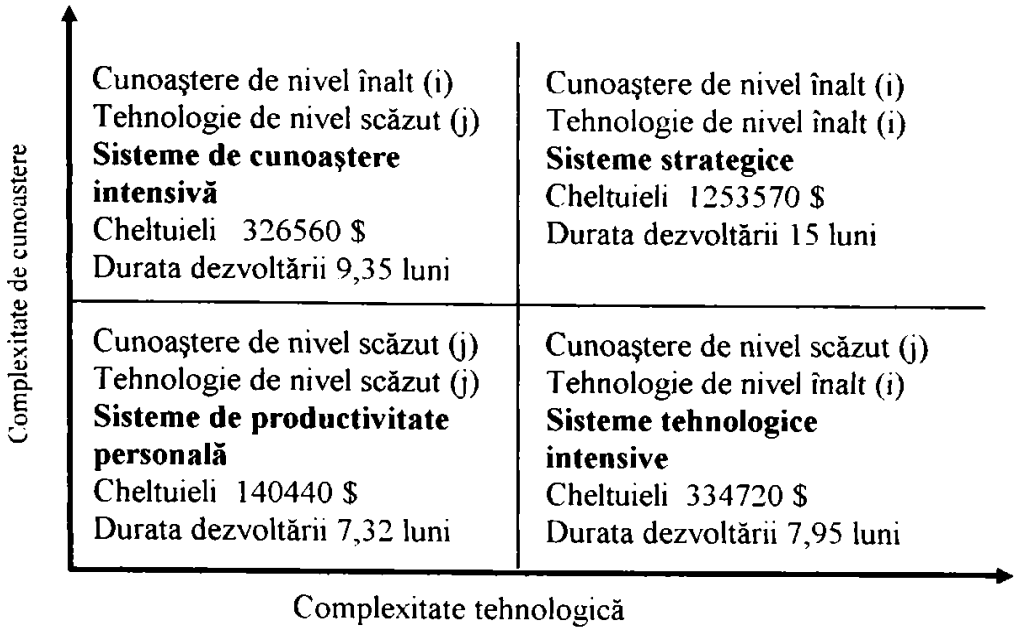


Fig.2.3.

Domeniul	Global	EUROPA	AMERICA	JAPONIA
Fabricație	**37.04**	**5.56**	**9.26**	**22.22**
Inginerie	**17.59**	**7.41**	**4.63**	**5.56**
Sănătate	**1.85**	**0**	**0.93**	**0.93**
Guvern	**0.93**	**0.93**	**0**	**0**
Transporturi	**7.41**	**6.48**	**0**	**0.93**
Finanțe	**22.22**	**1.85**	**8.33**	**12.04**
Energie	**4.63**	**0**	**0**	**4.63**
Construcții	**3.70**	**0**	**0**	**3.70**
Vânzări	**4.63**	**1.85**	**0.93**	**1.85**

T2.24.

- Cele mai multe SE în domeniul transporturilor sunt dezvoltate în Europa;
- America de Nord nu deține nici o prioritate numerică în dezvoltarea SE(se exceptă aplicațiile militare și răspândirea în masă a metodei).
- Specialiștii din Europa și Japonia s-au orientat cu predilecție spre dezvoltarea SE de complexitate scăzută, iar cei din America spre SE strategice, după cum se observă în graficul din Fig.2.5 și în tabelul T2.24.
- Cele mai multe SE strategice (economic) sunt dezvoltate în Japonia și Asia.
- Cele mai multe SE simple sunt dezvoltate în Japonia și Asia.
- Răspândirea SE, cu o complexitate mai mare decât media, este redată în Fig.2.6.

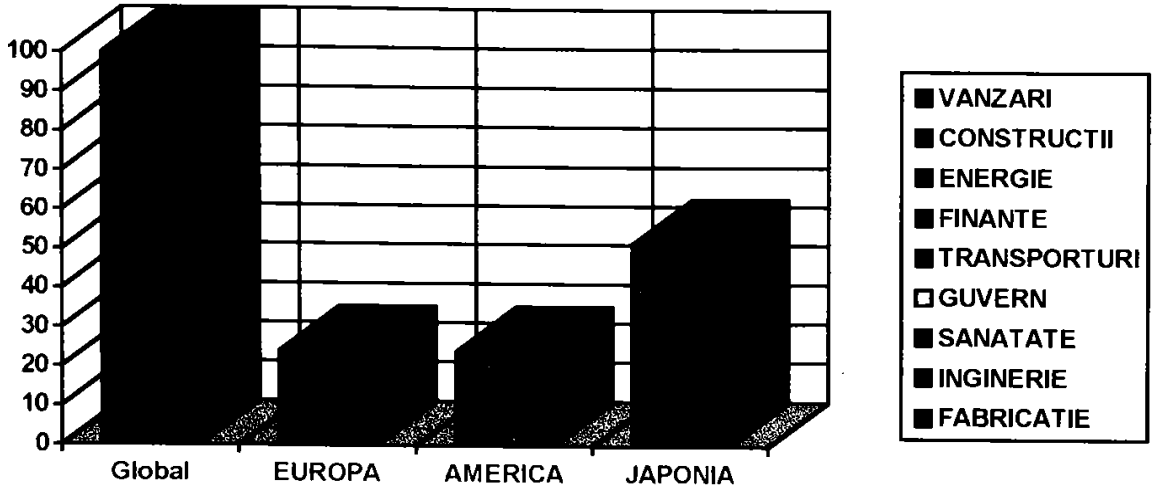


Fig.2.4.

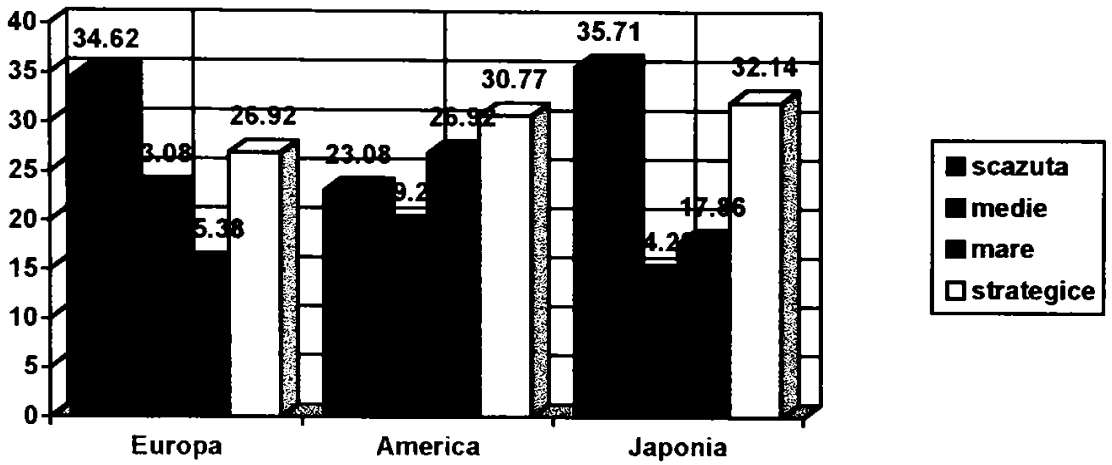


Fig.2.5.

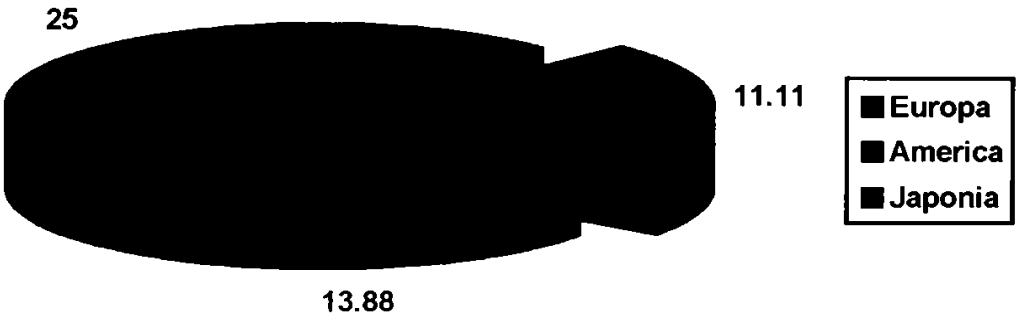


Fig.2.6.

- Domeniile abordate cu predilecție de către proiectanții şi utilizatorii din Europa sunt ingineria, transporturile şi fabricația.

2.9. CONDITIILE NECESARE DEZVOLTARII UNUI SISTEM EXPERT

Proiectarea, realizarea şi utilizarea unui Sistem Expert este posibilă atunci când, se îndeplinesc câteva condiții minimale, care sunt prezentate în continuare. Pentru a putea fi mai uşor percepute şi luate în considerare, ele au fost ordonate într-o clasificare ad hoc.

Condițiile necesare construirii unui Sistem Expert: - strategice;
- informatice;
- de specialitate;
- de procedură.

Condiții strategice: - definirea domeniului;
- definirea scopului;
- definirea funcțiilor sistemului;
- existența perspectivei progresului (progres = succes, câştig, putere);
- existența resurselor materiale.

Condiții informatice: - deținerea şi cunoaşterea soft-ului specific;
- cunoaşterea metodelor de reprezentare a cunoaşterii;
- existența metodelor de stabilire a coerenței bazei de cunoştințe.

Cunoştințe de specialitate: - cunoaşterea domeniului transmisiilor mecanice;
- existența posibilității definirii claselor;
- posibilitatea extragerii parametrilor necesari.

Parametri: - de performanță tehnică;
- de performanță economică şi comercială;
- ecologici şi sociali.

Parametri de performanță tehnică: - viteze; - masă; - puteri;
- randament; - gabarit; - fiabilitate.

Parametri de performanță economică: - preț de fabricație;
- consumuri specifice;
- cost de întreținere;
- aspect estetic, etc.

Parametri ecologici şi sociali: - emisia de noxe;
- nivelul de zgomot;
- pericolele de utilizare, etc.

Condiţii de procedură:

- stabilirea fazelor din procesul de proiectare, care sunt adecvate abordării în această manieră;
- stabilirea fazelor ce urmează să fie rezolvate, cu ajutorul Sistemului Expert;
- definirea restricţiilor de proiectare şi extragerea cunoştinţelor.

Surse pentru extragerea cunoştinţelor:

- din experienţa de proiectare;
- din experienţa de producţie;
- din experienţa de programare a calculatoarelor;
- din experienţa utilizării calculatoarelor;
- din experienţa matematică în domeniu;
- din experienţa economică a societăţilor comerciale.

2.10. DOTAREA UNEI BAZE DE DATE INTELIGENTE

Tehnicile şi limbajele, dezvoltate de activităţile din domeniul Sistemelor Expert, au atras atenţia proiectanţilor bazelor de date tradiţionale, asupra performanţelor de căutare şi deducţie pe care le-ar putea obţine prin folosirea acestora.

Arhitectura unui Sistem de Gestionare al Bazelor de Date Inteligente

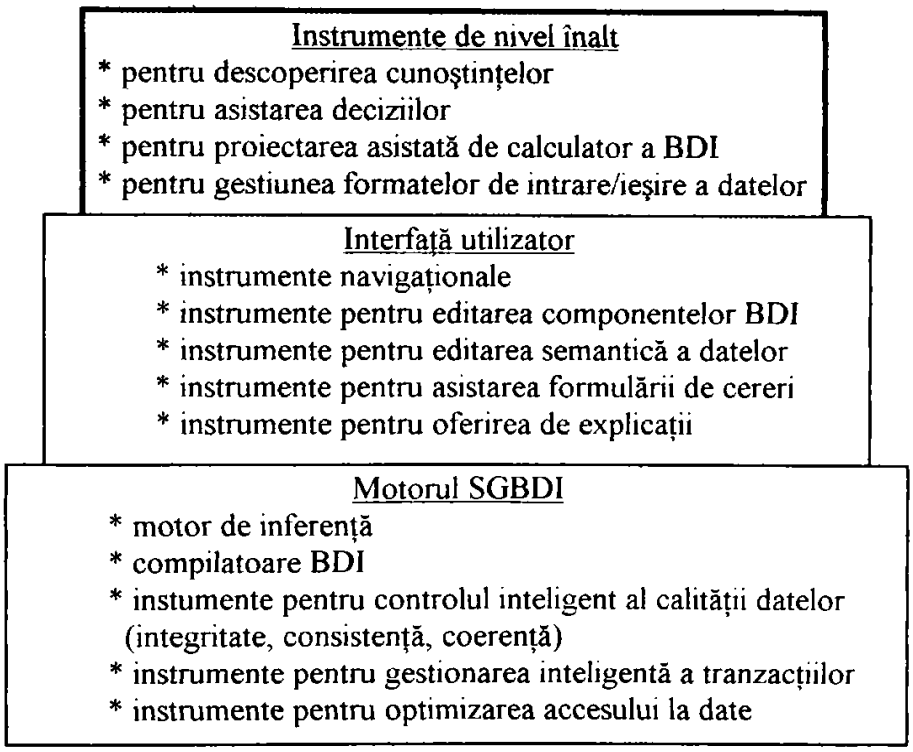


Fig. 2.7

Tehnica de interogare SQL şi cuplarea strânsă a Sistemelor de Gestionare a Bazelor de Date (SGBD), cu Sistemele Expert (SE) conduc la obţinerea unui Sistem Inteligent de Gestionare a Bazelor de Date (SIGBD). In [19] se prezintă alcătuirea necesară unui astfel de ansamblu. Bazele de Date Inteligente (BDI) sunt dotate cu un număr sporit de "instrumente" soft. Un exemplu, în sprijinul acestei afirmaţii, este prezentat în Fig.2.7.

Sistemele de Gestionare ale Bazelor de Date moderne, asemenea tuturor produselor soft, din ultimii ani, tind să înglobeze elemente de Inteligenţă Artificială. Acestea asigură o asistare inteligentă a proceselor de căutare, a editării cererilor şi explicaţiilor.

Maşina de inferenţă, inclusă într-un astfel de sistem, îi conferă calităţi de Sistem Expert, în domeniul prelucrării şi manipulării datelor. Aceste sisteme devin capabile să opereze cu date incomplete, să efectueze căutări sofisticate şi să realizeze tranzacţii inteligente. Si la acest nivel se constată convergenţa sistemelor spre o structură inteligentă standard.

3. TRANSMISII MECANICE. REPREZENTAREA SI ORDONAREA CUNOSTINTELOR

3.1. TEORIA REPREZENTARII CUNOASTERII

Cunoaşterea presupune o reflectare activă în conştiinţă a lumii reale, a esenţialului şi generalului din fenomene [15]. Astăzi, subiectul cunoscător al lumii dispune de un instrument suplimentar, calculatorul electronic, care mai mult decât hârtia şi creionul, îl poate ajuta la stocarea şi procesarea informaţiilor despre obiecte, fapte, fenomene.

Cunoaşterea este un tip special de activitate efectuată de fiinţa umană, ca agent sau subiect cunoscător. Cunoaşterea este un tip special de interacţiune între individul uman, ca subiect cunoscător şi realitatea fizică sau socială existentă, independentă de el [33].

Prelucrarea informaţiilor cu ajutorul calculatorului, odată în plus, este o activitate riguroasă, care are scop şi metodă, adică cele două atribute esenţiale ale ştiinţei. Astfel, tratarea sistemică a informaţiilor le ridică din starea de elemente de mulţime (relaţia de apartenenţă), în postura de elemente de structură şi le conferă acestora proprietăţi structurale (relaţii compoziţionale, cauzale, modale, epistemice), precum şi o denumire nouă, aceea de *date*.

Cunoaşterea poate fi clasificată în mai multe moduri astfel: **cunoaşterea comună,** fără metodă şi **cunoaşterea ştiinţifică,** cu metodă; **cunoaşterea empirică,** dacă informaţiile despre obiecte, fenomene, procese provin de la organele senzoriale sau de la instrumentele de măsură şi **cunoaşterea teoretică,** dacă ea provine dintr-un raţionament care operează cu legi cauzale, etc. Cunoaşterea teoretică se dezvoltă din cunoaşterea empirică prin analiză, sinteză, deducţie, particularizare, etc.

Finalitatea actului de cunoaştere ştiinţifică este **teoria ştiinţifică.** Cuvântul *teorie* vine de la grecescul *theoria*, care avea înţelesul de contemplare, meditaţie. Acest termen are astăzi, în filosofia ştiinţei, conţinut divers. Teoria este:

- o mulţime de *reguli şi principii de procedură;*
- o *schemă de terminologie şi clasificare*;
- un *sistem de concepte*;
- un *mod de descriere*;
- un *sistem de propoziţii*; formulate asupra unor entităţi neobservabile;
- un *sistem ipotetico-deductiv*.

d0. Teoria desemnează un sistem de propoziţii, logic organizat, care sintetizează o anumită cantitate de informaţii, referitoare la un domeniu al realităţii, pe care îl descrie şi explică [33].

Cunoaşterea este întotdeauna parţială şi incompletă. Esenţa cunoaşterii aparţine "obiectului", prin urmare ne este inaccesibilă [36]. Transpunerea activităţii de cunoaştere, pe suportul numit "calculator electronic", este o problemă a "inteligenţei artificiale".

Omul nu crează o altă inteligenţă, diferită de a sa, ci îşi foloseşte propria inteligenţă pentru a face calculatorul să aibă un astfel de "comportament" încât, să poată fi perceput ca inteligent. De aici rezultă şi o definiţie demitizatoare a Inteligenţei Artificiale, potrivit căreia:

d1. Inteligenţa Artificială este abilitatea omului de a instrui o maşină, astfel încât, în anumite împrejurări particulare, maşina să se comporte, prin reacţiile la stimulii externi, ca o entitate inteligentă.

Maşina inteligentă trebuie să-şi reprezinte realitatea prin simboluri şi să opereze cu acestea. Metoda de reprezentare presupune o anumită ordine conceptuală, ce cuprinde:

- sistemul de meta-reprezentare;
- sistemul de clasificare;
- sistemul de organizare.

Problema fundamentală a Inteligenţei Artificiale este cea de definire a unor metode pentru reprezentarea unei mari cantităţi de cunoştinţe, metodă care să permită stocarea şi utilizarea eficientă a acesteia. Metodele de reprezentare ale cunoaşterii pot fi grupate în:

- metode logice;
- metode relaţionale;
- metode procedurale.

d2. Metodele logice privesc cunoaşterea ca o serie de aserţiuni privind cunoştinţele şi relaţiile dintre acestea.

Metodele logice au avantajul aplicării directe a regulilor de inferenţă, asupra pieselor de cunoaştere. Dezavantajul metodelor logice constă în, existenţa soluţiilor nesatisfăcătoare de sistematizare a bazei de cunoştinţe, în dificultatea reprezentării cunoaşterii despre acţiuni şi a regulilor euristice.

d3. Metodele relaţionale sunt acelea prin care cunoaşterea este reprezentată, pornind de la relaţiile dintre obiecte, sub formă de grafuri şi reţele.

Metodele, din această categorie, permit organizarea cunoştinţelor funcţie de omogenitatea acestora, rezultând clase şi sorturi.

d4. Metodele procedurale sunt acelea în care cunoaşterea este reprezentată sub formă de proceduri, care permit obţinerea stărilor momentane, pornind de la stările iniţiale sau intermediare.

3.1.1. Reprezentarea logică

Formalizarea problemelor

Orice problemă formulată corect şi complet poate fi rezolvată. Pentru ca rezolvarea să se facă cu ajutorul Inteligenţei Artificiale, este necesară translatarea enunţului din limbaj natural, în limbaj formal.

d5. Limbajele formale sunt acele limbaje care au la bază logica formală, respectiv calculul propoziţional şi calculul predicatelor.

Orice strategie de rezolvare a unei probleme necesită, în primul rând, o reprezentare a acesteia sub o formă adecvată intrumentelor folosite.

Programarea logică în Inteligența Artificială comportă un formalism calculatoriu, având la bază trei elemente constitutive, ce formeaza aşa numitul sistem de producţii şi anume:

- regulile de producție sau combinaționale;
- o bază globală de date;
- un sistem de control.

O caracteristică a oricarui sistem de Inteligență Artificială este posibilitatea de **acces aleator** la baza de date. Regulile nu sunt apeluri de tip subrutină, ele acționează strict în cadrul bazei sistemului, iar adaugarea de fapte la conținutul bazei se poate face în cadrul acțiunii regulilor, prin aşa numitele "aserțiuni de fapte obținute prin instațieri" ale regulilor.[22]

Elementele de bază ale programării logice sunt determinate natural printr-o singură structură de date, "termenul logic" şi prin trei tipuri de instrucțiuni fundamentale: fapte, reguli şi întrebări. Cea mai simplă instrucțiune sau afirmație este "faptul", care nu are altă misiune decât de a arăta că anumite obiecte sunt legate între ele printr-o relație. Orice relație predicativă complexă are ca suport inițial una sau mai multe fapte, din acest motiv, putem substitui faptele cu condițiile inițiale. O înşiruire de fapte este ea însăşi un program logic, cel mai simplu [23].

In afară de capacitatea de a deduce, inteligența presupune, chiar înainte de aptitudinea deductiv-calculatorie, abilitatea de a reprezenta intern, în imagini subiective sau configurații lexicale, relații şi structuri din mediul extern.

d6. Reprezentarea realității este o codificare a acesteia, în sisteme de ecuații sau structuri semiotice[24]. Acțiunea de reprezentare trebuie să preceadă orice demers deductiv.

Principiile logicii

Problema centrală a logicii [9] este validitatea. Operațiile logice sunt valide, dacă satisfac anumite cerințe.

d7. Majoritatea cerințelor logice derivă una din alta, dar există câteva de maximă generalitate, care sunt primordiale şi se numesc **principii logice**.

Principiile logicii au fost identificate şi formulate de către Aristotel şi Leibnitz.

d8. **Principiul identității** afirmă că fiecare lucru este ceea ce este [35].

Prin aceasta se înțelege că, există un grup de proprietăți constante, care îi conferă calitatea de a fi ceea ce este, chiar dacă celelalte proprietăți suferă schimbări. Acest principiu pretinde termenilor să-şi păstreze înțelesul. Exigența impusă de acest principiu este **precizia**. Incălcarea acestui principiu este frecventă în limbajul natural.

d9. **Principiul necontradicției** afirmă că, este imposibil ca două propoziții A şi non-A să fie, în acelaşi timp şi sub acelaşi raport, adevărate.

Principiul necontradicției pretinde, să nu se admită în cadrul aceluiaşi demers rațional, două propoziții contradictorii. Exigența impusă de acest principiu este **consistența**. Necontradicția este un principiu logic şi nu ontologic; el exprimă faptul că gândirea nu poate reflecta o realitate contradictorie, decât în mod necontradictoriu.

d10. **Principiul terțiului exclus** stipulează că o propoziție este sau adevărată sau falsă.

Acesta este numit şi pricipiul bivalenței; el nu are universalitatea primelor două şi este acceptat numai de logica bivalentă.

d11. **Principiul raţiunii suficiente** afirmă că, nici un fapt nu poate fi adevărat sau real, nici o propoziţie veridică, fără să existe un temei, o raţiune suficientă pentru care lucrurile sunt aşa şi nu altfel, deşi temeiurile acestea, de cele mai multe ori, nu ne pot fi cunoscute (Leibnitz).

Calculul propoziţional

Propoziţia este forma de bază a gândirii omeneşti. Orice proces de gândire se exprimă printr-o judecată, care face un bilanţ provizoriu al unei stări de fapt. Forma primordială a gândirii este propoziţia sau judecata.. In sens gramatical, distingem propoziţii enunţiative, interogative, optative şi imperative. Indiferent de forma gramaticală, pentru logică este important conţinutul, adică gândirea exprimată de o propoziţie. Dintre propoziţiile gramaticale, logica foloseşte propoziţiile afirmative şi enunţiative.

d12. **Propoziţia** este acel produs ale gândirii, exprimat prin forme lingvistice, care are proprietatea de a deţine o valoare de adevăr.

d13. Adevărul ataşat propoziţiei este un **adevăr logic** şi are două valori: *adevarat* sau *fals.*

Definiţia propoziţiei necesită a fi completată cu definiţia adevărului, în sens logic.

d14. Acea parte din logică, în care se folosesc componentele propoziţiilor sub forma noţiunilor de subiect şi predicat, se numeşte **logica predicatelor.**

Sistematic, logica propoziţiilor precede logica predicatelor. Istoric, logica predicatelor a fost creeată de Aristotel, iar logica propoziţiilor a fost creeată de filosofii stoici. Cronologia apariţiei nu contrazice afirmaţia că, logica propoziţiilor este baza elementară pe care se construieşte logica predicatelor, căci Aristotel a folosit legile fundamentale ale logicii propoziţiilor fără să le exprime explicit. Logica predicatelor introduce **noţiunea** şi operatorii: **existenţial** şi **universal**.[25]

d15. Noţiunea este o formă logică fundamentală, care reflectă însuşirile esenţiale, necesare şi generale ale unei clase de obiecte.

Orice noţiune se compune din *conţinut şi sferă.* Noţiunile au diferite grade de generalitale, de la *noţiunea infimă,* până la *genul suprem* - categoria. In ce priveşte relaţiile dintre noţiuni, acestea pot fi: *identice, ordonate, contrare, contradictorii.* Operaţiile cu noţiuni sunt *definiţia, diviziunea şi clasificarea.*

Simboluri şi propoziţii.

Primul pas în descrierea unui limbaj îl constituie definirea elementelor sale constitutive şi anume, simbolurile.

d16. Simbolurile pot fi caractere sau seturi de caractere, precum şi semne grafice cărora li se atribuie câte o semnificaţie.

Propoziţiile, în această lucrare, sunt desemnate de literele mari de la sfârşitul alfabetului Englez. Simbolurile calculului propoziţional se numesc simboluri propoziţionale. [7] Exemple:

- P, Q, R, S, T, (propoziţii)
- true, false (simboluri de adevăr)
- $\wedge$ (conjuncţia)
- $\vee$ (disjuncţia)
- $\neg$ (negaţia)
- $\Rightarrow$ (implicaţia)
- $\approx$ (echivalenţa)

Propoziţiile, în calculul propoziţional, sunt formate din aceste simboluri (atomi) corelate de urmatoarele reguli:

- Fiecare simbol propozitional şi simbolul de adevăr constituie câte o propoziţie.

  ex.: true, P, Q şi R sunt 4 propoziţii.

- Negaţia unei propoziţii este o propoziţie.

  ex.: $\neg$ P şi $\neg$ false sunt 2 propoziţii

In expresia P $\wedge$ Q, *P* şi *Q* sunt conjunctori. In expresia P $\vee$ Q, *P* şi *Q* sunt disjunctori. In implicaţia P $\Rightarrow$ Q, *P* este premisa, iar *Q* este concluzia sau consecinţa. Pentru a controla ordinea de efectuare a evaluării se folosesc parantezele (,[,),].

Semantica pentru calculul propoziţional.

Adevărul concluziilor unei judecăţi depinde de adevărul cunoştinţelor iniţiale. Fiecare simbol propoziţional reprezintă o parte a realităţii. Interpretarea fiecărui simbol, în raport cu elementul reprezentat, înseamnă atribuirea unei valori de adevăr: *adevarat* sau *fals*. Pentru un ansamblu de 2 propoziţii există 4 posibilităţi. Propoziţiile care folosesc operatorii logici pot fi reprezentate grafic, foarte sugestiv, prin tabelele de adevăr. In tabelul T3.1 sunt prezentate negaţia, conjuncţia, disjuncţia, sau exclusiv.

P	Q	$\neg$ P	P $\wedge$ Q	P $\vee$ Q	P $\Rightarrow$ Q	P $\Leftrightarrow$ Q
1	1	0	1	1	1	1
1	0	0	0	1	0	0
0	1	1	0	1	1	0
0	0	1	0	0	1	1

T 3.1

In calculul propoziţional se folosesc o serie de identităţi, pentru a transforma o expresie, în altă expresie echivalentă.

legea dublei negaţii $\neg(\neg P) \approx P$

$(P \vee Q) \approx (\neg P \Rightarrow Q)$

legile lui De Morgan : $\neg (P \wedge Q) \approx (\neg P) \vee (\neg Q)$

$\neg (P \vee Q) \approx (\neg P) \wedge (\neg Q)$

Tabelele de adevăr pentru legile De Morgan sunt date mai jos.

P	Q	$\neg$P	$\neg$Q	P $\wedge$ Q	$\neg$(P $\wedge$ Q)	$\neg$ P $\vee$ $\neg$Q	$\neg$(P$\wedge$Q) $\Leftrightarrow$ $\neg$P$\vee\neg$Q
1	1	0	0	1	0	0	1
1	0	0	1	0	1	1	1
0	1	1	0	0	1	1	1
0	0	1	1	0	1	1	1
P	Q	$\neg$P	$\neg$Q	P $\vee$ Q	$\neg$(P $\vee$ Q)	$\neg$ P $\wedge$ $\neg$Q	$\neg$(P$\vee$Q) $\Leftrightarrow$ $\neg$P$\wedge\neg$Q
1	1	0	0	1	0	0	1
1	0	0	1	1	0	0	1
0	1	1	0	1	0	0	1
0	0	1	1	0	1	1	1

T 3.2

asociativitatea : $((P \wedge Q) \wedge R) \approx (P \wedge (Q \wedge R))$ **(3.1)**

$((P \vee Q) \vee R) \approx (P \vee (Q \vee R))$ **(3.1')**

distributivitatea : $P \vee (Q \wedge R) \approx (P \vee Q) \wedge (P \vee R)$ **(3.2)**

$P \wedge (Q \vee R) \approx (P \wedge Q) \vee (P \wedge R)$ **(3.2')**

comutativitatea : $(P \wedge Q) \approx (Q \wedge P)$ **(3.3)**

$(P \vee Q) \approx (Q \vee P)$ **(3.3')**

idempotenţa: $P \vee P \approx P$ **(3.4)**

$P \wedge P \approx P$ **(3.4')**

contrapunerea : $(P \Rightarrow Q) \approx (\neg Q \Rightarrow \neg P)$ **(3.5)**

P	Q	$\neg P$	$\neg Q$	$P \Rightarrow Q$	$\neg Q \Rightarrow \neg P$	$(P \Rightarrow Q) \Leftrightarrow (\neg Q \Rightarrow \neg P)$
1	1	0	0	1	1	1
1	0	0	1	0	0	1
0	1	1	0	1	1	1
0	0	1	1	1	1	1

T 3.3

P	Q	R	$P \Rightarrow Q$	$Q \Rightarrow R$	$(P \Rightarrow Q) \wedge (Q \Rightarrow R)$	$P \Rightarrow R$	$(P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)$
1	1	1	1	1	1	1	1
1	1	0	1	0	0	0	1
1	0	1	0	1	0	1	1
1	0	0	0	1	0	0	1
0	1	1	1	1	1	1	1
0	1	0	1	0	0	1	1
0	0	1	0	1	0	1	1
0	0	0	0	1	0	1	1

T 3.4

legea silogismului: $(P \Rightarrow Q) \wedge (Q \Rightarrow R) \Rightarrow (P \Rightarrow R)$ **(3.6)**

Aceste identităţi se folosesc pentru a transforma o expresie, în altă expresie logică echivalentă. In calculul propoziţional se notează cu V, mulţimea variabilelor propoziţionale şi cu F mulţimea formulelor.

d17. Se numesc **literali pozitivi** toate elementele lui V

d18. Negaţiile elementelor lui V se numesc **literali negativi**.

d19. O disjuncţie de literali se numeşte **clauză.**

O clauză este o formulă, a calculului propoziţional, alcătuită dintr-o mulţime de literali.

d20. O clauză care are cel mult un literal pozitiv se numeşte **clauză Horn**.

d21. O clauză care conţine exact un literal pozitiv se numeşte **clauză Horn strictă**.

d22. Clauza $\neg x \vee \neg y \vee z$ este logic echivalentă cu $\neg (x \wedge y) \vee z$, care devine $x \wedge y \Rightarrow z$. Această formulă în limbaj natural înseamnă: "z este adevărat dacă x şi y sunt adevărate". In programarea logică aceasta se notează sub forma:

$$z :- x , y \qquad \text{şi se numeşte } \textbf{regulă.} \qquad \textbf{(3.7)}$$

d23. O clauză Horn care se reduce la un singur literal pozitiv se numeşte **fapt**.

d24. O mulţime de clauze Horn desemnează o **bază de date logice.**

d25. O interogare a unei baze de date logice $\{a_1,...,a_n\}$ este echivalentă cu o deducţie semantică de forma, $\{a_1,...,a_n\} \models C$. Pentru cazul particular în care baza conţine numai clauze Horn stricte, deducţia semantică are ca rezultat, $\{p, q, \neg p \vee \neg q \vee r\} \models r$, în care **r** se mai numeşte **scop**.

d26. **Metoda rezoluţiei**. Pentru formulele a, b, $p \in F$ dacă sunt adevărate $a \vee p$ şi $b \vee \neg p$, atunci este adevărată şi formula $a \vee b$; deci

$$\{ a \vee p, b \vee \neg p \} \models a \vee b. \qquad \textbf{(3.8)}$$

d27. Regula rezoluţiei generalizează regula de raţionament, cunoscută în logică sub numele de **modus ponens**, care precizează că dacă p şi $p \Rightarrow q$ sunt adevărate, atunci q este adevărată.

Demonstrarea inconsistenţei unei formule şi implicit deducţia semantică se poate face pe baza rezoluţiei. Mulţimile de clauze $Q_1,...,Q_n$ sunt înlănţuite prin rezoluţie, dacă pentru fiecare $i \in \{2,...,n\}$ mulţimea Q_i se obţine din Q_{i-1} prin adăugarea unei singure clauze, obţinută ca rezolvantă a două clauze din Q_{i-1}.

Inconsistenţa unei mulţimi de clauze.

d28. Mulţimea Q de clauze este **inconsistentă,** dacă şi numai dacă, clauza vidă f se poate obţine prin rezoluţie din Q.

Procedural, se calculează rezolvanta r a două clauze din Q şi se înlocuieşte Q cu $Q \cup \{r\}$. Operaţia se repetă atâta timp cât clauza vidă $f \notin Q$.

Calculul predicatelor.

d29. Predicatul este un simbol de forma $P(t_1,...,t_n)$, creeat cu scopul de a reprezenta realitatea. P este o constantă predicativă, iar $t_1,...,t_n$, sunt constante.

d30. Un **atom** este o formă predicativă sau o egalitate de forma $t_1 = t_2$.

Formal, predicatul este un şir de caractere care respectă următoarele reguli:

- este format din literele alfabetului Englez;
- cuprinde cifrele 0...9;
- cuprinde caracterul de subliniere;
- începe întotdeauna cu o literă.

Simbolurile calculului predicatelor pot reprezenta: variabile, constante, funcţii sau predicate.

d31. Constantele sunt simboluri, care nu îşi schimbă în timp, conţinutul reprezentat. Formal, ele încep întodeauna cu o literă mică, de exemplu: obiect, alexandru, verde, înalt.

d32. Variabilele sunt simboluri, care îşi schimbă în timp, conţinutul reprezentat şi desemnează clase generale de obiecte şi proprietăţi.

Formal, variabilele sunt reprezentate de simboluri, care încep cu o literă mare. Calculul predicatelor nu include calculul numeric, în schimb acesta este definit axiomatic folosind predicate predefinite. O expresie funcţională este un simbol de funcţie urmat de argumentele funcţiei. Argumentele sunt elemente din domeniul funcţiei. Argumentele sunt incluse în paranteze şi sunt separate de virgule. De exemplu: g (Y,Z), preţ(motor,mare), curea_de_transmisie(pirrelli), plus(3,4).

d33. Acţiunea de înlocuire a funcţiei, cu valoarea sa, se numeşte **evaluare**. Termenii în calculul predicatelor pot fi constante, variabile sau expresii funcţionale.

Calculul predicatelor include:

1. Simbolurile de adevăr: *true* şi *false* (simboluri rezervate);
2. Simboluri de constante;
3. Simboluri de variabile;
4. Simboluri de funcţii; sunt simboluri de expresii care încep cu literă mică.

Argumentele unui predicat sunt termenii acestuia şi pot fi variabile sau expresii funcţionale. Propoziţiile atomice sunt numite expresii atomice, atomi sau propoziţii. Predicatele cu acelaşi nume, dar cu domenii diferite, sunt considerate distincte.

Ex.: componenta(arbore_de_intrare, reductor_conic),

componenta(arbore(arbore_intermediar), reductor(reductor_planetar)).

Prin combinarea, legarea propoziţiilor cu ajutorul operatorilor logici se obţin propoziţii noi. O variabilă, ca argument într-o propoziţie, se referă la un obiect nespecificat din domeniu. Calculul predicatelor cuprinde doi cuantificatori ai variabilelor:

d34. - **Cuantificatorul universal** $\forall$ indică faptul că, propoziţia este adevărată, pentru toate valorile din domeniu luate de variabilele sale.

$\forall$ X transmite(X, mişcarea).

d35. - **Cuantificatorul existenţial** $\exists$ indică faptul că, o propoziţie este adevărată, numai pentru anumite valori din domeniu.

$\exists$ Y asigură(Y, ungerea).

d36. O formulă de expresie $C_1x_1\ C_2x_2,...,C_nx_n\ M$, unde $C_1,...,C_n$ sunt cuantificatori (existenţial şi/sau universal), iar M este o expresie care nu conţine cuantificatori, se numeşte **formă prenex.**

Expresia M se numeşte matrice. Orice formulă poate fi adusă la o formă prenex echivalentă. Tranformarea unei formule în formă prenex se face cu ajutorul proprietăţilor următoare:

$$A \Leftrightarrow B \approx (A \Rightarrow B) \wedge (A \Leftarrow B) \quad \textbf{(3.9)}$$

$$A \Rightarrow B \approx (\neg A \vee B) \quad \textbf{(3.10)}$$

$$\neg (A \vee B) \approx \neg A \wedge \neg B \quad \textbf{(3.11)}$$

$$\neg (A \wedge B) \approx \neg A \vee \neg B \quad \textbf{(3.12)}$$

$$\neg \neg A \approx A \quad \textbf{(3.13)}$$

d37. O formă prenex a cărei matrice este o conjuncție de clauze, se numeşte **formă normal conjunctivă.**

Se poate ataşa unei formule A, o formulă S_A, astfel încât, să existe proprietatea ca: A şi S_A să fie simultan consistente şi inconsistente. Formula S_A se numeşte Skolem.

d38. O formulă este în **formă Skolem,** dacă este în formă prenex şi prefixul ei nu conține cuantificatori existențiali. Deoarece prefixul formei Skolem conține numai cuantificatori universali, prin convenție, nu se mai scrie.

d39. O formă Skolem, în care matricea este sub formă normal conjunctivă, se numeşte **formă clauzală.**

O semantică pentru calculul predicatelor.

In calculul predicatelor există două căi de folosire sau cuantificare a variabilelor. Prima, constă în a considera propoziția adevarată, pentru toate constantele, care pot fi substituite pentru interpretarea variabilelor.

Astfel:

$$\forall X(p(X)) \wedge q(Y) \Rightarrow r(X)) \qquad \textbf{(3.14)}$$

indică faptul că, X este o variabilă cuantificată universal, atât în p(X), cât şi în r(X). Dacă domeniul unei interpretări este infinit, testul de adevăr al tuturor substituțiilor unei cuantificări universale este imposibil, algoritmul nu se opreşte niciodată, iar calculul predicatului este indecis. Variabilele pot fi cuantificate existențial. In acest caz, expresia se numeşte adevarată, pentru cel puțin o substituție din domeniul de definiție.

Evaluarea adevărului unei expresii ce conține cuantificatorul existențial poate să nu fie mai uşoară decât una ce conține cuantificatorul universal. Dacă domeniul variabilei este infinit, iar expresia este falsă pentru toate substituțiile, algoritmul nu se opreşte niciodată, cu aceleaşi consecințe ca şi în cazul precedent. Câteva relații dintre cuantificatorii existențial şi universal sunt date mai jos:

$$\begin{aligned}
&\neg \exists X\, p(X) = \forall X\ \neg p(X)\\
&\neg \forall X\, p(X) = \exists X\ \neg p(X)\\
&\exists X\, p(X)\ = \exists Y\, p(Y) \qquad \textbf{(3.15)}\\
&\forall X\, (p(X) \wedge q(X)) = \forall X\, p(X)\ \wedge \forall Y\, q(Y)\\
&\exists X\, (p(X) \vee q(X)) = \exists X\, p(X) \vee\ \exists Y\, q(Y)
\end{aligned}$$

In limbajul definit până acum variabilele cuantificate universal, respectiv existențial, pot să se refere numai la obiecte (constante) din domeniul discursului. Predicatele şi numele de funcții nu pot fi înlocuite de variabile cuantificate. Acest limbaj se numeşte calculul predicatelor de ordinul întâi. Aproape toate propozițiile formulate corect, din punct de vedere gramatical, în Engleză pot fi reprezentate în predicate de ordinul întâi, folosind simboluri, conectori şi variabile. Exemple:

Dacă roțile nu sunt coplanare, transmisia va funcționa defectuos.

sau

$\neg$ poziție (roți, coplanar[1]) $\Rightarrow$ funcționare (transmisie , defectuos)

Toate angrenajele sunt zgomotoase.

[1] Conceptul de coplanar se referă la coplanaritatea axelor de rotație, vectorilor viteză unghiulară şi la planele mediane ale roților.

sau

$$\forall X (\text{angrenaje} (X) \Rightarrow \text{zgomotoase} (X))$$

In calculul predicativ, chestiunea importantă nu este unicitatea reprezentării, ci faptul că, interpretarea poate sau nu să furnizeze o valoare de adevăr pentru toate expresiile din set, în condiţiile în care expresiile descriu realitatea cu suficiente detalii, astfel ca deducţia să poată fi dusă la bun sfârşit, prin manipularea expresiilor simbolice.

Reguli şi proceduri de inferenţă.

Semantica calculului predicatelor oferă o bază pentru teoria deducţiei logice. Capacitatea de a deduce expresii corecte noi, pornind de la un set de afirmaţii adevărate, este o trăsătură importantă a calculului predicatelor. O expresie X, din calculul predicatelor, este dedusă logic dintr-un set de expresii, dacă fiecare interpretare şi fiecare atribuire de variabile, satisface atât pe S, cât şi pe X. O regulă de deducţie este solidă, dacă fiecare expresie obţinută din setul de expresii S, derivă de asemenea logic din S. O regulă de deducţie este completă, dacă dându-se un set S de expresii, aceasta poate să deducă fiecare expresie care derivă logic din S. Dacă expresiile P şi $P \Rightarrow Q$ sunt adevărate, acestea permit să-l deducem pe Q. Dacă $P \Rightarrow Q$ este adevărată , iar Q este falsă, atunci se poate deduce $\neg P$. Dacă $P \wedge Q$ este adevărată, atunci P, respectiv Q sunt adevărate. Reciproc, dacă P şi Q sunt adevărate, atunci şi $P \wedge Q$ este adevărată.

Unificarea.

Un sistem de inferenţă (deducţie) trebuie să fie capabil să determine, când două expresii sunt la fel, sau când pot fi comparate. In calculul propoziţional, aceasta înseamnă că, două expresii pot fi comparate, numai şi numai dacă, sunt identice din punct de vedere sintactic. Procesul de comparare a două propoziţii este complicat de existenţa variabilelor în expresii. Instanţierea universală permite variabilelor cuantificate să fie înlocuite prin termeni echivalenţi din domeniu. Aceasta cere un proces de decizie, pentru determinarea substituţiilor de variabile, sub acţiunea căruia două sau mai multe expresii pot fi făcute identice.

d40. **Unificarea** este un algoritm, care determină substituţiile necesare, pentru a face două expresii compatibile (procesabile). In momentul când P(X) şi P(Y) sunt echivalente, Y poate fi substituit prin X, pentru a face propoziţiile comparabile.

Exemplu clasic: om(socrates),

$X(\text{om}(X)) \Rightarrow \text{muritor}(X)$

Din cele două propoziţii prin unificare rezultă:

muritor(socrates)

Unificarea şi regulile de inferenţă ne permit să efectuăm deducţii, asupra unui set de expresii logice, dar baza de date trebuie pusă într-o formă adecvată. Unificarea cere ca toate variabilele să fie cuantificate universal. Cuantificarea universală oferă libertate deplină în procesarea substituţiilor. Variabilele cuantificate existenţial pot fi eliminate din propoziţii, în baza de date, prin înlocuirea lor cu constante, care fac propoziţia adevărată.

De exemplu: $\exists X$, antrenează(X, pompa)

poate fi înlocuită cu expresia

antrenează(motor_electric, pompa) sau

antrenează(motor_termic, pompa).

Procesul de eliminare al variabilelor cuantificate existenţial este complicat de faptul că, valoarea acestor substituţii poate să depindă de valoarea altor variabile din expresie. De exemplu, în expresia:

$$\forall X, \exists Y \text{ antrenează}(Y,X)$$

valoarea variabilei existenţiale Y depinde de valoarea lui X. Rezolvarea situaţiei se face cu o funcţie Skolem care înlocuieşte variabila existenţială Y, dependentă de X. In exemplul anterior predicatul devine:

$$\forall X \text{ antrenează}(X, f(X)).$$

După ce variabilele existenţiale au fost înlocuite de cunoştinţele din baza de cunoştinţe, unificarea poate să înceapă, în ordinea aplicării regulilor de inferenţă. Unificarea este complicată şi de faptul că, o variabilă poate fi înlocuită de orice termen, incluzând alte variabile şi expresii funcţionale, de complexitate arbitrară. Algoritmul unificării, care realizează substituţia, cere două expresii şi un număr de surse de cunoştinţe. Câteva din regulile unificării sunt următoarele:

- o constantă poate fi substituită printr-o variabilă;
- orice constantă este considerată "instanţă fundamentală" (ground instance) şi nu poate fi reamplasată;
- două constante diferite nu pot fi substituite niciodată printr-o variabilă unică;
- o variabilă nu poate fi unificată cu un termen care conţine acea variabilă.

Ex: X nu poate fi înlocuită cu p(X), căci aceasta creează o expresie infinită de forma: p(p(p(...X))...). Procesul de rezolvare al unei probleme necesită inferenţe multiple şi unificări succesive. De exemplu:

p(a, X), unificat cu premisa $p(Y,Z) \Rightarrow q(Y,Z)$,

urmată de substituţia {a/Y, X/Z}, ne conduce la deducţia

q(Y,Z), urmată de aceeaşi substituţie q(a,X).

Dacă procesăm acest rezultat cu premisa:

$q(W,b) \Rightarrow r(W)$ putem deduce r(W), ca urmare a substituţiei {a/W, b/X}.

Setul {a/W, b/X} este derivat din compunerea primului set {a/Y, X/Z}, rezultat din prima unificare cu cel de al doilea {Y/W, b/Z}.

3.1.2. Reprezentarea cunoaşterii prin reţele de producţii

Cunoaşterea în reţelele de producţii este de natură procedurală [14] şi are următoarele componente:

- cunoaştere declarativă sau factuală (structuri de date, **context**);
- cunoaştere procedurală (baza de reguli);
- cunoaştere strategică sau de control (secvenţele de acţiuni în procesul de rezolvare).

Contextul poate furniza informaţii, care nu depind de importanţa pieselor de cunoaştere, ci doar de utilitatea şi gradul de activare al acestora. *Informaţiile de context* se referă la: *vechimea piesei* de cunoaştere, apreciată după data înscrierii în context; *numărul de participări* ale piesei de cunoaştere la regulile declanşate; *numărul de neutilizări*, etc.

In esență, un sistem de producții este compus dintr-o bază de date şi o un set de reguli. Concluzia unei reguli este o acțiune şi un control; acțiunea manipulează baza de date, iar controlul determină secvența regulilor utilizate.

d41. Un sistem de producții este un ansamblu cvintuplu de tipul **SP = (K, P, σ, φ, p_S)** în care,

- **K** reprezintă contextul format dintr-o mulțime de piese de cunoaştere factuală numite fapte, care sunt recunoscute ca atare de celelalte componente ale sistemului;
- **P** reprezintă baza de reguli şi este o mulțime finită de reguli;
- σ reprezintă funcția succesor, funcția care succede îndeplinirea cu succes a condiției (este o aplicație, definită pe P, cu valori în, $P \cup \{\lambda\}$);
- φ reprezintă funcția succesor, funcție care succede eşecul condiției;
- p_S reprezintă regula de producție inițială, de la care porneşte procesul de selectare al regulilor.
- λ producția vidă.

Forma generală a unei reguli de producție este dată de expresia 4.16

$$p = (c_1 \wedge c_2 \wedge \ldots \wedge c_i) \rightarrow (a_1; a_2; \ldots ; a_p) \qquad \textbf{(3.16)}$$

în care c_j, $1 <= j <= i$, sunt condițiile, iar a_k , $1 <= k <= p$, sunt acțiunile.

Controlul sistemelor de producții este specificat complet de cele trei componente ale sale σ, φ şi p_S. Strategiile de control pot fi, însă, diferite, de la caz la caz. Prima şi cea mai puțin rafinată, este strategia exhaustivă, care selectează toate regulile ce se potrivesc condiției date de context. Aceasta poate fi scrisă astfel:

contextul $K = \{k_1, k_2, \ldots, k_n\}$ **(3.17)**

producțiile $P = \{ p_1, p_2, \ldots, p_m \}$ **(3.18)**

strategia este următoarea:

$$\begin{cases} \sigma(p_i) = \varphi(p_i) = p_{i+1}, & \textbf{pentru } 1 \le i \le m-1 \\ \sigma(p_n) = \varphi(p_n) = p^n \\ \sigma(p_n) = \varphi(p_n) = \lambda \end{cases} \qquad \textbf{(3.19)}$$

$$p_S = p_1 \qquad \textbf{(3.20)}$$

Mulțimea regulilor, ce conțin piese de cunoaştere, formează "mulțimea candidată". Mulțimea aceasta intră într-o competiție, în urma căreia, se decide care regulă este aplicată efectiv. Apoi, se elimină regulile care duc la acelaşi rezultat, având ca rezultat modificarea contextului. Se execută părțile active ale regulilor, care sunt aplicabile. Se reia ciclul, pornind de la contextul modificat, până când se obține o nouă modificare a contextului. Oprirea mecanismului deductiv se face, dacă acțiunea unei producții o specifică, sau dacă se selectează o producție vidă.

O altă strategie de control cercetează condițiile din context, începând cu prima regulă, iar la întâlnirea primei reguli aplicabile o aplică, după care comută controlul de regulă p_1, cu noul context. In situația de succes, controlul este comutat la regula p_{i+1} cu noul context. Se realizează astfel o explorare succesivă a zonei aplicabile din context. Strategia este următoarea

$$\begin{cases} \sigma(p_i) = p_1, & \textbf{pentru } 1 \le i \le n \\ \varphi(p_i) = p_{i+1}, & \textbf{pentru } 1 \le i \le n-1 \\ \varphi(p_n) = \lambda \end{cases} \qquad (3.21)$$

$$p_S = p_1 \qquad (3.22)$$

Atunci când nici o regulă nu este aplicabilă, se selectează, conform funcţiei succesor la eşec, producţia vidă.

O altă strategie posibilă, alege din mulţimea candidată, producţia cu cel mai mare câştig informaţional, adică se începe cu regula care are în partea premisă cele mai multe condiţii, iar în partea acţiune are cele mai multe acţiuni.

Performanţele sistemelor de producţii.

Sistemele de producţii sunt caracterizate printr-o autonomie ridicată. Regulile sunt tratate ca elemente de cunoaştere independente. Prin acumularea cunoaşterii sistemul devine din ce în ce mai neperformant, în privinţa timpului de rezolvare al sarcinilor. Se pot lua măsuri pentru îmbunătăţirea performanţelor sistemului. Acestea urmăresc:

- organizarea contextului pentru o explorare parţială;
- organizarea bazei de reguli pentru evitarea, în procesul de inferenţă, a regulilor neadecvate;
- înglobarea în grupurile de piese de cunoaştere a unor funcţii auxiliare, care faciliteză accesul, selecţia, execuţia şi comunicarea.

O problemă importantă a sistemelor de producţii o constituie inconsistenţa. Dezvoltarea bazelor de cunoştinţe, prin simplă completare cu reguli şi fapte, conduce inevitabil, la o bază de cunoştinţe inconsistentă. Se defineşte ca fiind inconsistentă acea bază D, pentru care se îndeplineşte următoarea condiţie:

$$(x = a) \in D \cup (x \neq a) \in D \qquad (3.23)$$

In vederea menţinerii consistenţei este necesar să se aibă în vedere faptul că, un parametru poate avea o singură valoare, sau mai multe valori. Restricţia prin care un parametru are valoare unică poate fi, ea însăşi, exprimată ca regulă. Păstrarea consistenţei este o problemă centrală a modulului de introducere al cunoştinţelor, în baza de cunoştinţe.

3.1.3 Reprezentarea cunoaşterii prin reţele semantice

Noţiunea de reţea semantică cuprinde o familie de reprezentări ale realităţii, cu ajutorul grafurilor. Atributul de "semantic" provine de la primele aplicaţii funcţionale ale acestor reţele, care datează din anii 1960. Reţeaua semantică a fost gândită, pentru a rezolva problema traducerii automate. **Masterman M.** [16] a definit, în 1960, un set de 100 tipuri de concepte primitive, pe care le-a folosit la alcătuirea unui dicţionar cu 15000 de concepte (definiţii).

Un program esenţial în dezvoltarea reţelelor l-a întocmit **Quillian Ross**, [17] în 1966, şi a avut ca scop, definirea cuvintelor Limbii Engleze într-un mod mult mai nuanţat decât o făceau dicţionarele. Fiecare definiţie de cuvânt leagă noţiunea de alte definiţii. Fiecare nod din reţeaua Quillian corespunde unui concept. Baza de cunoştinţe a fost organizată în plane, fiecare plan conţinea un graf, care reprezintă un cuvânt. Lucrările lui Quillian au stabilit multe dintre trăsăturile semnificative ale reţelelor semantice, cum ar fi: arce cu denumire, moştenire

ierarhică, deducţie prin legături asociative. Această reprezentare cuprinde o mare varietate de metode ce folosesc structura de reţea, pentru construirea bazelor de cunoştinţe. Descrierea se face cu ajutorul conceptelor legate prin relaţii. Se obţine o structură ierarhică ca în Fig. 3.1. Relaţiile care pot exista într-o reţea pot fi de următoarele tipuri:

- relaţii epistemice - sunt acele relaţii care descriu semnificaţia unui concept cu ajutorul unor concepte mai puţin generale;
- relaţii compoziţionale - sunt relaţiile care se stabilesc între concepte şi componentele lor;
- relaţii de apartenenţă - sunt relaţiile care exprimă apartenenţa conceptelor sau specimenelor la clase sau mulţimi;
- relaţii cauzale - sunt relaţiile care se stabilesc între concepte de tip obiect şi concepte de tip acţiune;
- relaţii modale - sunt relaţiile care se stabilesc între componente şi modalităţile de ataşare a acestora la concepte.

d42. Relaţiile taxonomice dintre obiectele plasate în nodurile reţelei sunt descrise cu ajutorul primitivelor relaţionale, care asigură o interpretare uniformă a relaţiilor în toată reţeaua.

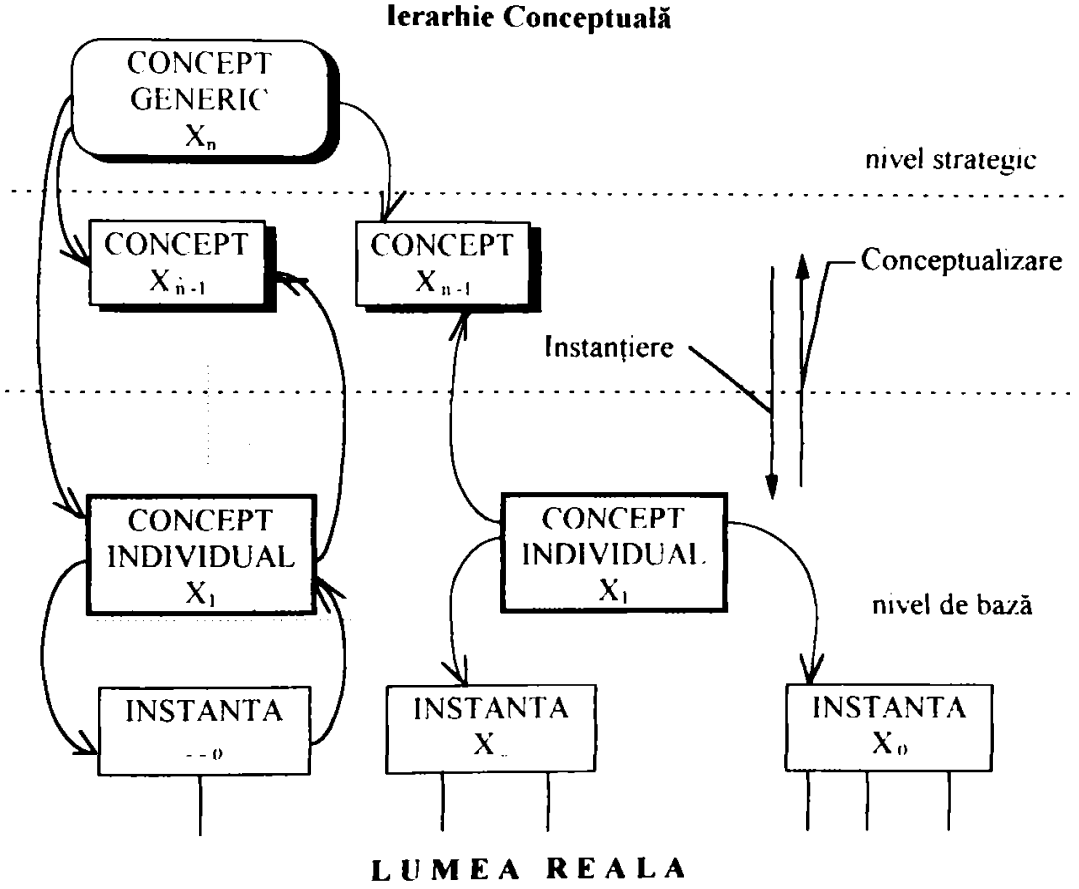


Fig.3.1

Principalele primitive taxonomice sunt:

relaţia de apartenenţă ESTE (< element >, < clasă >);

relaţia de incluziune. ESTE_SUBMULTIME (< mulţime 1>, <mulţime 2 >).

Reprezentarea cunoaşterii cu ajutorul primitivelor semantice este restrânsă. Instrumentul cel mai performant de reprezentare al cunoaşterii este limbajul natural.

Pentru aplicaţiile de Inteligenţă Artificială au fost elaborate o serie de primitive semantice, dintre care mai cunoscute sunt primitivele elaborate de **Wilks Yorrick** [18]. In limbajul natural există cuvinte cu semnificaţii primitive şi cuvinte fără semnificaţii primitive. Pornind de la această observaţie se definesc următoarele clase de primitive semantice:

- primitive care reprezintă entităţi;
- primitive care reprezintă acţiuni;
- primitive care reprezintă relaţii între entităţi de tip obiect;
- primitive care reprezintă calificative ale entităţilor şi acţiunilor (adjective şi adverbe);
- primitive care reprezintă tipuri de entităţi;
- primitive care reprezintă clase de primitive.

Toate cuvintele care nu sunt primitive se definesc cu ajutorul primitivelor semantice; se asigură astfel, consistenţa sistemului. Intr-o reţea semantică, conceptele sunt legate prin relaţii numite dependenţe. Acestea sunt de tip cauzal, de tip subiectival şi de tip predicativ. Reţelele semantice pot fi clasificate, funcţie de structură şi tipul de reprezentare, în:

- reţele semantice simple;
- reţele semantice sortate;
- reţele semantice extinse.

Reţelele semantice simple sunt omogene structural. Ele nu oferă noduri preferenţiale pentru iniţierea procesului de interpretare. Selecţia unui nod, ca nod iniţial, este determinată de aplicaţia pentru care este construită baza de cunoştinţe. Descrierea unei reţele semantice se poate face, fie cu ajutorul tripletelor (<nod_sursă> <relaţie> <nod_destinaţie>), fie prin asocierea fiecărui nod sursă a unei perechi (<relaţie> <nod_destinaţie>).

Reţelele semantice sortate folosesc noduri şi arce specializate. Se foloseşte o tehnică de etichetare a nodurilor şi a relaţiilor. Câteva tipuri de noduri sunt: predicaţia, agentul, receptorul, obiectul, instrumentul, locul, timpul.

3.1.4 Reprezentarea cunoaşterii prin cadre. Ontologii.

Cadre.

Reprezentarea prin cadre reuneşte o parte din conceptele de bază ale reprezentării procedurale, prin sisteme de producţii şi prin reţele semantice.

d43. Un cadru este un obiect-soft structurat, care conţine informaţia despre un obiect din universul reprezentat, o acţiune, o regulă într-o formă stereotipă.

Locaţiile cadrului conţin informaţii despre:

- identificarea cadrului;
- relaţiile existente între cadrul nominalizat şi celelalte cadre;
- descriptori esenţiali necesari operaţiilor de procesare a datelor;

- considerente procedurale;
- date certe referitoare la cadru;
- date suplimentare.

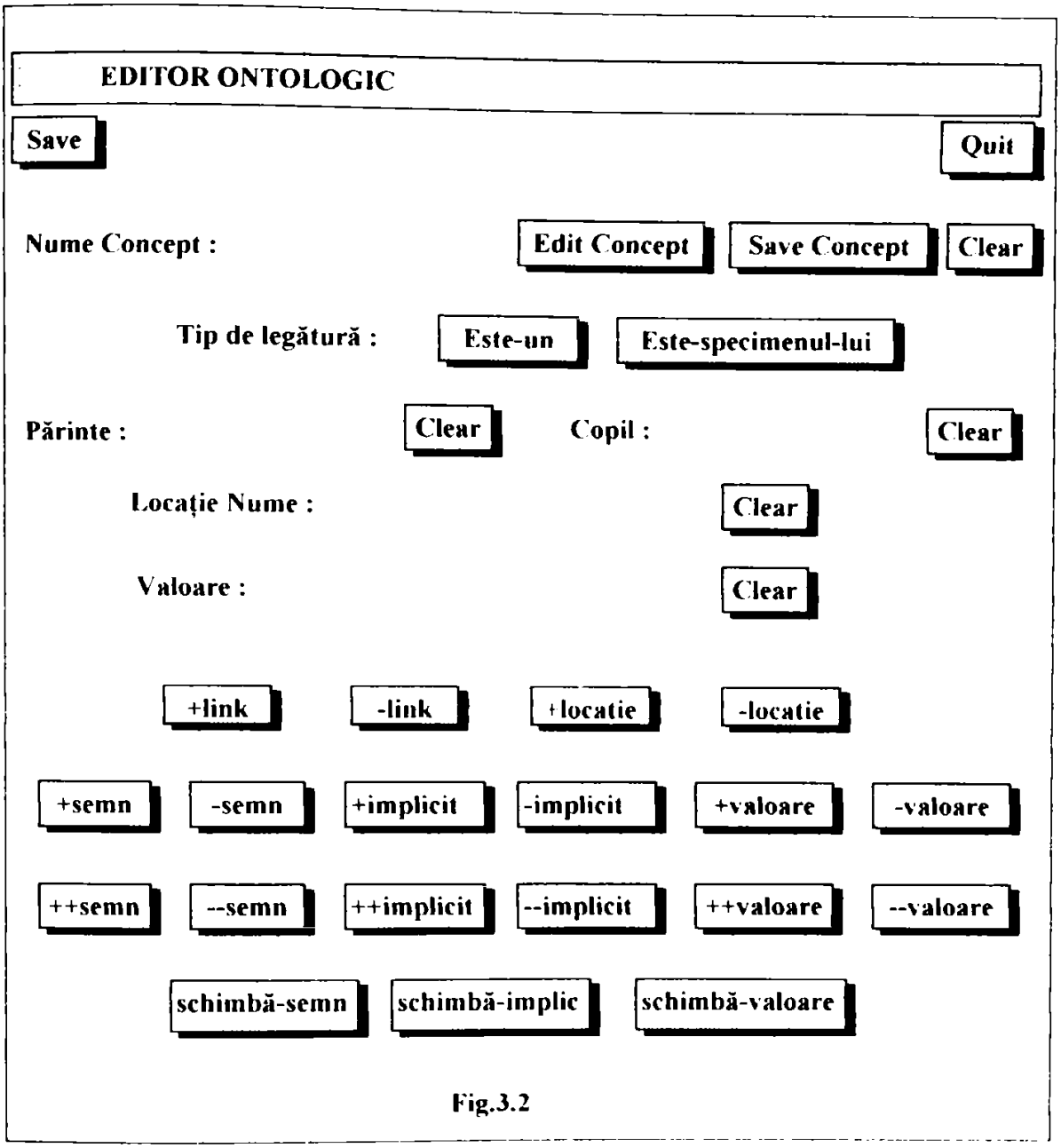


Fig.3.2

Cadrele fac posibilă o organizare ierarhică uşoară a cunoaşterii [7]. Într-o reţea fiecare concept este reprezentat prin noduri şi arce, aflate la acelaşi nivel de specificare.

O altă trăsătură importantă a cadrelor este posibilitatea ataşării procedurilor. Sistemele care utilizează cadre au calitatea de moştenire a trăsăturilor clasei.

Atunci când se instanţiază un cadru, sistemul încearcă să-i completeze locaţiile de valoare, fie prin moştenirea valorii clasei din care face parte, fie prin execuţia unei proceduri predefinite. Un cadru este un instrument soft, care foloseşte la normalizarea reprezentării cunoaşterii, în

sistemele de Inteligență Artificială. Conținutul elementar al cadrului este dependent de ontologia care stă la baza sistemului. Din punct de vedere grafic un cadru de editare arată ca în Fig.3.2.

3.1.5. Ontologii

Termenul de ontologie, în ştiința calculatoarelor, se referă la un model de organizare al reprezentării realității, prin concepte înlănțuite într-o ierarhie particulară. Ontologia, în filosofie, este un proiect de cunoaştere atotcuprinzător şi unitar a ceea ce există, sau ar putea exista în realitate, adică o teorie a existenței [34]. Conceptele care alcătuiesc sfera noțiunii de ontologie sunt: *lucrurile, evenimentele şi relațiile*.

Intrările în ontologie se referă la noțiunea generică a conceptului în modelul adoptat. Pe lângă legăturile taxonometrice de tip *este-un*, ontologia mai conține numeroase alte legături între concepte. Toate legăturile pot fi reprezentate prin arce de graf cu etichete, respectiv prin locații, într-un chestionar cadru. Relațiile, însele, reprezintă concepte ale ontologiei şi apar în locațiile de ierarhie şi moştenire. Este util pentru o ontologie să aibă o bază de cunoştințe cu specimenele particulare ale conceptelor pe care le stochează. Specimenele bazei de cunoştințe au nume şi se poate face referire la ele prin nume. Baza de cunoştințe mai poate folosi pe lângă nume şi un cod. Baza de cunoştințe a specimenelor poate fi statică sau dinamică.

Cunoaşterea este împărțită în două baze de cunoştințe: una care cuprinde cunoaşterea despre concepte şi cea de-a doua, care se mai numeşte *onomasticon* şi conține specimenele care concretizează conceptele.

Principiile proiectării ontologice

De la bun început trebuie precizat că, nu exită un set de principii consacrate, care să definească cu precizie structura sau conținutul unei ontologii. Ideile care s-au dovedit utile, pentru construirea unei ontologii practice, sunt prezentate succint mai jos. O practică mai îndelungată va selecta şi completa ideile principale, care guvernează acest domeniu.

1. Ontologia este un **limbaj independent**, nu un dicționar de echivalențe între două limbaje. Ea nu este specifică nici unui limbaj natural. Conceptele care aparțin ontologiei nu procedează la o atribuire unu-la-unu, în sensul propriu, al limbajului natural. Unele dintre concepte se pot conecta cu un singur cuvânt din limbajul natural, altele însă, cu mai multe.
2. O ontologie trebuie să fie **motivată independent**, adică să nu fie aservită lexicului.
3. O ontologie trebuie să fie **bine formată,** în acord cu o specificație axiomatică precisă.
4. Ontologia trebuie să fie **consistentă** şi compatibilă cu alte resurse de cunoaştere şi de procesare: lexic, analizor gramatical, etc.
5. Ontologia trebuie să fie **bogată în conținut**, concepte de structură, grad de conectivitate între concepte.
6. **Expresivitatea** ontologiei trebuie să fie **limitată.** Aceasta îi ajută pe utilizatori să parcurgă şi să condenseze cunoaşterea.
7. Ontologia trebuie să ofere o **căutare uşoară a conceptelor** şi să fie uşor de condensat.
8. Ontologia trebuie să aibă o mare **utilitate** pentru îndeplinirea sarcinilor
9. Ontologia trebuie să aibă **scop limitat**.
10. **Cunoaşterea** achiziționată de către ontologie este **conceptuală** şi nu episodică.

11. Ontologia **nu este dedicată exclusiv** unui domeniu, dar **este dirijată** spre un domeniu precis.

Ce concepte se introduc în ontologie ?

La construirea unei ontologii este important să se considere statutul ontologic al unui simbol şi să se decidă dacă simbolul trebuie introdus, sau nu.

1. Nu se introduc în ontologie specimene pe post de concepte.
2. Nu se introduc concepte care nu sunt imediat necesare.
3. Nu se introduc, ca şi concepte, evenimente speciale cu argumente particulare.
4. Achiziţia cunoaşterii se face sub formă conceptuală, se evită elementele episodice.

Teoria FRAME-ONTOLOGY

O parte dintre ideile ontologiei cadru sunt enunţate în [1001]. Ontologia cadru defineşte termenii, care capteză convenţiile folosite în sistemele de reprezentare ale cunoaşterii centrate pe obiect. Ontologia cadru este baza conceptuală pentru translatoarele de tip Ontolingua. Unul dintre scopurile acestei ontologii este de a oferi utilizatorilor posibilitatea să folosească diferite sisteme de reprezentare ale cunoaşterii.

Enunţurile axiomatice ale acestei ontologii sunt:

- relaţiile sunt seturi de ramuri -- numite prin predicate;
- funcţiile sunt cazuri speciale de relaţii;
- clasele sunt relaţii unnare -- nu există nici o sintaxă specială pentru aceste tipuri;
- semanticile extinse pentru clase -- sunt definite ca seturi şi nu ca descriptori;
- locaţiile sunt tratate numai cu relaţii binare şi cu funcţii unnare;
- relaţiile asupra relaţiilor sunt tratate de speciile KL-ONE.

Sunt definite următoarele clase:

partiţia de clasă

relaţia

funcţia

funcţia unnară

relaţia-unu-la-unu

evaluarea-singulară

relaţia-mai-mulţi-la-unu

clasa

relaţia-de-N-aritate

relaţia-unnară

relaţia-binară

funcţia-unnară

evaluarea-singulară

relaţia-asimetrică

alias; compus-din; specimen-direct-de; subclasă-directă-de; documentare; domeniu; domeniu-de; partiţie-exhaustivă-în-subclase; are-valoare; locaţie-pentru-valoare-moştenită; specimen_de; locaţie-pentru-maxim-de-cardinalitate; valoare-pentru-maxim-de-cardinalitate; locaţie-pentru-minim-de-cardinalitate; valoare-pentru-minim-de-cardinalitate; domeniu-al-N-lea; existenţa; magnitudinea; magnitudinea-lui; locaţie-pentru-valori-identice; valori-identice; locaţie-pentru-evaluare-singulară; locaţie-pentru-valoare-tipizată; este-subclasă-lui; partiţie-subclasă; este-superrelaţie-lui; este-superclasă-lui; total; valoare-tipizată.

locaţie-pentru-toate-valorile-moştenite; toate-specimenele; toate-valorile; arităţi; elemente-compuse; alcătuire domeniu-exact; magnitudi-nea-exactă; inversiune; este-unul-dintre; proiecţie; relaţia-cu-universul; locaţie-de-cardinalitate; valoare-de-cardinalitate

relaţia-antisimetrică

relaţia-de-ordine-parţială

relaţia-de-ordine-totală

relaţia-antireflexivă

relaţia-ireflexivă

relaţia-reflexivă

relaţia-echivalenţă

relaţia-de-ordine-parţială

relaţia-simetrică

relaţia-de-echivalenţă

relaţia-tranzitivă

relaţia-de-echivalenţă

relaţia-de-ordine-parţială

relaţia-tranzitivă-slabă

relaţia-mulţi-la-unu

relaţia-unu-la-mulţi

relaţia-mulţi-la-mulţi

obiect

Clasa RELATIE

O *clasă relaţie* este un set de ramuri, care reprezintă o relaţie dintre obiectele ce aparţin universului discursului. Fiecare ramură este o secvenţă finită şi ordonată de obiecte. *Relaţia* este la rândul ei un obiect, numit ca set de ramuri. Ramurile sunt şi ele entităţi, care aparţin universului discursului şi pot fi reprezentate ca obiecte individuale. Prin convenţie, relaţiile sunt definite prin precizarea restricţiilor, care trebuie plasate printre obiectele din fiecare ramură.

Relaţiile sunt denumite prin intermediul constantelor relaţionale. Faptul că, o anume ramură este membrul unei relaţii este dat de (<nume-relaţie> arg_1 arg_2 .... arg_n), unde arg_i sunt obiectele din ramură. In cazul relaţiilor binare faptul poate fi citit ca "<nume-relaţie> a arg_1 este arg_2". este-subclasa-lui Set

Axiome:

(<=> (Relaţia ?Relaţia)

(şi (Set ?Relaţia)

(Pentru_toate (?Ramura)

(=> (Membru ?Ramura ?Relaţia) (Lista ?Ramura)))))

Clasa FUNCTIE

O *clasă funcţie* este o formulă, care asociază un element domeniu, la un singur element magnitudine. La fel ca şi în clasa *relaţie*, elementele domeniului sunt ramuri. Magnitudinea este o clasă, iar un element al magnitudinii este un specimen al acelei clase. Funcţiile sunt obiecte de clasa întâi, în acelaşi sens în care, relaţiile sunt obiecte.

Funcţie este-subclasa-lui Relaţie.

Axiome:

(<=> (Funcţia ?Relaţia)
(And (Relaţia ?Relaţia)
(Pentru_toate (?Ramura1 ?Ramura2)
(=> (Membru ?Ramura1 ?Relaţia)
(Membru ?Ramura2 ?Relaţia)
(= (Aproape_ultimul ?Ramura1) (Aproape_ultimul ?Ramura2))
(= (Ultimul ?Ramura1) (Ultimul ?Ramura2))))))

Clasa CLASA

O *clasă* poate fi concepută ca o colecţie de entităţi distincte. Formal, o clasă este o relaţie unnară , adică un set de ramuri cu lungimea unu. Fiecare ramură conţine un obiect care se spune că este un specimen al clasei.

Clasa este introdusă pentru a completa vocabularul. Clasele au rolul de *sortimente* şi *tipuri*. Nu există nici o distincţie de ordinul întâi între clase şi relaţii unnare. Predicatele de ordinul doi fac distincţie între clase şi relaţii unnare.

Exemplu:

(specimen-al-lui, i, C)	i este specimen al lui C
(membru-al-lui, i, C)	i este membru al lui C

Cele două expresii nu sunt echivalente. Un specimen al unei clase nu este un membru al setului teoretic al clasei; mai mult, ramura care conţine specimenul este un element al unui set de ramuri, ceea ce constituie o relaţie.

Definiţia unei clase este un predicat ce operează asupra unei singure variabile libere, asfel încât, predicatul să lege specimenele clasei. Clasele sunt definite intenţional.

Clasa este-subclasa-lui relaţia

Clasa locaţia-lui specimen

Aritate: 1

Axiome:

(<=> (Clasa ?Clasa) (şi (Relaţia ?Clasa) (= (Mărimea ?Clasa) 1)))

Clasa OBIECT

Este o clasă care conţine orice obiect din universul discursului, care poate fi inclus într-o clasă. Aceasta include toate relaţiile şi obiectele predefinite sau definite de utilizator.

Relaţia SPECIMEN-AL

Un obiect este un *specimen-al* unei clase, dacă este un membru al unui grup, denumit de acea clasă. Dacă individualul i este specimen-al clasei C, aceasta se prezintă sub forma (C, i), ceea ce este echivalent cu relaţia (Specimen-al i C).

Relaţia *specimen-al* este utilă la definirea relaţiilor de ordinul doi. Un individual poate fi specimen în multe clase; o parte dintre ele pot fi subclasele altora.

Aritate: 2

Magnitudine: clasă

Axiome:

(<=> (Specimenul-lui ?Individ ?Clasa)

(şi (Clasa ?Clasa) (Capturează ?Clasa ?Individ)))

Funcţia TOATE-SPECIMENELE

Specimenele unor clase pot fi specificate extensional. Listarea specimenelor din clasa C se solicită astfel: (=(toate-specimenele C) (setof V_1 V_2...V_n)), unde V_i sunt specimenele. Funcţia impune o restricţie monotonă. Orice subclasă a lui C nu poate avea nici un specimen în afara lui *toate-specimenele lui C*.

Aritate: 2

Domeniu: Clasa

Magnitudine: Set

Axiome:

(<=> (toate-specimenele ?Clasa ?Set-de-specimene)

(şi (Clasa ?Clasa)

(Set ?Set-de-specimene)

(pentru-toate (?Specimen)

(<=> (Membru ?Specimen ?Set-de-specimene)

(specimenul-lui ?Specimen ?Clasa)))))

Funcţia ESTE-UNUL-DINTRE

Este-unul-dintre este o funcţie destinată definirii claselor, prin enumerarea specimenelor acestora. Funcţia primeşte ca variabile, un număr variabil de termeni şi defineşte clasa, exact prin specimenele pe care termenii le reprezintă.

De exemplu (este-unul-dintre *da nu*) defineşte clasa care conţine obiectele *da* sau *nu*. O funcţie compusă de forma (este-specimen-al-lui *da* (este-unul-dintre *da nu*)) este adevărată, pe când funcţia (este-specimen-al-lui *poate-fi* (este-unul-dintre *da nu*)) este falsă.

Utilitatea majoră a acestei funcţii este definirea restricţiilor tip. De exemplu, relaţia tip-valoare ia ca argument o clasă. Pentru a specifica un set finit de valori posibile pentru o locaţie, se poate folosi:

(tip-valoare ?Specimen ?Locaţie (este-unul-dintre a b c)).

Axiome:

(Nedefinită (Mărime Unul-dintre))

(<=> (Unul-dintre @Membri ?Clasa)

(Pentru-toate (?Specimen)

(<=> (Specimenul-lui ?Specimen ?Clasa)

(Membru ? Specimen (Setde @Membri)))))

Relaţia ESTE-SUBCLASA-LUI

Clasa C, este o subclasă a unei clase părinte P, dacă şi numai dacă orice specimen al lui C este în acelaşi timp şi specimen al lui P. O clasă poate avea multiple superclase şi subclase. Relaţia *este-subclasa-lui* este tranzitivă:

dacă(este-subclasa-lui C1 C2) şi (este-subclasa-lui C2 C3) atunci (este-subclasa-lui C1 C3).

Sistemele centrate pe obiect fac distincţie, adesea, între relaţia *este-subclasa-lui* adăugată şi relaţia care este dedusă.

Exemplu:

(*este-subclasa-lui* C1 C3) poate fi dedusă din relaţiile (*este-subclasa-lui* C1 C2) şi (*este-subclasa-lui* C2 C3).

relaţii înrudite: *este-specimen-al-lui, relaţie, relaţie-tranzitivă*

Aritate: 2

Domeniu: clasă

Magnitudine: clasă

Axiome:

```
(<=> (Este-subclasa-lui ?Clasa-copil ?Clasa-părinte)
   ( şi (Clasa ?Clasa-părinte)
      (Clasa ?Clasa-Copil)
      (Pentru-toate (?Specimen)
         (=> (Este-specimenul-lui?Specimen ?Clasa-copil)
            (Este-specimenul-lui?Specimen?Clasa-părinte) ))))
```

Relaţia ESTE-SUPERCLASA-LUI

Este-superclasa-lui este inversa relaţiei *este-subclasa-lui*. Această clasă nu este logic necesară, dar este utilă.

Aritate: 2

Inversa: Subclasa-lui

Relaţia ESTE-SUBRELATIA-LUI

O *relaţie* R este o *subrelaţie* a *relaţiei* R' dacă, R este un subset a lui R'. Fiecare ramură a lui R este şi ramură a lui R'. Dacă R are argumentele arg_1, arg_2, ... arg_n, atunci şi R' are aceleaşi argumente. Relaţia şi subrelaţiile sale trebuie să aibă aceeaşi aritate, care ar putea să nu fie definită.

Aritate: 2

Domeniu: Relaţie

Magnitudine: Relaţie

Axiome:

```
(<=> (este-subrelaţia-lui?Relaţia-copil ?Relaţia-părinte)
   ( şi  (Relaţia ?Relaţia-copil)
```

(Relaţia ?Relaţia-părinte)

(pentru-toate (?Ramura)

(=> (Membru ?Ramura ?Relaţia-copil)

(Membru ?Ramura ?Relaţia-părinte)))))

Relaţia ESTE-SPECIMEN-DIRECT-AL-LUI

Un individual i este un *este-specimen-direct-al-lui* al clasei C, dacă i este *instanţa-lui* C şi nu există nici o altă subclasă a lui C, definită în ontologia curentă, astfel încât, i să fie şi specimen al acelei subclase. O asemenea clasă este o clasă "minimală" sau cea mai specifică clasă a lui i. Clasa directă nu este în mod necesar unică, un individual putând avea mai multe clase specifice. Această relaţie este indexabilă, adevărul ei depinde mai mult de conţinutul bazei de cunoştinţe, decât de lumea exterioară.

Distincţia dintre *este-specimenul-lui* şi *este-specimen-direct-al-lui* nu este aceeaşi ca şi relaţia dintre asertarea directă a *este-specimenul-lui* şi un sistem dedus. Înţelesul relaţiei *este-specimenul-lui*, respectiv *este-specimen-direct-al-lui*, cât şi al altor relaţii de nivel obiect, din punctul de vedere al bazei de cunoştinţe, este independent de faptul că, ele sunt asertate direct sau sunt deduse.

Relaţia *este-specimenul-lui* este sensibil diferită de *este-specimen-direct-al-lui*. O ontologie evolutivă are nevoie de acestă diferenţă pentru a-şi elimina inconsistenţa. Atunci când se efectuează o asertare la baza de cunoştinţe de tipul (*este-specimenul-lui* i C), aceasta înseamnă acelaşi lucru cu (*este-specimen-direct-al-lui* i C), cu distincţia că sistemul ontologic creează un pointer între specimen şi colecţia din care face parte acesta. Această legătură face posibil ca mai târziu să se poată defini o subclasă C_*sub*, a lui C, astfel încât, asertarea (*este-specimenul-lui* i C), să devină (*este-specimenul-lui* i C_sub). Cele două aserţiuni sunt consistente şi contribuie la dezvoltarea bazei de cunoştinţe, menţinând-o într-o formă canonică.

Aritate: 2

Domeniu: Este-Subrelaţia-lui, Este-Specimenul-lui

Relaţiile prezentate mai sus se pot constitui într-un nucleu capabil să genereze principalele atribute ale inteligentei umane. Unele dintre acestea sunt relaţii epistemice şi au menirea de a construi clase din ce în ce mai generale, deci de a creea o cunoaştere proprie, metafizică. Altele sunt relaţii compoziţionale şi au rolul de a stabili apartenenţele corecte dintre clase şi specimene. Pentru a fi complet, adică pentru a putea simula mai fidel comportamentul inteligenţei umane, acest grup de relaţii trebuie completat şi cu relaţii cauzale şi modale.

3.1.6. Reprezentarea cunoaşterii prin Fuzzy-logic

Incercările convenţionale de reprezentare a cunoaşterii prin reţele semantice, cadre, calculul predicatelor şi PROLOG se bazează pe logica bivalentă. Un impediment serios al acestor încercări este inabilitatea de a reprezenta incertitudinea şi imprecizia. Reprezentarea convenţională nu oferă un model adecvat pentru metodele de raţionament, care sunt mai degrabă aproximative decât exacte. Majoritatea metodelor raţionamentului uman şi toate metodele raţionamentului comun se înscriu în această categorie.

Fuzzy logic, care poate fi privită ca o extensie a sistemelor logice clasice, oferă un cadru conceptual, care abordează problema reprezentării cunoaşterii, într-un mediu al incertitudinii şi impreciziei. Esenţa reprezentării în Fuzzy logic se bazează pe semanticile test-scor. In cadrul acestor semantici o

propoziţie este interpretată ca un sistem de restricţii elastice, iar raţionamentul este privit ca o propagare a restricţiilor elastice.

Câteva trăsături caracteristice ale Fuzzy logic:

- In Fuzzy logic raţionamentul exact este privit ca un caz limită al raţionamentului aproximativ;
- In Fuzzy logic orice lucru este exprimat gradual;
- Orice sistem logic poate fi fuzzyficat;
- In Fuzzy logic cunoaşterea este interpretată ca o colecţie de restricţii Fuzzy elastice sau de echivalenţă pe o colecţie de variabile;
- Deducţia este privită ca un proces de propagare al restricţiilor elastice.

Diferenţele dintre Fuzzy logic şi sistemele logice tradiţionale.

SISTEM LOGIC BIVALENT	SISTEM LOGIC MULTIVALENT
ADEVARUL	
In sistemele cu logică bivalentă, adevărul poate lua doar două valori: adevărat sau fals.	In sistemele multivaloare, valoarea de adevăr a unei propoziţii poate fi un element aparţinând unui set finit, aparţinând unui interval [0,1], sau algebrei boolene. In Fuzzy logic valoarea de adevăr a unei propoziţii poate fi un subset Fuzzy al oricărui set parţial ordonat, dar de obicei se impune să fie un subset Fuzzy al intervalului [0,1] sau mai simplu, un punct în acest interval. Valorile aşa numitului "adevăr lingvistic" - adevărat, foarte adevărat, nu tocmai adevărat - sunt interpretate ca etichete ale subsetului Fuzzy pe intervalul unitar.
PREDICATELE	
In sistemele bivalente predicatele sunt categorice şi precise: *mai mare decât; mai mic decât.*	In Fuzzy logic predicatele sunt vagi: mult mai mare decât, curând etc. In limbajul natural marea majoritate a predicatelor sunt vagi şi nu categorice.

MODIFICATORII

In sistemele clasice se foloseşte numai modificatorul "NU".

In Fuzzy logic există o mare varietate de modificatori: foarte, mult mai mult, mult mai puţin, extrem, etc. Asemenea modificatori joacă un rol esenţial în generarea valorilor variabilelor lingvistice: foarte tânăr, mai mult sau mai putin tânăr, nu tocmai tânăr, etc.

CUANTIFICATORII

In sistemele clasice există numai doi cuantificatori: universal şi existenţial.

Fuzzy logic admite o largă varietate de cuantificatori cum ar fi: puţin, mai multe, de obicei, mai mult, întotdeauna, frecvent, etc. Cuantificatorul este interpretat ca un număr Fuzzy sau ca o proporţie Fuzzy.

PROBABILITATILE

In sistemele clasice probabilitatea este evaluată numeric sau pe un interval.

In Fuzzy logic probabilităţile pot fi interpretate ca numere Fuzzy, care pot fi manipulate în aritmetica Fuzzy. De remarcat că, există o interschimbabilitate între probabilităţile Fuzzy şi cuantificatorii Fuzzy sau, mai general, între măsurile Fuzzy. Astfel, o propoziţie care conţine etichetele probabilităţilor Fuzzy poate fi exprimată într-o formă echivalentă cu una care conţine cuantificatori Fuzzy.

POSIBILITATILE

Se evaluează numeric, la fel ca şi probabilităţile

Conceptul de posibilitate în Fuzzy logic este mai mult gradual decât bivalent. Mai mult decât în cazul probabilităţilor, posibilităţile pot fi tratate ca variabile lingvistice cu valori ca: la fel de posibil, puţin posibil, aproape imposibil, etc.Un concept suplimentar în Fuzzy logic este distribuţia de posibilitate care constituie un set cu toate valorile posibile p_x ale variabilei X.

Noţiunile fundamentale referitoare la mulţimile Fuzzy şi la operaţiile cu acestea sunt prezentate în Anexa 1.

Variabile lingvistice

O variabilă lingvistică este o entitate ale cărei valori nu sunt numere ci cuvinte, sau propoziţii ale limbajului natural, sau propoziţii logice ale limbajului artificial. O astfel de variabilă este definită de un quintuplu format din: x, T(x), σ, U, M.

- x este numele variabilei lingvistice;
- T(x) este mulţimea nevagă a numelor valorilor variabilei lingvistice, adică un termen al variabilei lingvistice A;
- σ este operatorul (gramatica), care pe baza unor reguli sintactice permite generarea termenilor variabilei lingvistice;
- U este universul de discuţie aferent variabilei lingvistice;
- u este variabilă de bază;
- M este operator semantic care este folosit pentru fiecare ataşare a unei submulţimi vagi în U, fiecărui termen T, al variabilelor lingvistice.

Reprezentarea cunoaşterii şi a înţelesului.

Reprezentarea înţelesului unei propoziţii p, prin folosirea semanticilor de tip test-scor, se petrece în următoarele etape:

1. Identificare variabilelor $X_1,.....,X_n$ ale căror valori sunt impuse de propoziţie. Acestea sunt în mod uzual implicite în p.
2. Identificarea restricţiilor $C_1,.....,C_m$ care sunt introduse de către p.
3. Caracterizarea fiecărei restricţii, C_i , prin descrierea unei proceduri test, care asociază cu C_i un test-scor τ_I, care reprezintă gradul de satisfacere a lui C_i. Uzual τ_I este exprimat de un număr cuprins în intervalul [0,1]. In sens mai general, acest test-scor poate fi distribuţia de probabilitate/posibilitate pe intervalul unitar.
4. Agregarea scorurilor parţiale $\tau_1,.....,\tau_m$ într-un număr mic de test-scoruri $\tau_1,.....,\tau_k$, care este reprezentat de un vector test-scor, $\tau = (\tau_1,....,\tau_k)$. In cele mai multe cazuri k = 1 şi avem de-a face cu un scalar.

In semantica test-scor, testarea restricţiilor introduse prin p este realizată de o colecţie de relaţii Fuzzy, care constituie o bază de date explicativă ED (explanatory database).

Modul Fuzzy de decizie multi-atribut.

Există mai multe metode de luare a deciziilor multi-atribut de tip Fuzzy; unele sunt metode aditive, altele sunt procese analitice ierarhice. O metodă nouă de decizie Fuzzy este propusă în lucrarea [21].

$$\text{Fie } \mathbf{S} = \{(\mathbf{x}, \mathbf{u_a(x)}) \mid \mathbf{x} \in \mathbf{U}\}, \text{ un set Fuzzy,} \qquad \mathbf{(3.24)}$$

unde **U** este un set ordinar de obiecte numit univers. Setul Fuzzy este caracterizat de o funcţie asociată $u_a(x)$, care ataşează fiecărui obiect din U, un număr real din intervalul [0, 1]. Numărul trapezoidal Fuzzy este un tip de funcţie asociată, des folosită în mediile Fuzzy, pentru avantajele pe care le are în achiziţia şi procesarea datelor. Numărul trapezoidal Fuzzy este arătat în Fig.3.3 şi este reprezentat de funcţia **M = (a, b, c, d)**.

Valoarea funcţiei creşte liniar sau progresiv între a şi b şi descreşte între c şi d. Principiul fundamental al metodei lexicografice este acela de a compara toate dispozitivele, după

importanţa atributelor. Inginerul de cunoaştere este solicitat să culeagă toate informaţiile disponibile, pentru a atribui valori relative tuturor criteriilor, şi să determine raţia Fuzzy pentru fiecare criteriu. Fazele metodei lexicografice ale modulului de luare a deciziilor sunt:

1. Stabilirea criteriului pe care îl folosesc experţii umani pentru categorisirea dispozitivelor;
2. Stabilirea gradului de importanţă pentru acel criteriu, astfel încât, rangul acelui criteriu să poată fi identificat;
3. Definirea câtorva termeni lingvistici, care trebuie să fie capabili să distingă gradele de importanţă relativă ale criteriului (foarte mare, mare, mediu, mic, foarte mic, etc.);
4. Definirea funcţiei asociate Fuzzy pentru toţi termenii ligvistici, respectând fiecare criteriu;
5. Normalizarea termenilor lingvistici;
6. Atribuirea celor mai potriviţi termeni lingvistici fiecărui criteriu, ţinînd seama de toate dispozitivele de la acel nivel (operaţia se începe de la cel mai înalt nivel al arborelui de decizie);
7. Criteriul de rangul cel mai înalt este luat în considerare primul, iar numărul Fuzzy este convertit într-un număr caracteristic;
8. Dacă mai multe criterii au acelaşi rang, se iau în seamă valorile funcţiei asociate şi se sare la poziţia 7;
9. Dacă mai multe dispozitive au acelaşi număr caracteristic, se ia în considerare criteriul cu rangul al doilea şi se compară dispozitivele de la poziţiile 7 şi 8;
10. Se procedează în acest mod, până când se determină prioritătile tuturor dispozitivelor;
11. Se coboară la nivelul următor al arborelui de diagnoză şi se reia procesul de la punctul 6. Acest proces de clasificare continuă, până la atingerea celui mai scăzut nivel al arborelui.

Pentru parcurgerea acestui proces sunt necesare mai multe formule şi un set de notaţii, după cum urmează:

$\mathbf{X} = \{ x_i \}$, $i = 1,\ldots,n$ setul de dispozitive.

$\mathbf{G} = \{ g_i \}$, $i = 1,\ldots,m$ setul de criterii.

V_{jk} = termenul lingvistic k al criteriului j, $k = 1,\ldots,p$.

R_{ij} = raţia Fuzzy i, a criteriului j.

W_j = mărimea criteriului j.

$U_{Rij}(r_{ij})$ = funcţia asociată pentru raţia dispozitivului i, a criteriului j.

$U_{vjk}(v_{jk})$ = funcţia asociată pentru termenul lingvistic k, al criteriului j.

Normalizarea.

Se folosesc două proceduri de normalizare, una după criteriul factorului de beneficiu (3.25) şi cealaltă după criteriul factorului cost (3.26). Cele două formule sunt:

$$V'_{jk} = \left(\frac{a - a^*}{d^* - a^*}, \frac{b - a^*}{d^* - a^*}, \frac{c - a^*}{d^* - a^*}, \frac{d - a^*}{d^* - a^*} \right) \qquad (3.25)$$

$$V'_{jk} = \left(\frac{d^* - d}{d^* - a^*}, \frac{d^* - c}{d^* - a^*}, \frac{d^* - b}{d^* - a^*}, \frac{d^* - a}{d^* - a^*} \right) \qquad (3.26)$$

unde a^* este valoarea cea mai mică în V_{jk} , k = 1, . . . , p;

d^* este valoarea cea mai mare în V_{jk} , k = 1, . . . , p.

Combinaţia.

Formula de aditivare a două numere Fuzzy este următoarea:

dacă $R_{ip} = (a_1, b_1, c_1, d_1)$, i = 1, . . . , n şi $R_{iq} = (a_2, b_2, c_2, d_2)$, i = 1, . . . , n

atunci $R_{ip} + R_{iq} = (a_1 + a_2, b_1 + b_2, c_1 + c_2, d_1 + d_2)$, i = 1, . . . , n.

Conversia.

Pentru un număr Fuzzy trapezoidal, M = (a, b, c, d), numărul caracteristic **C**, este dat de formula (3.27).

$$\mathbf{C} = \frac{\mathbf{1}}{\mathbf{2}}\left(\frac{\mathbf{d}}{\mathbf{1 - c + d}} + \frac{\mathbf{b}}{\mathbf{b - a + 1}}\right) \qquad \textbf{(3.27)}$$

Factori de încredere.

Increderea, într-o soluţie oferită de un sistem de decizie automată, se poate alcătui din încrederea în cunoştinţele folosite la elaborarea deciziei, combinată cu încrederea în solicitarea utilizatorului. Atingerea scopului se face printr-o decizie completă, adică prin parcurgerea completă a unei ramuri, dintr-un arbore de decizie şi formularea unei soluţii rafinate. Actul de parcurgere al arborelui de decizie este însoţit de decizii temporare(parţiale) nerafinate, care pot sau nu, să fie aduse la cunoştinţa utilizatorului.

Decizia $\mathbf{D} = \{\mathbf{D}_1 \subset \mathbf{D}_2 \subset \ldots\ldots \subset \mathbf{D}_{n-1}\}$

Factorul de încredere $\mathbf{Cf} = \mathbf{Min}\ \{\ \mathbf{Cf}_i\ \}$

Factorul de încredere local $\mathbf{Cf}_i = \mathbf{Cfd}_i * \mathbf{Cfu}_i$

unde: Cfd - factorul de încredere în date;

Cfu - factorul de încredere în utilizator;

n - numărul de nivele de decizie.

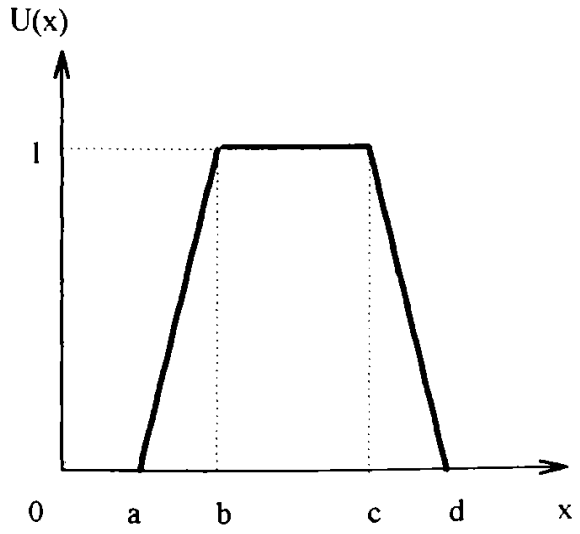


Fig.3.3

Pentru evaluarea factorilor de încredere se propune relaţia (4).

$$\mathbf{Cf} = \mathbf{1} - \frac{\mathbf{K}_1}{\mathbf{N}_r + \mathbf{K}_2^{N_r}} \qquad \textbf{(3.28)}$$

unde:

- $N_r \equiv A_v$, se numesc argumente valide şi folosesc la calculul lui Cfd, la introducerea sau asimilarea datelor în baza de cunoştinţe;

- $N_r \equiv R_v$, se numesc răspunsuri valide şi folosesc la calculul lui Cfu, în conversaţia cu utilizatorul.

3.2. CRITERII PENTRU ALEGEREA UNEI TRANSMISII MECANICE

Considerentele organizatorice ale concepţiei şi proiectării au făcut ca, atât activităţile inginerilor, cât şi specializările acestora, să fie împărţite în mai multe compartimente. Ţinând cont de această structurare şi de cerinţele generale de calitate impuse produselor muncii inginereşti, am grupat criteriile de selecţie astfel:

F criterii de performanţă funcţională;

C criterii constructive;

T criterii tehnologice;

I criterii de întreţinere;

M criterii conjuncturale (de modă).

Observaţii:

1. Toate criteriile sunt criterii de performanţă, deşi aceasta s-a specificat numai la prima grupă.
2. Caracteristicile funcţionale limită se constituie în criterii de performanţă funcţională, pentru operaţia de selecţie a transmisiei cea mai adecvată unei aplicaţii date.

O prezentare sintetică a principalelor caracteristici funcţionale ale transmisiilor mecanice elementare, care se constituie în criterii de performanţă funcţională, este prezentată în tabelul T6.1 [78],[79],[96].

F1. Randamentul transmisiei. Exprimarea cantitativă indirectă a pierderilor energetice cu ajutorul randamentului este criteriu de performanţă important, pentru toate transmisiile. In cazul trasmiterii puterilor mari pe unitate, când pierderile energetice devin costisitoare şi perturbă buna funcţionare, atât a transmisiei, cât şi a agregatului în care este plasată, valoarea randamentului devine hotărâtoare.

$$\eta = \frac{P_2}{P_1} \quad \textbf{sau} \ \forall \ \Delta t \quad \eta = \frac{W_2}{W_1} \tag{3.29}$$

Comportarea energetică a transmisiei, exprimată prin randament, este dependentă de o serie de condiţii de funcţionare, cum ar fi: gradul de încărcare, starea de uzură a lagărelor, condiţiile de ungere, etc. Randamentul poate fi luat în considerare ca randament global al transmisiei, cu includerea pierderilor din lagăre [47], [48]. In acest caz, mulţimea factorilor care îl influenţează este destul de mare, iar cuantificarea influenţei fiecăruia este dificilă. Transmisiile solicitate de aplicaţiile practice sunt alcătuite dintr-una sau mai multe trepte formate din transmisii elementare. Conectarea transmisiilor de bază, elementare, este realizată în majoritatea cazurilor prin înserierea mai multor transmisii elementare. In acest caz, randamentul global al transmisiei se poate obţine cu relaţia:

$$\eta = \sum_{i=1}^{n} \eta_i \qquad \eta = \prod_{i=1}^{n} \eta_i \qquad \eta = \sum_{i=1}^{k} \prod_{i=k}^{n} \eta_i \quad \mathbf{i, n} \in \mathbf{N} \tag{3.30}$$

F2. Puterea. Puterea transmisă, în sine, nupoate fi considerată ca un criteriu de selecţie, pentru tipul transmisiei. Pentru a caracteriza, din punct de vedere energetic, un anume tip de transmisie, sunt necesare atât valorile limită absolute, cât şi valorile medii pentru următoarele mărimile relative:

- puterea absolută maximă pe transmisia elementară [KW] (vezi T3.1);

- puterea maximă relativă la masa transmisiei elementare [KW/Kg];
- puterea maximă relativă la volumul transmisiei elementare[KW/m^3];
- puterea maximă relativă la costul transmisiei elementare [KW/$]
- factorul de suprasarcină P_{vf} / P_n .

Tot legată de puterea transmisă este şi dinamica celor două maşini, cea motoare, respectiv de lucru, între care se interpune transmisia în cauză. Variaţia de putere consumată de maşina de lucru, cât şi variaţia de putere furnizată de maşina motoare, sunt exprimate global, prin factorul dinamic.

Tipul transmisiei	*Randamentul transmisiei*	*Puterea maximă [KW]*	*Puterea minimă [KW]*	*Turaţia maximă [rot/min]*	*Viteza periferică [m/s]*	*Raportul de transmitere i_{max}*
Transmisie prin curea trapezoidală clasică	0.96	3000	0.37	10000	50	1:12
Transmisie prin curea trapezoidală îngustă	0.96	3000	0.37	10000	50	1:12
Transmisie prin curea trapezoidală (flanc nud)	0.96	3000	0.37	10000	60	1:12
Transmisie prin curea trapezoidală lată	0.95	70	0.37	10000	25	1:10
Transmisie prin curea lată multistrat	0.98	5000	0.1	130000	200	1:12
Transmisie prin curea trapezoidală multiplă	0.96	1000	0.1	12500	60	1:35
Transmisie prin curea rotundă	0.96	1	0.001	10000	50	1:12
Transmisie prin curea sincronă, profil trapez	0.98	1000	0.1	20000	80	1:10
Transmisie prin curea sincronă, profil rotund	0.98	1000	0.1	20000	80	1:10
Transmisie prin lanţ cu role şi zale	0.9	1000	0.1	3000	20	1:10
Transmisie prin lanţ cu eclise	0.9	1000	0.1	5000	25	1:10
Transmisie prin roţi de fricţiune	0.95	100	0.001	300000	200	1:40
Transmisie prin angrenaj cilindric cu dinţi drepţi	0.95	1000	0.0001	3000	5	1:8
Transmisie prin angrenaj cilindric cu dinţi înclinaţi	0.98	3000	0.01	40000	50	1:10
Transmisie prin angrenaj cilindric cu dinţi în V	0.98	10000	1	10000	50	1:8
Transmisie prin angrenaj conic cu dinţi drepţi	0.95	500	0.1	3000	1	1:6
Transmisie prin angrenaj conic cu dinţi curbi şi înclinaţi	0.98	1000	0.1	15000	10	1:8
Transmisie prin angrenaj melcat	0.75	500	0.001	3000	5	1:60

T3.1

Acest parametru devine foarte important la proiectare, căci el contribuie esenţial la subdimensionarea, respectiv supradimensionarea transmisiei. Comparaţia între două transmisii date, din punctul de vedere al puterii transmise, este temeinică, dacă se face la o valoare unificată a acestui factor dinamic.

Dificultatea stabilirii puterii de calcul provine şi de la faptul că, diagrama puterii este foarte diferită, de la un consumator la altul. In fig.3.4 se arată diagramele de putere a doi consumatori, care consumă aceeaşi putere medie, dar prezintă valori de vârf foarte diferite. Includerea acestui aspect, în stabilirea puterii de calcul, este o operaţie dificilă şi controversată. Factorul dinamic nu caracterizează transmisia şi deci, nu poate fi folosit ca şi criteriu de alegere al tipului de transmisie. Pentru selecţie rămâne doar valoarea maximă absolută a puterii.

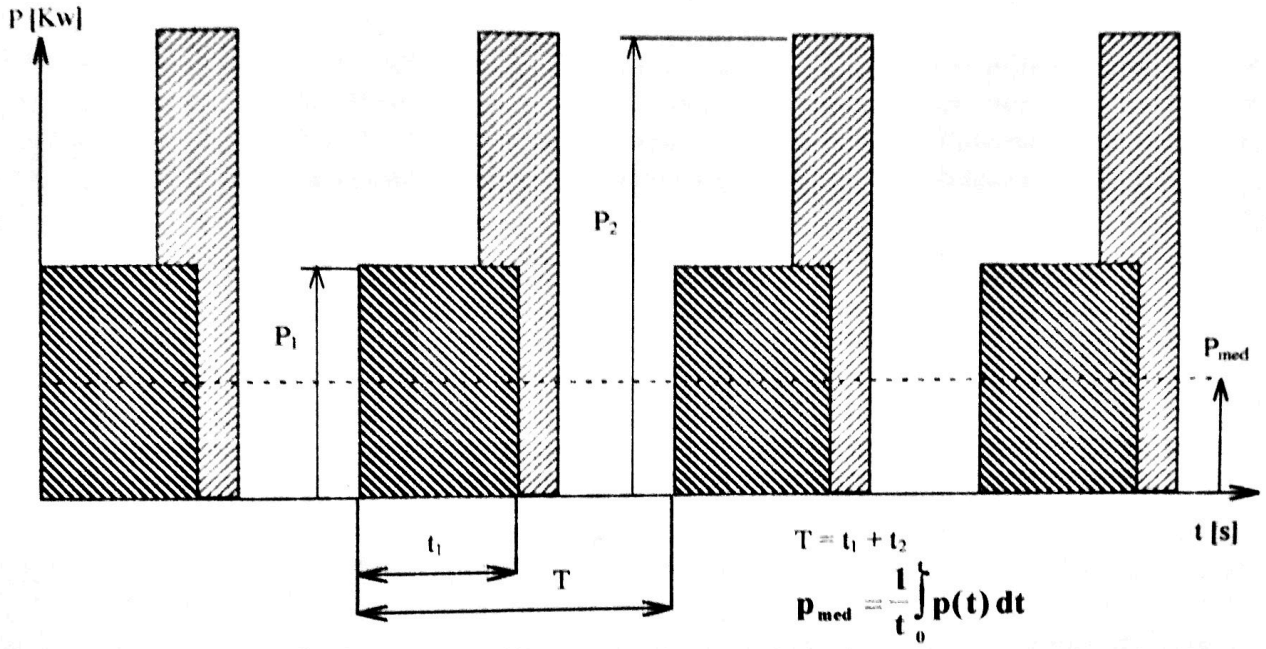


Fig.3.4

F3. Fiabilitatea şi mentenanţa. Cei doi parametrii se exprimă prin indicatori de fiabilitate[49]. Aceşti indicatori devin criterii de selecţie, în măsura în care există date statistice necesare evaluării lor. Fiabilitatea generală a transmisiei se compune din fiabilitatea lagărelor, a etanşărilor, a arborilor, din fiabilitatea curelelor, a lanţurilor de transmisie, a angrenajelor. Principalele tipuri de uzări din transmisiile mecanice sunt: adeziunea, abraziunea, oboseala, coroziunea, impactul, deformarea la rece, cojirea, brinelarea, deformarea la cald şi pătarea.

F4. Turaţia. Viteza unghiulară. Viteza periferică. Turaţia şi viteza unghiulară exprimă în două forme aceeaşi mărime. Unităţile de măsură sunt [rot/min], respectiv [rad/sec] Viteza periferică este o mărime cinematică cu valoare restrictivă, în special, în cazul transmisiilor care funcţionează prin angrenare şi la turaţii mari, respectiv în [m/sec]. Turaţia şi viteza periferică, ca şi criterii de selecţie, se pot folosi prin valorile maxime absolute. Valorile de referinţă se găsesc în tabelul T3.1.

F5. Precizia cinematică. Precizia cinematică se referă la capacitatea transmisiei de a menţine o relaţie strictă, între legea de mişcare a elementului motor şi a elementului condus. Funcţia de transfer, numită raport de transmitere şi notată cu i, are relaţia de definiţie:

$$\mathbf{i} = \frac{\omega_1}{\omega_2} \tag{3.31}$$

sau

$$\mathbf{i} = \frac{\varphi_1}{\varphi_2} \tag{3.32}$$

Admiţînd că transmisiile mecanice nu sunt generatoare propriu-zise de legi de mişcare, pe durata regimului stabilizat de funcţionare, putem defini precizia cinematică absolută astfel:

$$\forall \varphi_0 \in [0, 2\pi] \quad \wedge \quad \mathbf{i} = \frac{\mathbf{d}\varphi_1}{\mathbf{d}\varphi_2} \quad \Rightarrow \mathbf{i} = \mathbf{cons\,tan\,t} \tag{3.33}$$

Relaţia de definiţie (3.33) este valabilă şi pentru transmisiile mecanice prin frecare, la care apare alunecarea elastică. Pentru a da mai multă rigoare definiţiei, trebuie introdusă o restricţie suplimentară, referitoare la regimul de încărcare al transmisiei. Puterea instantanee de funcţionare influenţează valoarea alunecării. Definiţia preciziei se completează astfel:

$$\forall \varphi_0 \in [0, 2\pi] \wedge \mathbf{P} \in [0, \mathbf{P}_{max}] \wedge \mathbf{i} = \frac{\mathbf{d}\varphi_1}{\mathbf{d}\varphi_2} \quad \Rightarrow \mathbf{i} = \mathbf{cons\,tan\,t} \tag{3.34}$$

Relaţia de definiţie (3.34) are deficienţa că nu prinde cerinţa rigidităţii legăturii cinematice pe termen lung, cerută de anumite transmisii, cu funcţie cinematică importantă. Definiţia următoare elimină deficienţa semnalată mai sus:

$$\mathbf{i}_{mediu} = \frac{\sum_{i=1}^{k} \Delta_i \varphi_1}{\sum_{i=1}^{k} \Delta_i \varphi_2} \quad \wedge \quad \forall \mathbf{k} \in \mathbf{N} - \{0, 1\} \Rightarrow \mathbf{i}_{mediu} = \mathbf{cons\,tan\,t} \tag{3.35}$$

Relaţia de definiţie (3.35) nu conţine restricţia de constanţă a raportului de transmitere instantaneu. O definiţie completă a preciziei cinematice este dată de relaţia:

$$\begin{cases} \forall \mathbf{k} \in \mathbf{N} - \{0, 1\} \quad \wedge \quad \mathbf{i}_{mediu} = \dfrac{\sum_{i=1}^{k} \Delta_i \varphi_1}{\sum_{i=1}^{k} \Delta_i \varphi_2} \quad \Rightarrow \quad \mathbf{i}_{mediu} = \mathbf{cons\,tan\,t} \\ \forall \varphi_0 \in [0, 2\pi] \quad \wedge \quad \mathbf{i} = \dfrac{\mathbf{d}\varphi_1}{\mathbf{d}\varphi_2} \quad \Rightarrow \quad \mathbf{i} = \mathbf{cons\,tan\,t} \end{cases} \tag{3.36}$$

Indeplinirea graduală a relaţiilor (3.34) - (3.36) constituie un criteriu de apreciere al diferitelor transmisii, în ceea ce priveşte precizia cinematică. Gradele de îndeplinire ale relaţiilor şi simbolizarea acestora sunt prezentate în tabelul T3.2.

	Definiţia îndeplinită	**Valoarea atributului**	**Simbol**	**Exemple**
1	Nici una	fără precizie	- pc	transmisia prin curea trapezoidală
2	6.6	precizie instantanee	+ pc_1	transmisia prin roţi de fricţiune
3	6.7	precizie de reversare	+ pc_2	transmisia prin curea sincronă
4	6.8	precizie globală	+ pc_3	transmisia prin angrenaje cu dinţi înclinaţi, respectiv în V

T3.2

F6. Domeniul raportului de transmitere. Domeniul raportului de transmitere poate fi o mărime caracteristică, numai pentru transmisia elementară. Ca şi criteriu de selecţie poate fi folosită valoarea maximă absolută, prezentată în tabelul T3.1.

F7. Flexibilitatea raportului de transmitere. Lipsa flexibilităţii este o caracteristică a transmisiilor cu raport de transmitere fix. Flexibilitatea există la acele tipuri de transmisii, la care raportul de transmitere este reglabil în trepte, respectiv reglabil continuu. Este importantă corelarea flexibilităţii raportului de transmitere, cu precizia cinematică a transmisiei. Parametrul consacrat este gama de reglare a raportului de transmitere:

$$\Delta_m = \frac{\omega_{2\,max}}{\omega_{2\,min}} \qquad (3.37)$$

Tipul transmisiei	*Gama de reglare*	*Precizia cinematică*	*Temperatura minimă [° C]*	*Temperatura maximă [° C]*	*Distanţa dintre axe*	*Poziţia spaţială a roţilor*
Transmisie prin curea trapezoidală clasică	1.6	- pc	-30	80	mare	coplanară
Transmisie prin curea trapezoidală înaltă	1	- pc	-30	80	mare	coplanară
Transmisie prin curea trapezoidală anizotropă	1	- pc	-30	80	mare	coplanară
Transmisie prin curea trapezoidală lată	9	- pc	-30	80	medie	coplanară
Transmisie prin curea lată multistrat	1.6	- pc	-30	150	foarte mare	coplanară+ spaţială
Transmisie prin curea trapezoidală multiplă	1	- pc	-30	80	mare	coplanară
Transmisie prin curea rotundă	1	- pc	-30	100	mare	coplanară+ spaţială
Transmisie prin curea sincronă profil trapez	1	+ pc_2	-30	100	mare	coplanară+ spaţială
Transmisie prin curea sincronă profil rotund	1	+ pc_2	0	80	medie	coplanară
Transmisie prin lanţ cu role şi zale	1	+ pc_2	-50	150	mare	coplanară
Transmisie prin lanţ cu eclise	1	+ pc_2	-50	150	mare	coplanară
Transmisie prin roţi de fricţiune	9	+ pc_1	-50	150	mică	coplanară+ spaţială
Transmisie prin angrenaj cilindric cu dinţi drepţi	1	+ pc_2	-50	150	mică	coplanară
Transmisie prin angrenaj cilindric cu dinţi înclinaţi	1	+ pc_3	-50	150	mică	coplanară
Transmisie prin angrenaj cilindric cu dinţi în V	1	+ pc_3	-50	150	mică	coplanară
Transmisie prin angrenaj conic cu dinţi drepţi	1	+ pc_2	-50	150	mică	coplanară ortogonală
Transmisie prin angrenaj conic cu dinţi curbi şi înclinaţi	1	+ pc_3	-50	150	mică	coplanară ortogonală
Transmisie prin angrenaj melcat	1	+ pc_3	-50	100	mică	spaţială

T3.3

F8. Capabilitatea transmiterii mişcării la distanţă. Se referă, atât la distanţele dintre axe, cât şi la poziţiile spaţiale ale axelor celor două maşini. Această caracteristică poate să fie calitate sau deficienţă, în funcţie de cerinţele aplicaţiei concrete. Evaluarea se prezintă în tabelul T3.3.

F9. Nivelul de zgomot creat de funcţionarea transmisiei. Are o exprimare cantitativă în [dB], în condiţii standard de măsură. Aprecierea calitativă a diferitelor tipuri de transmisii este prezentată în tabelul T3.4.

Atribut	Tip de transmisie
Foarte silenţioase	Transmisii cu curele late. transmisii cu curele rotunde
Silenţioase	Transmisii cu curele trapezoidale. transmisii cu curele sincrone. transmisii cu roţi de fricţiune
Zgomotoase	Transmisii cu lanţ cu eclise. transmisii cu angrenaje
Foarte zgomotoase	Transmisii cu lanţ cu role şi zale

T3.4

F10. Nivelul de vibraţii produs de funcţionarea transmisiei. Criteriul se referă la vibraţiile provenite din funcţionarea transmisiei. Aceste vibraţii se suprapun peste mişcarea transmisă şi perturbă funţionarea maşinii de lucru. Grupele de transmisii din tabelul T3.4. sunt valabile şi pentru criteriul nivelului vibraţiilor. Nivelul vibraţiilor constituie o restricţie impusă de maşina de lucru. In cazul maşinilor prelucrătoare de precizie, nivelul vibraţiilor introduse în sistemul elastic de către transmisia mecanică, devine un argument esenţial, în alegerea tipului de transmisie. Efectele vibraţiilor sunt grupate în efecte asupra omului, efecte asupra maşinilor şi efecte asupra clădirilor. Conform cu precizările din [75] şi [97] treptele de percepere ale vibraţiilor, de către om, sunt cele redate în tabelul T3.5. K este un coeficient adimensional conform cu VDI Richtlinien 2057.

Coeficientul de percepere K	Treapta	Modul de percepere
> 0.1	A	Imperceptibil
0.1 - 0.25	B	Abia perceptibil
0.25 - 0.63	C	Perceptibil
0.63 - 1.4	D	Bine perceptibil
1.4 - 4.0	E	Puternic perceptibil
4.0 - 10	F	Perceptibil foarte puternic
10 - 25	G	
25 - 63	H	
> 63	I	

T3.5

Identificarea surselor de vibraţii se face printr-o corelare a frecvenţelor maximelor din spectrograme, cu parametri funcţionali şi constructivi ai transmisiilor. Dintre sursele de vibraţii de frecvenţă joasă, la o transmisie care este antrenată de o maşină cu turaţia n [rot/min], se menţionează următoarele:

- dezechilibrul masic	$f_1 = n / 60$ [Hz];
- excentricitatea fusurilor arborilor	f_1 ;
- jocurile din lagăre	$2f_1$;
- curelele de transmisie defecte	f_1 , $2f_1$, $3f_1$, $4f_1$;
- motoarele electrice	f_1 ;
- mecanismele bielă - manivelă	f_1 ;
- instabilitatea fusului într-un lagăr de alunecare	$0.5\ f_1$;
- rulmenţii cu inel exterior fix	f_1 ; $f_{bilă}$; $f_{colivie}$;
- roţile dinţate	$f = f_1\ z_1$; $f_2 = f_{arbore} / z_2$.

In afara frecvenţelor f_1, f_2, şi f_{arbore} spectrul vibraţiilor produse de un angrenaj mai cuprinde şi o gamă largă de armonici ale acestora, rezultate prin însumare algebrică.

F11. Temperatura mediului. Factorul de mediu *temperatura*, este restrictiv pentru toate transmisiile, mai ales pentru cele care conţin componente nemetalice. Acest factor devine criteriu de selecţie, în condiţiile cuantificării sale, pentru fiecare tip de transmisie. Folosirea sa este relevantă, în cadrul creeat de logica Fuzzy. In acest scop, este necesară definirea variabilelor lingvistice aferente. Reprezentarea grafică a acestora este dată în Fig.3.5, iar relaţiile de definiţie sunt scrise în continuare.

Funcţia de apartenenţă pentru temperatura extrem de mică a fost centrată pe valoarea de -50° C, valoare apropiată de recordul de temperatură negativă, din Europa. Domeniul de insensibilitate adoptat este de ± 5° C, corespunzător clasei de precizie de 10 %, pentru scara de temperatură Celsius.

temperatură extrem de mică

$$\begin{cases} \mu_t = \dfrac{t + 75}{20} & t \in (-75,-55] \\ \mu_t = 1 & t \in (-55,-45) \\ \mu_t = \dfrac{-25 - t}{20} & t \in [-45,-25) \end{cases}$$

Funcţia de apartenenţă pentru temperatura foarte mică a fost centrată pe valoarea de -20° C, care este media minimelor de temperatură a iernilor din clima temperat continentală, din Europa Centrală şi Răsăriteană. Domeniul de insensibilitate adoptat este de ± 5° C, corespunzător clasei de precizie de 10 %, pentru scara de temperatură Celsius.

temperatură foarte mică

$$\begin{cases} \mu_t = \dfrac{t + 45}{20} & t \in (-45, -25] \\ \mu_t = 1 & t \in (-25, -15) \\ \mu_t = \dfrac{-2 - t}{13} & t \in [-15, -2) \end{cases}$$

Funcția de apartenență pentru temperatura mică a fost centrată pe valoarea de 0° C, care este originea scării Celsius. Domeniul de insensibilitate adoptat este de ± 2° C, corespunzător clasei de precizie de 4 %, pentru scara de temperatură Celsius.

Funcția de apartenență pentru temperatura normală a fost centrată pe valoarea de 20° C, care este valoare standardizată. Domeniul de insensibilitate adoptat este de ± 2° C, corespunzător clasei de precizie de 4 %, pentru scara de temperatură Celsius.

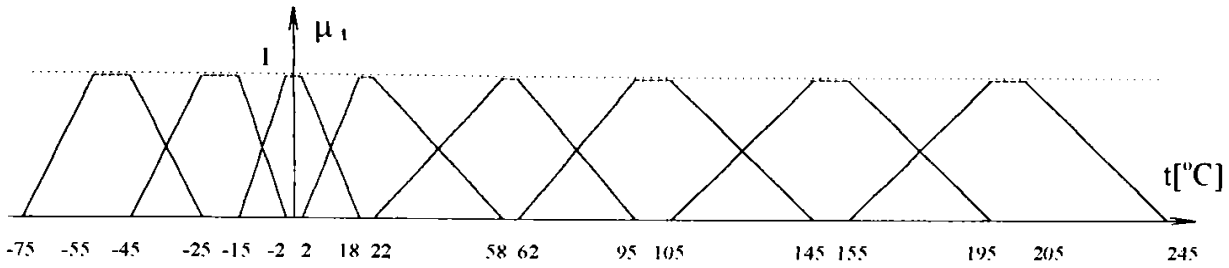


Fig.3.5

$$\text{temperatură mică} \begin{cases} \mu_t = \dfrac{t+15}{13} & t \in (-15, -2] \\ \mu_t = 1 & t \in (-2, 2) \\ \mu_t = \dfrac{18-t}{16} & t \in [2, 18) \end{cases}$$

Funcția de apartenență pentru temperatura medie a fost centrată pe valoarea de 60° C, care este valoarea medie, între temperatura normală şi temperatura mare. Domeniul de insensibilitate adoptat este de ± 2° C, corespunzător clasei de precizie de 4 %, pentru scara de temperatură Celsius.

Funcția de apartenență pentru temperatura mare a fost centrată pe valoarea de 100° C, care este a limita superioară a scării Celsius. Domeniul de insensibilitate adoptat este de ± 5° C, corespunzător clasei de precizie de 10 %, pentru scara de temperatură Celsius.

$$\text{temperatură normală} \begin{cases} \mu_t = \dfrac{t-2}{16} & t \in (2, 18] \\ \mu_t = 1 & t \in (18, 22) \\ \mu_t = \dfrac{58-t}{36} & t \in [22, 58) \end{cases}$$

temperatură medie

$$\begin{cases} \mu_t = \dfrac{t-22}{36} & t \in (22, 58] \\ \mu_t = 1 & t \in (58, 62) \\ \mu_t = \dfrac{95-t}{33} & t \in [62, 95) \end{cases}$$

temperatură mare

$$\begin{cases} \mu_t = \dfrac{t-62}{33} & t \in (62, 95] \\ \mu_t = 1 & t \in (95, 105) \\ \mu_t = \dfrac{145-t}{40} & t \in [105, 145) \end{cases}$$

temperatură foarte mare

$$\begin{cases} \mu_t = \dfrac{t-105}{40} & t \in (105, 145] \\ \mu_t = 1 & t \in (145, 155) \\ \mu_t = \dfrac{195-t}{40} & t \in [155, 195) \end{cases}$$

Funcţia de apartenenţă pentru temperatura foarte mare a fost centrată pe valoarea de 150° C, care este limita inferioară a domeniului temperaturilor de revenire joasă a oţelurilor. Peste această valoare apar transformări de structură la oţeluri. Domeniul de insensibilitate adoptat este de ± 5° C, corespunzător clasei de precizie de 10 %, pentru scara de temperatură Celsius.

Funcţia de apartenenţă pentru temperatura extrem de mare a fost centrată pe valoarea de 200° C, care este o valoare centrată pe domeniul limitelor de temperatură ale majorităţii polimerilor, folosiţi în construcţia de maşini. Domeniul de insensibilitate adoptat este de ± 5° C, corespunzător clasei de precizie de 10 %, pentru scara de temperatură Celsius.

temperatură extrem de mare

$$\begin{cases} \mu_t = \dfrac{t-155}{40} & t \in (155, 195] \\ \mu_t = 1 & t \in (195, 205) \\ \mu_t = \dfrac{245-t}{40} & t \in [205, 245) \end{cases}$$

F12. Nivelul de poluare al mediului. Praf, substanţe chimice, etc. Acest criteriu devine important în mediile puternic poluate, unde carcasa transmisiei nu poate realiza etanşarea mediului intern al transmisiei, de mediul extern. Nu există o clasificare cantitativă standardizată a tipurilor de medii poluate, pentru uzul proiectantului, în construcţia de maşini. Pentru exprimarea cantitativă a poluării se foloseşte [mg subst/ m^3aer]. Pe lîngă exprimarea cantitativă, mai este necesar, să se cuantifice şi agresivitatea agentului poluant asupra transmisiei.

F13. Umiditatea mediului. Se referă la umiditatea maximă a mediului în care transmisia poate funcţiona. Se exprimă uzual prin umiditate relativă în [%]. Pentru transmisiile deschise valoarea umidităţii relative maxime este limitată superior. Astfel, pentru transmisiile prin curea se recomandă maxim 60 - 75 % [79]. Trasmisiile închise în carcase etanşe faţă de mediu, pot funcţiona la orice valoare a umidităţii. Astfel, umiditatea devine criteriu de selecţie, atunci când, umiditatea depăşeşte pragul menţionat mai sus.

F14. Condiţii speciale de mediu - Cetrale nucleare. Prezenţa radiaţiilor de înaltă energie şi a temperaturilor extreme provoacă degradarea polimerilor şi în general, a materialelor organice. Inlocuirea pieselor de uzură din interiorul reactorului nuclear este sever retricţionată de normele de funcţionare ale acestuia.

C1. Simplicitatea / complexitatea construcţiei. Poate fi exprimată în primă aproximaţie, prin numărul de piese componente. Exprimarea prin număr este nerelevantă dacă nu se ţine cont de complexitatea pieselor, clasa de precizie, complexitatea tehnologiei (prelucrări şi reglaje). Se propune clasarea tipurilor de transmisii, după compexitatea constructivă, în trei grupe: transmisie simplă, transmisie mediu complexă şi transmisie complexă. Pentru o departajare uşoară, se restrânge numărul de componente luate în considerare la evaluarea compexităţii transmisiei, şi anume, complexitatea roţilor şi prezenţa carcasei proprii, după cum se arată în tabelul T3.6.

Complexitatea transmisiei	**Transmisie simplă s**	**Transmisie mediu complexă mc**	**Transmisie complexă c**
Complexitatea roţilor	simple	mediu complexe	complexe
Prezenţa carcasei proprii	nu	nu / da	da

T3.6

Aplicarea metodei se concretizează în coloana corespunzătoare complexităţii constructive a transmisiei din tabelul T3.8.

C2. Simplicitatea / complexitatea roţilor care intră în alcătuirea transmisiei. Exprimarea complexităţii roţilor se poate face prin compararea numărului total de suprafeţe, cu (n - 1) valori prag, pentru n atribute (complexitate mică, medie, etc.). Metoda numărului total de suprafeţe este prea laborioasă pentru a fi practică. In locul ei se propune o metodă, ce ia în consideraţie numai suprafaţa exterioară a obadei şi care este prezentată în tabelul T3.7.

Tipul roţii după complexitate	**Suprafaţa exterioară a obadei**
Roată simplă	Obadă cu suprafaţă exterioară lisă
Roată mediu complexă	Obadă cu suprafaţa exterioară canelată
Roată complexă	Obada cu suprafaţa exterioară danturată

T3.7

C3. Simplicitatea / Complexitatea lagărelor. Cu o bună aproximare se poate aprecia complexitatea lagărelor a două transmisii echivalente prin costul rulmenţilor, sau al cuzineţilor, care intră în componenţa transmisiei.

C4. Gabaritul transmisiei. Pentru a putea fi folosit ca şi criteriu de selecţie, este necesară o exprimare a gabaritului, în mărimi relative [mm^3/Kw], etc. Pe lângă mărimea volumului ocupat este importantă, uneori esenţială, şi configuraţia spaţială a acestuia. Configuraţia spaţială se poate exprima prin *semnătura spaţială* a ansamblului.

C5. Masa netă. Devine un criteriu de selecţie, dacă este exprimată în unităţi relative [Kg/Kw], sau dacă este restrictivă, în valoare absolută.

T1. Simplicitatea / complexitatea tehnologiei de fabricaţie. Se poate exprima prin numărul mediu ponderat de operaţii tehnologice.

T2. Materiale uşor / dificil de procurat. Criteriul intervine când se pune problema fabricaţiei.

Tipul transmisiei	*Simplicit/ Compexit. construct.*	*Simplicit./ Complexit. tehnol.*	*Cost fabricaţie*	*Amploare întreţinere*	*Durată menţinre reglaje*	*Poli-valenţa*
Transmisie prin curea trapezoidală clasică	mc	s	mic	mare	mică	nu
Transmisie prin curea trapezoidală înaltă	mc	s	mic	mare	mică	nu
Transmisie prin curea trapezoidală anizotropă	mc	s	mic	mare	mică	nu
Transmisie prin curea trapezoidală lată	s	s	mic	mare	mică	nu
Transmisie prin curea lată multistrat	s	s	mic	mare	mică	da
Transmisie prin curea trapezoidală multiplă	mc	s	mic	mare	mică	da
Transmisie prin curea rotundă	mc	s	mic	mare	mică	nu
Transmisie prin curea sincronă profil trapez	mc	s	mediu	mare	medie	nu
Transmisie prin curea sincronă profil rotund	mc	s	mediu	mare	medie	nu
Transmisie prin lanţ cu role şi zale	c	m	mediu	mare	medie	da
Transmisie prin lanţ cu eclise	c	m	mediu	mare	medie	da
Transmisie prin roţi de fricţiune	s	s	mic	mică	mare	nu
Transmisie prin angrenaj cilindric cu dinţi drepţi	c	c	mare	mică	mare	nu
Transmisie prin angrenaj cilindric cu dinţi înclinaţi	c	c	mare	mică	mare	nu
Transmisie prin angrenaj cilindric cu dinţi în V	c	fc	foarte mare	mică	mare	nu
Transmisie prin angrenaj conic cu dinţi drepţi	fc	c	mare	mică	mare	nu
Transmisie prin angrenaj conic cu dinţi curbi şi înclinaţi	c	fc	foarte mare	mică	mare	nu
Transmisie prin angrenaj melcat	c	fc	foarte mare	mică	mare	nu

T3.8

T3. Existenţa în procesul de fabricaţie a operaţiilor dificile. Criteriul intervine când se pune problema fabricaţiei.

T4. Simplicitatea / complexitatea montajului. Criteriul intervine când se pune problema fabricaţiei.

T5 Simplicitatea / complexitatea reglajelor. Criteriul intervine când se pune problema fabricaţiei.

T6. Posibilitatea adecvării formă - tehnologie - serie de fabricaţie. Criteriul intervine când se pune problema fabricaţiei.

T7. Costul fabricaţiei. Criteriul intervine când se pune problema fabricaţiei.

I1. Numărul şi amploarea operaţiilor de întreţinere. Se consideră ca fiind operaţii de întreţinere de amploare mare, cele referitoare la transmisiile care conţin elemente flexibile şi de amploare mică, cele pentru transmisiile care nu conţin element flexibil.

I2. Durata menţinerii reglajelor iniţiale. Durata menţinerii reglajelor este mică la transmisiile prin curele, exclusiv cele sincrone, la care fonomenul de relaxare impune reglaje dese, pentru restabilire tensiunii T_0. La transmisiile prin lanţuri durata este medie, iar la celelalte este mare.

I3. Simplicitatea / complexitatea verificării gradului de uzură şi al dereglărilor. Criteriul devine relevant în cazul transmisiilor de mare importanţă, plasate în locuri greu accesibile sau cu acces restricţionat.

I4. Simplicitatea / complexitatea activităţilor de înlocuire ale pieselor uzate sau deteriorate. Criteriul devine relevant în cazul transmisiilor de mare importanţă, plasate în locuri greu accesibile sau cu acces restricţionat.

I5. Ungerea pretenţioasă / nepretenţioasă. Ungerea este considerată pretenţioasă, în cazul transmisiilor prin angrenaje, respectiv prin lanţuri.

I6. Numărul pieselor de uzură. Piesele de uzură sunt cele declarate ca atare, de către proiectant. Acesta prognozează durabilitatea pieselor în cauză, ca fiind mult mai mică decât durata de viaţă a ansamblului. Numărul pieselor de uzură este mai mare în cazul transmisiilor prin curea.

M1. Tradiţia acţionării în domeniu. In general, este un element retrograd, care împiedică progresul tehnicii. Poate fi de ajutor în cazul rezolvării situaţiilor comune. Tradiţia în utilizarea transmisiilor prin curele, conform VDI este prezentată în tabelul T3.9.

Tipul transmisiei	Domeniul	Exemple
Transmisia prin curea trapezoidală (toate variantele)	Producerea energiei	Turbine cu apă. Agregate electrice.
	Prelucrarea lemnului	Fierăstraie
	Industria de morărit	Mori cu ciocane
	Compresoare	Transmisii de acţionare
	Industria hârtiei	Laminoare
	Industria textilă	Maşini de spălat
	Utilaj pentru construcţii	Concasoare
Transmisia prin curea trapezoidală anizotropă	Aparatură electrocasnică	Mixere
	Autovehicole	Agregate mici
	Maşini unelte	Strunguri
Transmisia prin curea trapezoidală clasică	Trasmisii reglabile	Variatoare turaţie cu roţi întrepătrunse
	Benzi transportoare	Transmisia benzilor transportoare
	Maşini agricole	Combine

Transmisia prin curea trapezoidală înaltă	Utilaj pentru construcţii	Prese pt. construcţii; Malaxoare
	Maşini tipografice	Maşini rotative
	Maşini electrice	Agregate electrice
	Benzi transportoare	Acţionările de ridicare
	Maşini pentru industria lemnului	Gatter; Maşină de rindeluit
	Maşini pentru mase plastice	Extruder de mase plastice
	Compresoare	Compresor individual
	Maşini agricole	Cositoare
	Maşini pentru industria textilă	Maşini de tricotat; Uscător cu cilindru
	Pompe	Pompe centrifugale
	Ventilatoare	Ventilatoare axiale
Transmisia prin curea lată multistrat	Construcţii	Valţuri vibratoare
	Transporturi	Trasportoare de piese
	Maşini pentru mase plastice	Malaxoare
	Maşini agricole	Secerătoare
	Maşini pentru industria hârtiei	Dezintegratoare
Transmisia prin curea trapezoidală multiplă	Industria tipografică	Maşini Offset
	Aparate mari de uz casnic	Maşini de spălat
	Aparate mici de uz casnic	Maşini de găurit
	Maşini pentru industria lemnului	Maşini de frezat
	Motoare termice	Dispozitive auxiliare la motoare
	Maşini unelte	Acţionarea arborelui principal
Transmisia prin curea trapezoidală lată	Variatoare de turaţie	Motovariatoare
	Industria tipografică	Maşini Offset multicolor
	Maşini agricole	Combine
	Maşini pentru industria textilă	Maşini de bobinat
	Maşini unelte	Strunguri
Transmisia prin curea sincronă cu profil trapez	Maşini de birou	Maşini de scris
	Dispozitive periferice	Plotere
	Proiectoare cinematografice	Transmisia rolelor de film
	Transport	Lifturi
	Transmisii	Transmisii
	Aparate electrocasnice	Maşini de spălat
	Maşini pentru industria hârtiei	Acţionarea disp. de prindere
	Roboţi	Acţionarea de poziţionare
	Motoare cu ardere internă	Acţionarea distribuţiei
	Maşini de împachetat	Numărătoare
Transmisii prin curea sincronă cu profil arc de cerc	Maşini de birou	Copiatoare
	Dispozitive periferice	Imprimante
	Dispozitive electrice	Maşini de rindeluit
	Transmisii	Transmisii reductoare
	Aparate mici de uz casnic	Roboţi de bucătărie
	Maşini pentru industria lemnului	Maşini de încleiat
	Dispozitive pentru grădinărit	Maşini de tuns iarba
	Roboţi	Acţionarea de poziţionare
	Maşini pentru industria textilă	Maşini de tricotat
	Motoare cu ardere internă	Acţionarea distribuţiei
	Maşini unelte	Acţionarea de avans

T3.9

M2. Posibilitatea înlocuirii cu altă transmisie, cu eforturi minime. Cheltuielile de cercetare pentru înlocuirea unei transmisii consacrate cu alta nouă.

M3. Existenţa cercetărilor promiţătoare în domeniu. Numărul şi importanţa colectivelor de cercetare, care se ocupă de transmisii mecanice, specificate pe tipuri.

M4. Influenţa activităţilor comerciale şi a modei. Amploarea reclamei referitoare la transmisii mecanice.

M5. Polivalenţa transmisiei. Se referă la calitatea transmisiei de a putea îndeplini funcţii suplimentare celei de transmitere a mişcării. Polivalenţa este întâlnită la curele şi lanţuri speciale, care printr-un profil adecvat sau prin piese suplimentare ataşate, pot îndeplini funcţii de transport sau de acţionare.

Utilizarea unui număr prea mare de criterii conduce la o selecţie irelevantă. Tratatele de decizie multicriterială recomandă ca, numărul criteriilor într-o decizie să nu depăşească 10. In acest capitol, s-au prezentat parametri care pot fi folosiţi ca şi criterii la elaborarea unei decizii de alegere a tipului de transmisie.

Parametri consideraţi relevanţi sunt: F1, F2, F4, F5, F6, F7, F8, F9, F10, F11, F12, C1, T1, T7, I1, I2, M1 şi M5. Mulţimea şi diversitatea parametrilor prezentaţi nu neagă recomandarea referitoare la numărul maxim de criterii folosite într-o decizie.

3.3 TRANSMISII MECANICE. CLASE. RELATII. TEORII.

Organizarea pe clase a domeniului transmisiilor mecanice este necesară pentru construirea bazei de cunoştinţe, ca parte componentă a Sistemului Expert. Preîntâmpinarea apariţiei inconsistenţei bazei de cunoştinţe cere, printre altele, şi o riguroasă organizare a cunoştinţelor din domeniu. Gruparea specimenelor şi constituirea claselor s-a făcut pe baza analizei funcţiilor de bază, ale transmisiilor mecanice. S-a constatat că denumirea de transmisie mecanică este improprie clasificării şi s-a adoptat denumirea de transmisie de putere, păstrând vechea denumire pentru transmisia elementară.

Construirea structurii ierarhice a transmisiilor mecanice este posibilă, dacă se poate enunţa un set de definiţii riguroase, în care genul proxim să fie lipsit de ambiguitate, iar diferenţa specifică să fie semnificativă.

Clasa transmisie_mecanică. Definirea acestei clase porneşte de la cele două funcţii ale transmisiei: transmiterea mişcării şi transformarea cinematică.

1. Transmiterea mişcării există atunci când:

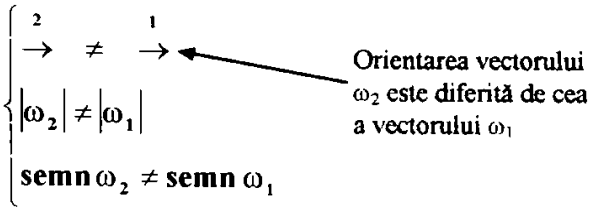
Funcţia de bază — Condiţiile forte care definesc funcţia de bază:
- există un element de transmitere, între două elemente mobile;
- elementul de transmitere este de natură mecanică;
- există transfer de energie, între cele două elemente.

Calitatea specifcă transmisiei mecanice

2. Transformarea cinematică este defintă de condiţia:

$$\overline{\omega_2} \neq \overline{\omega_1}$$

Aceea ce înseamnă, neidentitatea vectorilor vitezelor unghiulare ale maşinii motoare, respectiv e lucru. Situaţia se poate petrece atunci când, cel puţin una din condiţiile de mai jos este devărată:

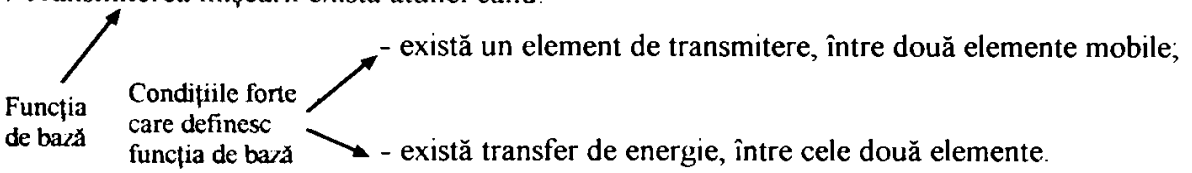


Pentru a putea enunța definiția transmisiei mecanice trebuie să răspundem la următoarele întrebări:

- Câte dintre condițiile ataşate celor două funcții este necesar să fie îndeplinite, pentru ca o construcție mecanică să aparțină clasei transmisie mecanică ?
- Din ce clasă părinte face parte transmisia mecanică ?
- Ce proprietăți este necesar să se moştenească ?

Admițând că, transmiterea mişcării mecanice are ca parametru central puterea mecanică, se denumeşte clasa care o cuprinde, anume clasa transmisie_ de_putere. Funcțiile acestei clase sunt tot cea energetică şi cea cinematică.

1. Transmiterea mişcării există atunci când:



2. Transformarea cinematică este defintă de condiția:

$$\overline{\omega_2} \neq \overline{\omega_1}$$

Legătura tare poate fi: hidraulică, mecanică, magnetică, electrică, etc. O definiția ontologică a transmisiei mecanice poate fi următoarea:

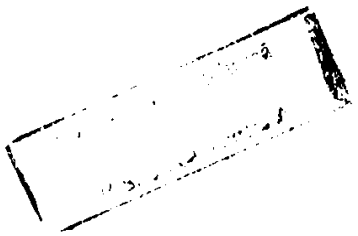
Este transmisie_mecanică acel sistem_tehnic_mobil care aparține clasei transmisie_de_putere şi care îndeplineşte condiția suficientă: elementul de transmitere este de natură mecanică.

Specimenul clasei transmisie_mecanică este tipul_transmisie_mecanică. Definiția clasei transmisie_de_putere poate fi:

Este transmisie_de_putere este acel sistem_tehnic care aparține clasei sistem_tehnic_mobil care îndeplineşte condițiile suficiente: transmite_miscarea, are cel puțin un element_de_transmitere, efectuează_transformare_cinematică.

Specimenul clasei transmisie_de_putere este transmisia. Separarea trăsăturilor esențiale şi comune de cele esențiale şi distincte ale grupurilor de obiecte tehnice, conduce la putința definirii claselor, în care acestea se includ. Orânduirea claselor este ierarhică, de la clasele generice, spre cele particulare, fără să se excludă conectarea întreţesută.

Pornind de la analiza funcțiilor transmisiei mecanice şi de la definițiile ontologice ale clasei, funcției, specimenului, s-a construit structura din Fig.3.6. Structura este organizată ierarhic pe mai multe nivele. Numărul nivelelor acestei structuri nu depăşeşte numărul nivelelor organizării naturale a lumii vii.



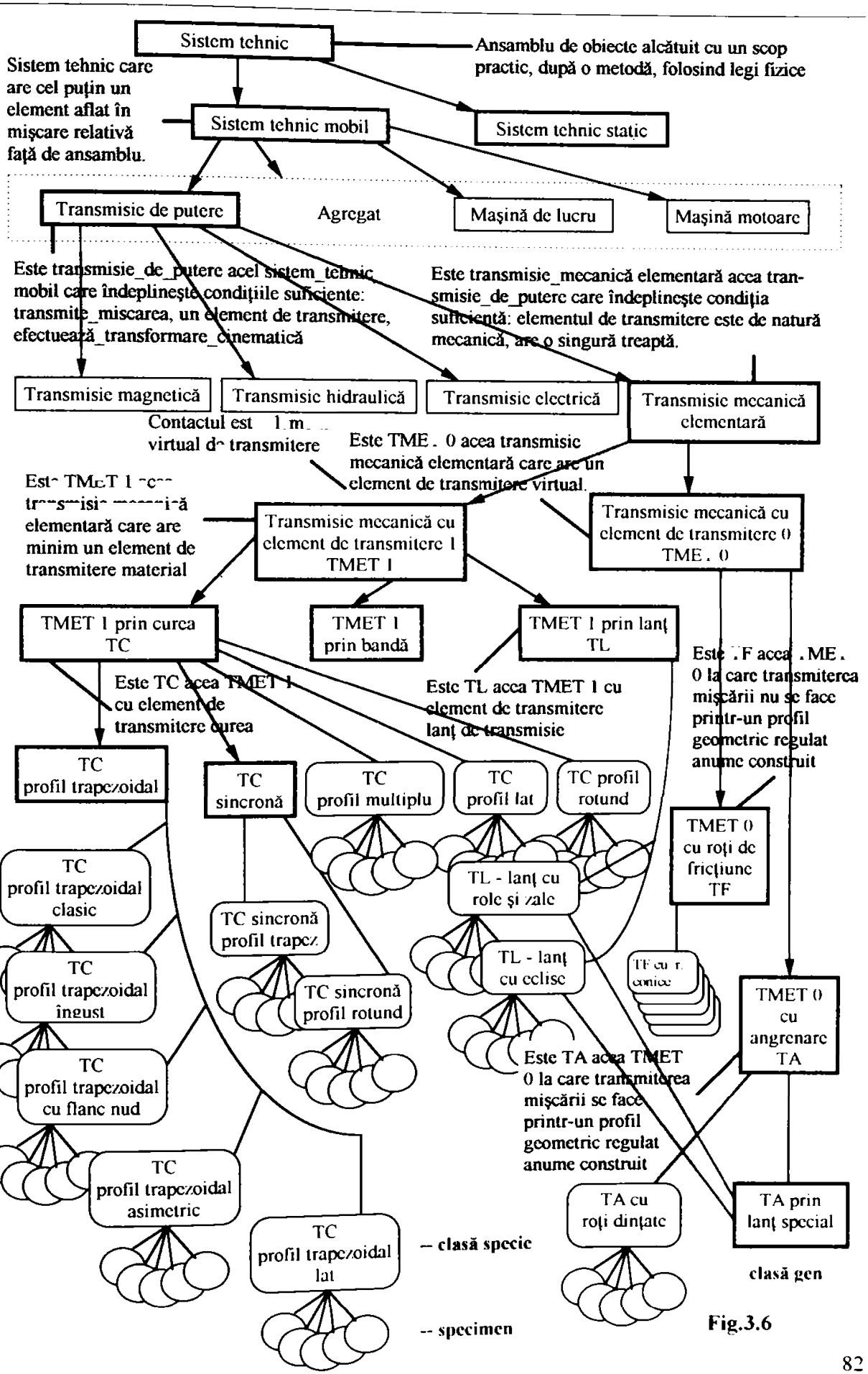

Sistem tehnic
Ansamblu de obiecte alcătuit cu un scop practic, după o metodă, folosind legi fizice
Sistem tehnic care are cel puţin un element aflat în mişcare relativă faţă de ansamblu.
Sistem tehnic mobil
Sistem tehnic static
Transmisie de putere
Agregat
Maşină de lucru
Maşină motoare
Este transmisie_de_putere acel sistem_tehnic mobil care îndeplineşte condiţiile suficiente: transmite_miscarea, un element de transmitere, efectuează_transformare_cinematică
Este transmisie_mecanică elementară acea transmisie_de_putere care îndeplineşte condiţia suficientă: elementul de transmitere este de natură mecanică, are o singură treaptă.
Transmisie magnetică
Transmisie hidraulică
Transmisie electrică
Transmisie mecanică elementară
Contactul est l m virtual d transmitere
Este TME . 0 acea transmisie mecanică elementară care are un element de transmitere virtual.
Est TMET 1 c transmisie mecanică elementară care are minim un element de transmitere material
Transmisie mecanică cu element de transmitere 1 TMET 1
Transmisie mecanică cu element de transmitere 0 TME . 0
TMET 1 prin curea TC
TMET 1 prin bandă
TMET 1 prin lanţ TL
Este TC acea TMET 1 cu element de transmitere curea
Este TL acea TMET 1 cu element de transmitere lanţ de transmisie
Este TF acea TME . 0 la care transmiterea mişcării nu se face printr-un profil geometric regulat anume construit
TC profil trapezoidal
TC sincronă
TC profil multiplu
TC profil lat
TC profil rotund
TMET 0 cu roţi de fricţiune TF
TC profil trapezoidal clasic
TC profil trapezoidal îngust
TC profil trapezoidal cu flanc nud
TC profil trapezoidal asimetric
TC sincronă profil trapez
TC sincronă profil rotund
TL - lanţ cu role şi zale
TL - lanţ cu eclise
TF cu r. conice
TMET 0 cu angrenare TA
Este TA acea TMET 0 la care transmiterea mişcării se face printr-un profil geometric regulat anume construit
TA cu roţi dinţate
TA prin lanţ special
TC profil trapezoidal lat
-- clasă specie
-- specimen
clasă gen

Fig.3.6

Nivelul suprem, cel al universului artificial, conţine o noţiune unică, cu valoare de categorie, intitulată *Sistem tehnic*. Definiţia sistemului tehnic, este o definiţie axiomatică, motiv pentru care nu urmează metoda gen proxim-diferenţă specifică. Aceasta precizează caracterul scopului construirii sistemului tehnic, modul de alcătuire şi mijloacele folosite. Celelalte definiţii, care descriu noţiunile plasate pe nivelele inferioare, folosesc metoda logică, consacrată, de definire.

Schema din Fig.3.6 conţine definiţiile principalelor clase, care alcătuiesc încrengătura transmisiilor mecanice. Lipsesc din schemă, din raţionamente practice, definiţiile claselor de bază şi definiţiile specimenelor.

Clasele aflate pe nivele diferite se leagă pe baza principiului moştenirii, adică a transmiterii proprietăţii principale, de la clasa superioară, la clasa inferioară.

Definiţiile enunţate mai sus au rolul de a stabili metoda de construcţie ierarhică a domeniului transmisiilor mecanice. Ele vor fi explicitate la realizarea modulelor de program care alcătuiesc Sistemul Expert.

Concluzii

- Folosind această metodă pentru construirea unei structuri ontologice se înlătură arbitrariul, des întâlnit în clasificările empirice.
- O structură astfel construită, permite funcţionarea moştenirii proprietăţilor generale, la nivelul tuturor relaţiilor structurii.
- Funcţionalitatea moştenirii conduce la o bună eficienţă, a comprehensiunii cunoştinţelor.

Structura astfel construită, facilitează implementarea cunoştinţelor de transmisii mecanice într-o bază de cunoştinţe ataşată unui Sistem Expert.

4. METODA DE ABORDARE A APLICATIEI

4.1. NIVELE DE CONVERSATIE

Startul unui dialog om-sistem constituie o problemă în sine. Aceasta se conturează, în mod pragmatic, pornind de la întrebarea: "La ce nivel al reprezentării cunoaşterii trebuie început dialogul ?".

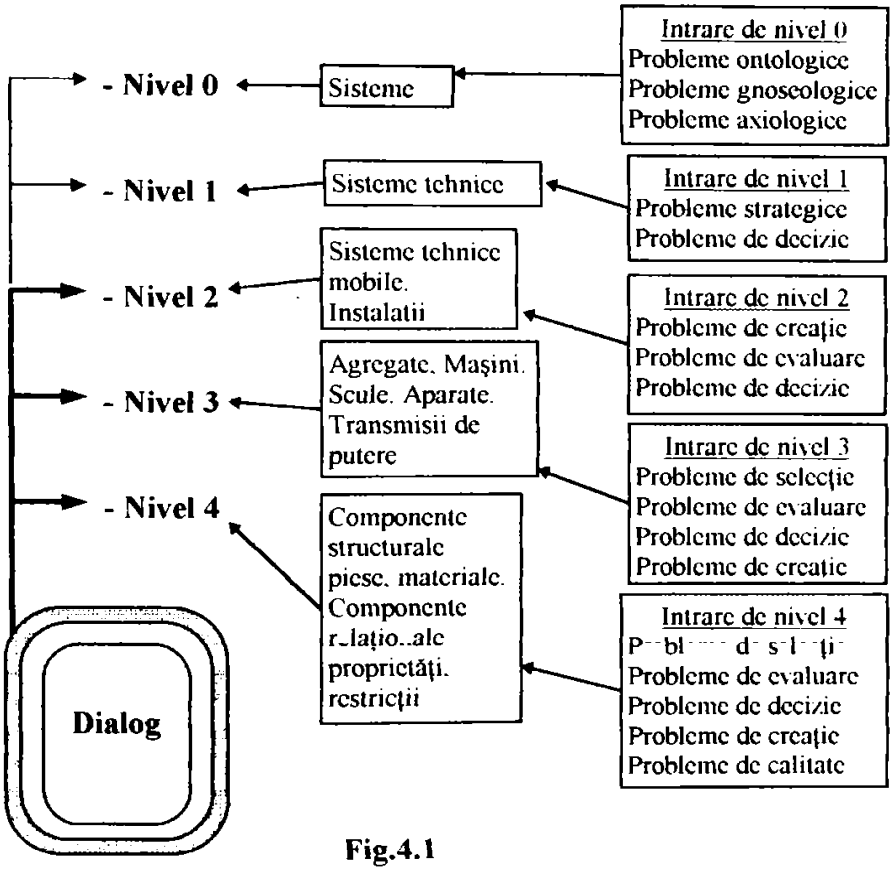


Fig.4.1

Privind ierarhia sumară, care este prezentată în capitolul de definiţii şi care este schiţată mai jos, se constată că, în principiu, se poate începe dialogul de la oricare dintre nivele.

Gradul de generalitate al temei influenţează locul de pornire al dialogului Este de preferat un nivel cât mai coborât, , entru a se putea ajunge repede la soluţii concrete. Se estimează că utilizatorul doreşte soluţii concrete, în timp scurt şi cu grad înalt de încredere.

O altă întrebare se referă la modul în care se evaluează şi apoi se propagă gradul de ignoranţă, atât al utilizatorului, cât şi al sistemului însuşi. Dialogul se poartă, de la general, spre particular, începând de la un anumit nivel. Nivelele superioare transferă proprietăţi generale nivelelor inferioare. Acest transfer este relaţia principală într-o ierarhie şi se numeşte moştenire. Proprietăţile moştenite îmbogăţesc baza de dialog.

Dialogul sistem-utilizator trebuie să constituie, atât un mijloc de satisfacere a necesităţilor utilizatorului, cât şi o sursă de învăţare pentru sistem.

Este necesar, pentru protecţia bazei de cunoştinţe, să existe o supervizare a ceea ce se reţine din dialog, pentru a putea deveni cunoştinţe permanente. Intr-o etapă preliminară de dezvoltare este necesar să se definească structura ontologică a universului.

4.1.1. Nivele de generalitate ale dialogului

Purtarea dialogului între sistem şi utilizator se poate optimiza, dacă problema globală este descompusă în sub-probleme şi dacă dialogul este de asemenea organizat pe nivele de generalitate. Intrebările pot să fie asfel grupate şi utilizate într-o anumită succesiune. Reprezentarea din Fig.4.1 foloseşte un număr de cinci nivele de generalitate, care grupează întrebări orientate spre domeniul transmisiilor mecanice (la nivelele inferioare).

4.2. DEFINIREA STRATEGIEI DE REZOLVARE A TEMEI

4.2.1. Postulatele căutării soluției şi ale dezvoltării Sistemului Expert

1. <u>Soluția cea mai simplă este cea mai bună.</u> O problemă tehnică acceptă o *infinitate* de soluții posibile. Submulțimea alcătuită din soluțiile cunoscute, plus cele predictibile, este restrânsă, dar dificil de ierarhizat. La parametri funcționali echivalenți, se constată, în mod evident, că soluțiile cele mai minuțios elaborate, şi care sunt exponentele tehnologiei celei mai avansate, sunt cele mai simple constructiv. Simplicitatea este o formă de manifestare a progresului tehnic. Aplicarea practică a acestui postulat conduce la o metodă de rezolvare a problemei, care se aplică în mai multe faze.

Faza I se enunță astfel:

Se renunță la transmisie dacă parametri arborelui motor satisfac necesitățile maşinii de lucru, adică $(P_2, \omega_2) \equiv (P_1, \omega_1)$.

Situația este ilustrată în Fig.4.2

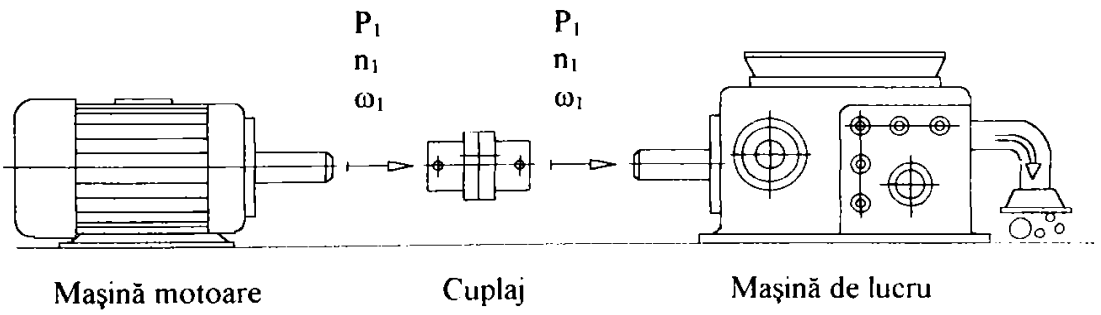


Fig.4.2

Din punct de vedere practic, rezolvarea urmăreşte găsirea *arborelui motor* cel mai adecvat cerințelor maşinii de lucru. *Cel mai adecvat* arbore motor însemnând acela care poate îndeplini simultan cele trei condiții de mai jos:

$$\begin{cases} (P_1)_{util} \geq (P_2)_{necesar} \\ (\omega_1)_i = (\omega_2)_i \\ (\omega_2)_i \in [\omega_{2\,minim\,necesar}, \omega_{2\,maxim\,necesar}] \end{cases} \qquad (4.1)$$

unde $i \in N$ şi reprezintă trepte de viteză ale maşinii de lucru.

Faza a II-a:

Se alege transmisia de putere tipizată, cea mai simplă, adecvată setului de restricţii ale maşinii de lucru, considerate semnificative.

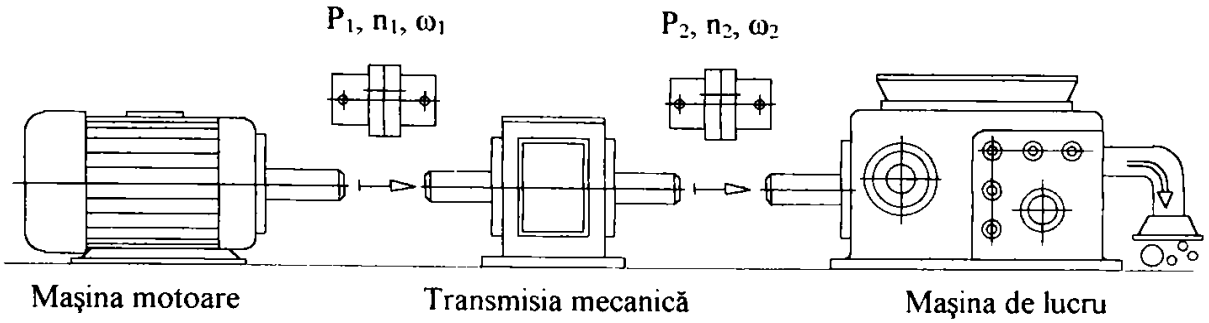


Fig.4.3

In această fază, restricţiile impuse arborelui motor (motorului) sunt diminuate, urmând ca adaptarea caracteristicilor mecanice, ale celor două maşini, să fie efectuată de transmisie. Alegerea tipului de transmisie se face cu luarea în considerare a cinci seturi de restricţii, conform schemei din Fig.4.5. Schema bloc a fazei a II-a este ilustrată Fig.4.3.

Faza a III-a:

Se proiectează transmisia de putere ca anasamblu şi apoi se proiectează fiecare transmisie elementară, prin folosirea preferenţială a componentelor tipizate.

Se realizează o proiectare eficientă prin folosirea de elemente tipizate, curele, lanţuri, roţi (Fig.4.4), respectând restricţiile impuse, în ordinea obţinută prin ierarhizare. Procesul de ierarhizare foloseşte un grup de criterii ordonate, după metoda deciziei impuse. La acest nivel, alegerea se referă la găsirea tipului de transmisie şi a formelor constructive, care să constituie cel puţin un progres tehnic minimal, faţă de soluţiile cunoscute. Atingerea acestui scop este posibilă prin formularea unui set, cât mai complet, de restricţii constructive.

Etapele de calcul conţin elemente de rutină şi elemente de alegere euristică a materialelor şi a elementelor de formă. Criteriul simplicităţii, în această fază, se concretizează prin minimizarea numărului de suprafeţe şi prin alegerea suprafeţelor cele mai simple (punctul de vedere tehnologic, mai sofisticat decât cel pur geometric, este primul luat în consideraţie).

2. <u>Corelarea etapizării rezolvarii unei probleme cu moştenirea.</u> Problema globală este descompusă în subprobleme, care sunt rezolvate individual, pe etape. Atât dialogul, cât şi inferenţa sistemului se desfăşoară pe nivele diferite. Nivelele de dialog sunt precizate în Fig.4.1. Inferenţa, în schimb, trebuie să folosească avantajul moştenirii, de către clasele inferioare, a proprietăţilor provenite de la clasele superioare.

Moştenirea face să se reducă drastic cantitatea de cunoştinţe necesare pentru rezolvarea unei probleme, prin păstrarea elementelor comune de cunoaştere, înglobate în cunoştinţe cu caracter general.

Folosirea moştenirii conduce la posibilitatea comprehensiunii cunoştinţelor. Etapizarea rezolvării unei probleme trebuie corelată cu structura bazei de cunoştinţe folosită, pentru a putea beneficia, în toate etapele inferenţei, de avantajele moştenirii.

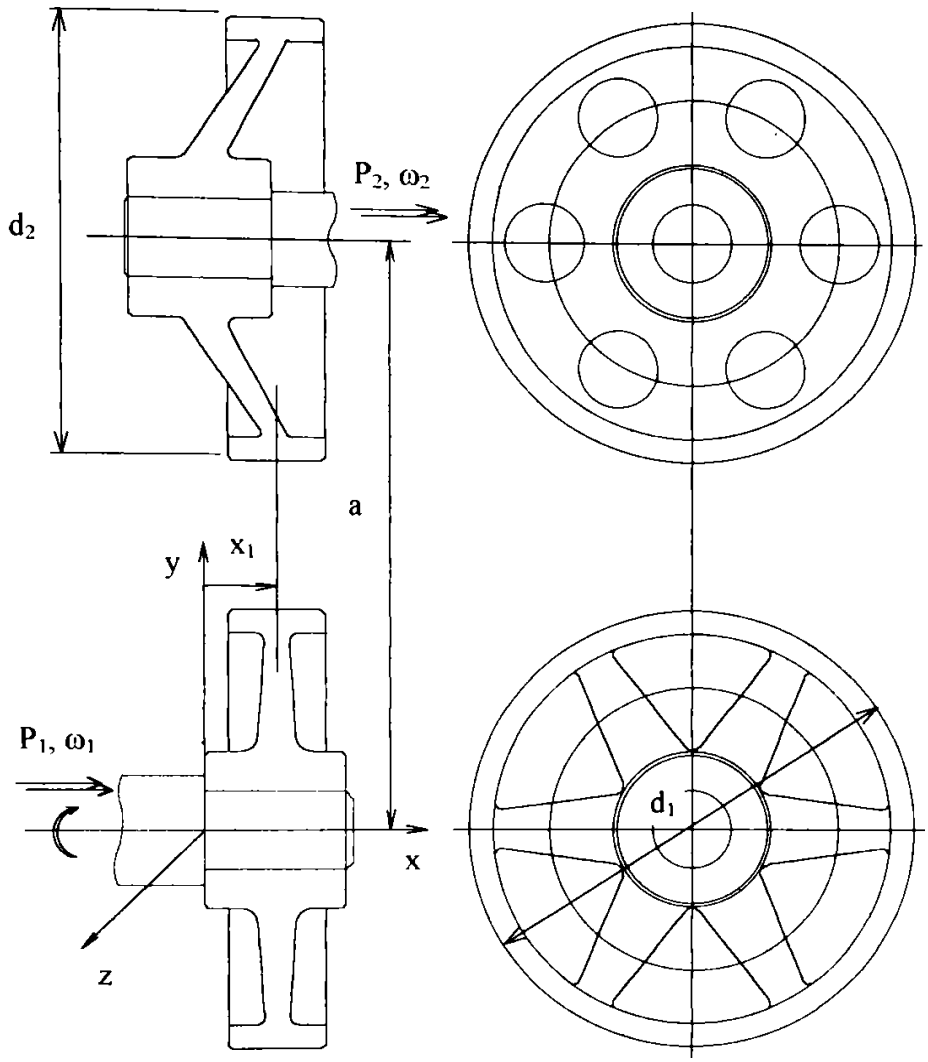


Fig.4.4

3. <u>Inconsistenţa bazei de cunoştinţe - caracteristica unui sistem de cunoaştere evolutiv</u>

Sensul invers moştenirii este cel al construcţiei epistemologice, acţiune care edifică noi componente (clase) ale structurii, atât din necesitatea pură de dezvoltare a bazei de cunoştinţe, cât şi din necesitatea acută de eliminare/limitare a inconsistenţei. Inconsistenţa este nocivă, pentru că blochează inferenţa prin nedeterminism, sau în cazul folosirii funcţiilor de ignorare, generează soluţii contradictorii.

Problema inconsistenţei solicită răspunsuri la câteva întrebări:

3.a. Cum evoluează consistenţa unei baze de cunoştinţe dinamice, la modificări de conţinut ?

3.b. Este necesar să eliminăm inconsistenţa sau să o menţinem ?

3.c. Cum eliminăm inconsistenţa ?

3.d. Cât este limita gradului de inconsistenţă, ce poate fi menţinută ?

Observaţie: Pentru enunţurile următoare se foloseşte operatorul :: cu semnificaţia

a # b ~ a şi b; numai a; numai b

3.a. Evoluţia bazei de cunoştinţe, privită ca etapă evolutivă elementară, are o stare iniţială, o stare finală şi o procedură de transformare. Atât starea iniţială, cât şi cea finală, pot avea una din cele două valori: consistentă sau inconsistentă. Procedura de transformare poate fi: de adăugare de fapte # reguli, de eliminare de fapte # reguli, de procesare a conţinutului existent, fără nici o adăugare, sau eliminare de conţinut gnostic.

Adăugarea şi eliminarea simplă, de fapte # reguli conduc, deopotrivă, la o stare finală incertă. Eliminarea incertitudinii transformării se face prin înlocuirea adăugării şi eliminării simple, (nefiltrate), cu proceduri filtrate. Din numărul total de 20, sunt utile următoarele 7 variante:

3.a.1. O bază de cunoştinţe inconsistentă, prin completare cu noi fapte # reguli, rămâne inconsistentă.

3.a.2. O bază de cunoştinţe consistentă, prin eliminare de fapte # reguli, rămâne consistentă.

3.a.3. O bază de cunoştinţe inconsistentă, prin eliminare nefiltrată de fapte # reguli, rămâne, cel mai probabil, inconsistentă.

3.a.4. O bază de cunoştinţe consistentă, prin completare nefiltrată cu fapte # reguli, devine, cel mai probabil, inconsistentă.

3.a.5. O bază de cunoştinţe consistentă, prin completare filtrată cu fapte # reguli, rămâne consistentă.

3.a.6. O bază de cunoştinţe inconsistentă, prin eliminare filtrată de fapte # reguli, devine consistentă.

3.a.7. O bază de cunoştinţe inconsistentă poate deveni consistentă, prin procesarea conţinutului. Se înţelege prin procesarea conţinutului, păstrarea conţinutului gnostic, concomitent cu dezvoltarea structurii bazei.

3.b. Principial, inconsistenţa apare în procesul de instruire al unui sistem. Intotdeauna cunoaşterea unui sistem este restrânsă, limitată şi învechită. Extinderea, deschiderea şi înnoirea sunt deziderate ale oricărui sistem cognitiv. Neacceptarea *ab initio* a apariţiei inconsistenţei, conduce la o frânare accentuată, a procesului de autoinstruire şi la diminuarea, sau chiar înlăturarea, capacităţii creative a acestuia. Inconsistenţa este atât acceptabilă, cât şi necesară, cu condiţia stabilirii unor reguli destinate să o supravegheze.

3.c. Pericolul pierderii consistenţei unei baze de cunoştinţe este iminent, la orice completare a acesteia cu fapte # reguli noi. Eliminarea inconsistenţei, atunci când devine necesară, se poate face: prin filtrare la completarea bazei de cunoştinţe, prin filtrare la epurarea bazei de cunoştinţe sau prin creearea de clase noi şi redefinirea faptelor şi regulilor, care au provocat inconsistenţa, în raport cu noile clase.

3.d. Exprimarea gradului de inconsistenţă se poate face prin raportul dintre numărul de contradicţii şi numărul total de fapte şi reguli. Problema dificilă constă în, stabilirea unui prag superior şi a unui prag inferior, al gradului de inconsistenţă, definit mai sus. Pragul superior, trebuie să declanşeze un proces de eliminare al contradicţiilor, iar cel inferior, să declanşeze un proces de activare al acumulării de cunoştinţe. Gradul de inconsistenţă poate să crească, până la nivelul la care deducţia devine imposibilă, sau lipsită de relevanţă.

4. <u>Sistemul devine evolutiv-independent, prin adăugarea unui mecanism epistemologic</u>

Sistemul Expert, pentru a-şi justifica necesitatea, trebuie să se constituie într-un factor remarcabil de progres. Sistemele Expert, de tip bază de cunoştinţe, nu îşi pot sesiza propria inconsistenţă, iar dacă o sesizează nu o pot elimina, fără ajutorul inginerului de cunoaştere.

Aceste Sisteme Expert nu pot aspira să treacă testul Turing. Incadrarea lor în domeniul Inteligenței Artificiale este forțată.

Introducerea unui mecanism epistemologic, dă sistemului capacitatea eliminării inconsistenței, prin construcția propriilor clase şi dezvoltarea structurii bazei de cunoştințe, care devine originală pe cale artificială. Astfel, se creează posibilitatea apariției unui germene de cunoaştere metafizică (specifică inteligenței umane) şi odată cu aceasta, perspectiva trecerii testului Turing.

4.2.2. Stabilirea restricțiilor proiectării transmisiilor mecanice

Parametri de performanță tehnică ai transmisiilor au fost prezentați în capitolul 3. O parte dintre aceştia sunt selectați pentru etapa de alegere a tipului de transmisie. Gruparea şi ierarhizarea criteriilor de selecție precede alcătuirea strategiei de căutare a soluției şi de învățare a sistemului. Schema restricțiilor este prezentată în Fig.4.5. Această schemă stă la baza stabilirii scopului programului Sistem Expert. Acest scop este:

GASESTE SOLUTIE

GASESTE SOLUTIE :-

ALEGE TIP şi

DECIZIE DE CONTINUARE.

ALEGE TIP :-

DIALOG ASUPRA TIPULUI şi

INFERENTA TIP şi

ENUNTUL CONCLUZIILOR şi

COMPLETAREA CUNOSTINTELOR şi

PROSPECTAREA CONSISTENTEI şi

CORECTAREA CONSISTENTEI.

DECIZIE DE CONTINUARE :-

DIALOG PENTRU CONTINUARE şi

(PROIECTEAZA TRANSMISIE sau

ALEGE TRANSMISIE EXISTENTA)

ALEGE TIP

{

DIALOG ASUPRA TIPULUI :-

CHESTIONAREA UTILIZATORULUI şi

Intrebări referitoare la consumator:
- Care este numărul consumatorilor ?
- Din ce grupă consacrată de consumatori fac parte, sau la ce grupă sunt afini ?
- Aveți sursă de putere ?
- Doriți o sursă de putere ?
- La ce distață sunt dispuşi consumatorii ?
- In ce poziție spațială sunt dispuşi consumatorii ?
- Care sunt caracteristicile energetice ale consumatorilor ?
- Care sunt parametri mediului?

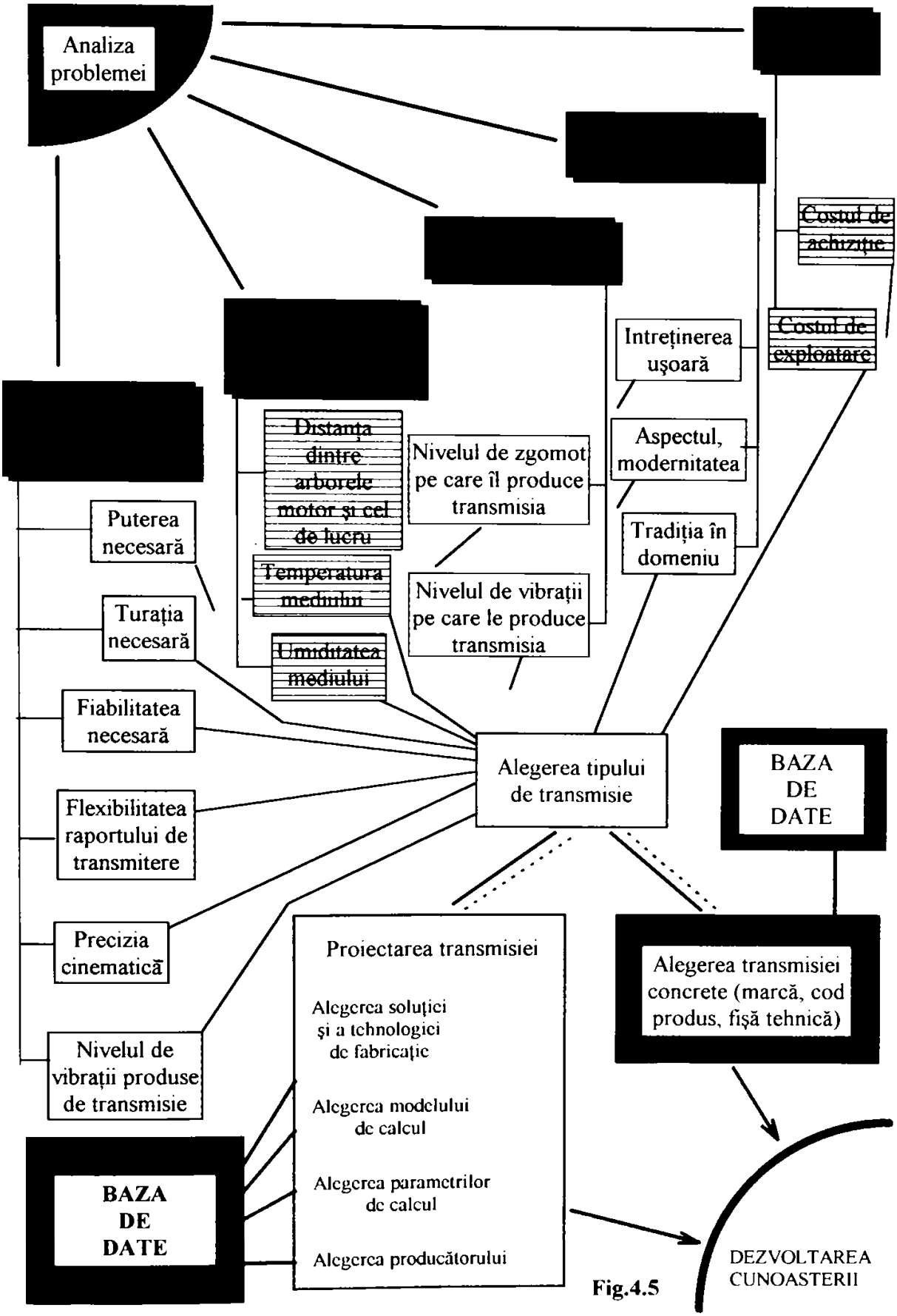

Analiza problemei
Costul de achiziție
Costul de exploatare
Întreținerea ușoară
Aspectul, modernitatea
Tradiția în domeniu
Distanța dintre arborele motor și cel de lucru
Nivelul de zgomot pe care îl produce transmisia
Temperatura mediului
Nivelul de vibrații pe care le produce transmisia
Umiditatea mediului
Puterea necesară
Turația necesară
Fiabilitatea necesară
Flexibilitatea raportului de transmitere
Precizia cinematică
Nivelul de vibrații produse de transmisie
Alegerea tipului de transmisie
BAZA DE DATE
Proiectarea transmisiei
Alegerea soluției și a tehnologiei de fabricație
Alegerea modelului de calcul
Alegerea parametrilor de calcul
Alegerea producătorului
Alegerea transmisiei concrete (marcă, cod produs, fișă tehnică)
BAZA DE DATE
DEZVOLTAREA CUNOAȘTERII

Fig.4.5

EVALUAREA CERERII şi

INFERENTA TIP şi

AFISAREA SOLUTIEI PROPUSE

INFERENTA TIP :-

ENUNTUL CONCLUZIILOR şi

COMPLETAREA CUNOSTINTELOR şi

PROSPECTAREA CONSISTENTEI şi

CORECTAREA CONSISTENTEI.}

Stabilirea datelor necesare alegerii tipurilor de transmisii necesare satisfacerii cererii utilizatorului. Completarea datelor din baza de date.

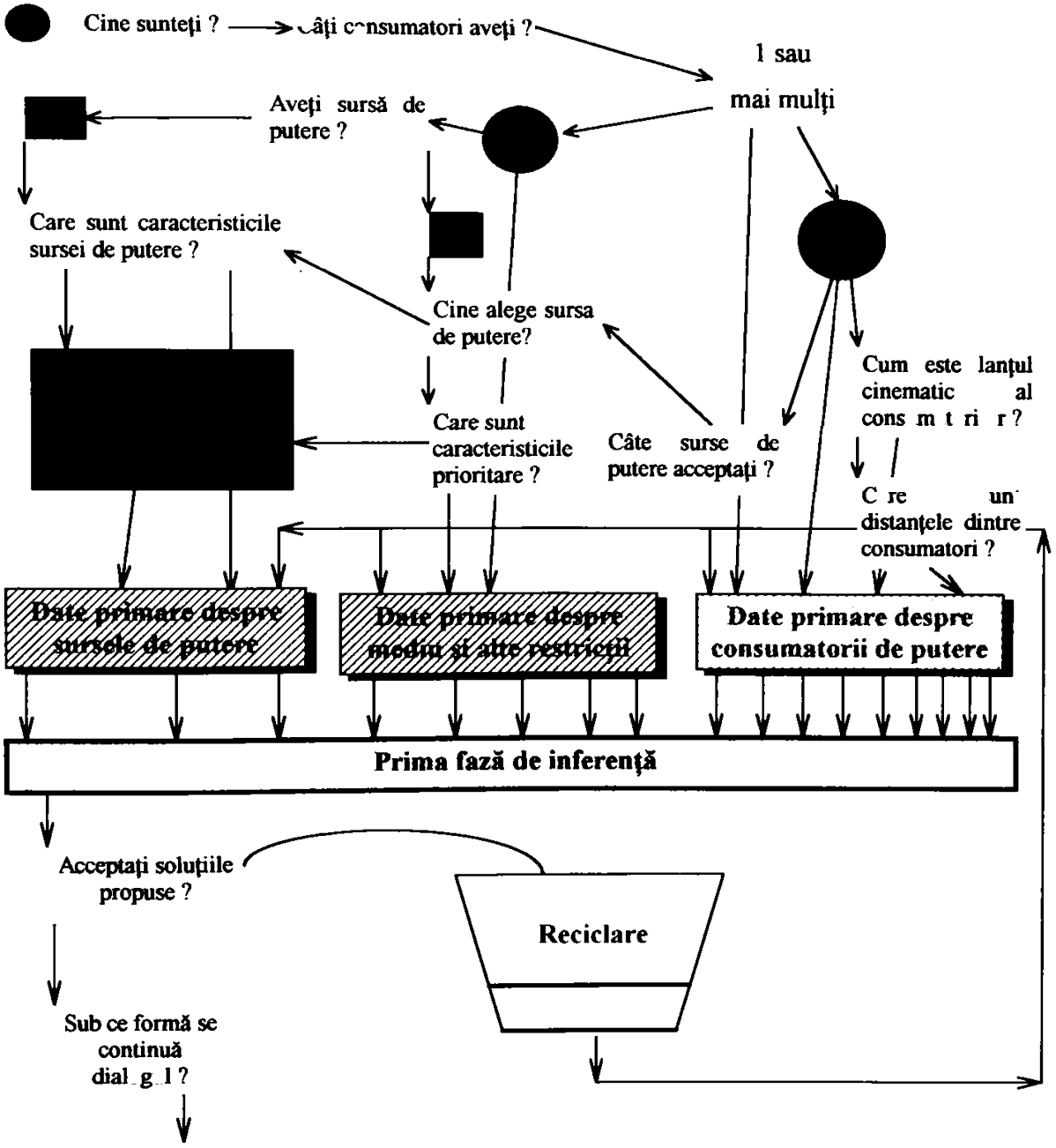


Fig.4.6

4.2.3.Chestionarea utilizatorului

Intrebările trebuie astfel grupate încât, să furnizeze informaţii necesare progresului inferenţei. Nu este productiv să se pună toate întrebările într-o singură etapă, căci nici sistemul şi nici utilizatorul nu au la început problema complet definitivată. Etapele propuse sunt următoarele:

1. întrebări referitoare la identitatea utilizatorului;
2. întrebări referitoare la domeniul tehnicii de care aparţine maşina proiectată;
3. întrebări referitoare la maşina de lucru şi la cele mai importante restricţii care se impun (atât ansamblului cât şi transmisiei);
4. întrebări de completare necesare rafinării soluţiilor;
5. întrebări de testare a utilizatorului.

Conţinutul primei etape de interogare, care precede partea de proces de inferenţă, până la alegerea tipului, se desfăşoară conform schemei din Fig.4.6.

4.3.ALEGEREA TIPULUI DE TRANSMISIE ELEMENTARA

4.3.1.Numărul de fapte necesar pentru decizia de tip.

Se propune ca operaţia de alegere a tipului de transmisie să folosească logica Fuzzy. Deciziile bazate pe logica Fuzzy folosesc reguli, care conţin atribute evaluate, cu ajutorul funcţiilor de apartenenţă. Aceste fapte au structura ilustrată în exemplul, de mai jos:

- transmisie_mecanică (Nr_valoric) :- distanţa_f_mică, atrib_dist, putere_f_mică, atrib_putere, cuplu_mare, atribut_cuplu, randament_f_mare, atribut_randament, context, procesare_ atribute(Nr_valoric).

- transmisie_mecanică(Nr_valoric) :- distanţa_f_mică, atrib_dist, putere_mică, atrib_putere, cuplu_mare, atribut_cuplu, randament_f_mare, atribut_randament, context, procesare_ atribute(Nr_valoric).

- transmisie_mecanică(Nr_valoric) :- distanţa_f_mică, atrib_dist, putere_medie, atrib_putere, cuplu_mare, atribut_cuplu, randament_f_mare, atribut_randament, context, procesare_ atribute(Nr_valoric).

- transmisie_mecanică(Nr_valoric) :- distanţa_f_mică, atrib_dist, putere_mare, atrib_putere, cuplu_mare, atribut_cuplu, randament_f_mare, atribut_randament, context, procesare_ atribute(Nr_valoric).

- transmisie_mecanică(Nr_valoric) :- distanţa_f_mică, atrib_dist, putere_f_mare, atrib_putere, cuplu_mare, atribut_cuplu, randament_f_mare, atribut_randament, context, procesare_ atribute(Nr_valoric).

- transmisie_mecanică(Nr_valoric) :- distanţa_mică, atrib_dist, putere_f_mică, atrib_putere, cuplu_mare, atribut_cuplu, randament_f_mare, atribut_randament, context, procesare_ atribute(Nr_valoric).

- transmisie_mecanică(Nr_valoric) :- distanţa_mică, atrib_dist, putere_mică, atrib_putere, cuplu_mare, atribut_cuplu, randament_f_mare, atribut_randament, context, procesare_ atribute(Nr_valoric).

- transmisie_mecanică(Nr_valoric) :- distanța_mică, atrib_dist, putere_medie, atrib_putere, cuplu_mare, atribut_cuplu, randament_f_mare, atribut_randament, context, procesare_ atribute(Nr_valoric).

- transmisie_mecanică(Nr_valoric) :- distanța_mică, atrib_dist, putere_mare, atrib_putere, cuplu_mare, atribut_cuplu, randament_f_mare, atribut_randament, context, procesare_ atribute(Nr_valoric).

- transmisie_mecanică(Nr_valoric) :- distanța_mică, atrib_dist, putere_f_mare, atrib_putere, cuplu_mare, atribut_cuplu, randament_f_mare, atribut_randament, context, procesare_ atribute(Nr_valoric).

- transmisie_hidraulică(Nr_valoric) :- distanța_f_mare, atrib_dist, putere_medie, atrib_putere, cuplu_mare, atribut_cuplu, randament_mic, atribut_randament, context, procesare_ atribute(Nr_valoric).

- transmisie_hidraulică(Nr_valoric) :- distanța_f_mare, atrib_dist, putere_mare, atrib_putere, cuplu_mare, atribut_cuplu, randament_mic, atribut_randament, context, procesare_ atribute(Nr_valoric).

- transmisie_hidraulică(Nr_valoric) :- distanța_f_mare, atrib_dist, putere_f_mare, atrib_putere, cuplu_mare, atribut_cuplu, randament_mic, atribut_randament, context, procesare_ atribute(Nr_valoric).

- transmisie_electrică(Nr_valoric) :- distanța_f_mare, atrib_dist, putere_mare, atrib_putere, cuplu_mic, atribut_cuplu, randament_mare, atribut_randament, context, procesare_ atribute(Nr_valoric).

- transmisie_electrică(Nr_valoric) :- distanța_f_mare, atrib_dist, putere_f_mare, atrib_putere, cuplu_mic, atribut_cuplu, randament_mare, atribut_randament, context, procesare_ atribute(Nr_valoric).

- [transmisie_electrică + transmisie_mecanică](Nr_valoric) :- distanța_f_mare, atrib_dist, putere _mare, atrib_ putere, cuplu_mare, atribut_cuplu, randament_mare, atribut_randament, context, procesare_atribute(Nr_valoric).

- [transmisie_electrică + transmisie_mecanică](Nr_valoric) :- distanța_f_mare, atrib_dist, putere_f_mare, atrib_putere, cuplu_mare, atribut_cuplu, randament_mare, atribut_randament, context, procesare_atribute(Nr_valoric).

Numărul de fapte/reguli, necesare pentru decizia de tip, este dat de relația (4.2). Se observă că Nr_reguli necesare, pentru luarea deciziei de tip, cu V tipuri, este excesiv de mare.

$$Nr\_fapte = (N_{atrib1}*N_{atrib2}*\ldots*N_{atribX})_{Sp1}+(N_{atrib1}*N_{atrib2}*\ldots*N_{atribY})_{Sp2}+ + \ldots+(N_{atrib1}*N_{atrib2}*\ldots*N_{atribZ})_{SpV} \quad (4.2)$$

unde: N_{atribX} este numărul de atribute pentru criteriul X, Spv este soluția parțială v.

De exemplu, pentru decizia de tip, cu trei tipuri înscrise în baza de cunoștințe și patru criterii ca în tabelul T4.1, rezultă un număr de 78 de fapte.

Tipul de transmisie	Distanţa dintre arbori	Puterea	Cuplul	Randamentul transmisiei
transmisie mecanică	f_mică mică	f_mică mică medie mare f_mare	f_mic mic mediu mare f_mare	f_mare
transmisie hidraulică	f_mică mică mare f_mare	medie mare f_mare	mare f_mare	mic
transmisie electrică	mare f_mare	mare f_mare	mic	mare

T4.1

La creşterea numărului de criterii, numărul de reguli, pentru cinci valori pe criteriu, poate să atingă maxim 5^{10}. In acest caz, pentru N tipuri se poate ajunge până la $N*5^{10}$ reguli. Această reprezentare extensivă este nepractică şi deci inacceptabilă.

4.3.2. Reducerea numărului necesar de fapte.

Reducerea drastică a numărului de fapte trebuie să se facă cu păstrarea fineţei deducţiei. Acest lucru este posibil, într-un sistem de producţii, prin introducerea funcţiei succesor (prezentată în paragraful 3.1.2). Funcţia succesor este îndeplinită de predicate care operează asupra claselor de atribute. Obţinerea claselor se face prin gruparea variantelor valorice ale atributelor, în atribute_tip. De exemplu:

- putere_tr_mec:-f_mică;mică;medie;mare.
- cuplu_tr_hidro:-mediu;mare;f_mare.
- dist_tr_electrica:-mare;f_mare.

In acest caz, deducţia din exemplul precedent recurge la următoarele reguli:

- transmisie_mecanică(Nr_valoric) :- distanţa_tr_mec, atrib_dist, putere_tr_mec, atrib_putere, cuplu_tr_mec, atribut_cuplu, randament_tr_mec, atribut_randament, context, procesare_atribute(Nr_valoric).

- transmisie_hidraulică(Nr_valoric) :- distanţa_tr_hidro, atrib_dist, putere_tr_hidro, atrib_putere, cuplu_tr_hidro, atribut_cuplu, randament_tr_hidro, atribut_randament, context, procesare_atribute(Nr_valoric).

- transmisie_electrică(Nr_valoric) :- distanţa_tr_electr, atrib_dist, putere_tr_electr, atrib_putere, cuplu_tr_electr, atribut_cuplu, randament_tr_electr, atribut_randament, context, procesare_atribute(Nr_valoric).

Pentru exemplul din tabelul T4.1, numărul necesar de fapte se reduce la 12.

4.4. ALEGEREA SPECIMENULUI DE TRANSMISIE MECANICA

Alegerea tipului de transmisie mecanică urmează acelaşi procedeu, derulat pe alt nivel, al ierarhiei transmisiilor. In această etapă, se poate aplica proprietatea de moştenire cu scopul, de

a simplifica dialogul cu utilizatorul şi de a completa elementele pe care acesta nu le cunoaşte. Scrierea faptelor porneşte de la un tablou sintetic, cu criteriile adoptate şi cu valorile lingvistice ale acestora. Exemplul referitor la transmisiile mecanice este prezentat în tabelul T4.2.

	Curea lată	**Curea trapezoidală**	**Curea rotundă**	**Curea sincronă**	**Curea multiplă**	**Lanţ**	**Angrenaj**	**Roţi de fricţiune**
Puterea	foarte mică mică, medie, mare	mică, medie, mare	foarte mică	mică, medie, mare	foarte mică mică, medie, mare, foarte mare	mică, medie, mare	foarte mică mică, medie, mare, foarte mare	foarte mică mică
Viteza	mică, medie, mare, foarte mare	mică, medie	mică	mică, medie	mică, medie, mare	mică, medie	mica, medie, mare, foarte mare	medie, mare
Raport de transmitere	mic, mediu, mare	mic, mediu, mare	mic, mediu	mic, mediu	mic,mediu, mare	mic, mediu, mare	mic, mediu, mare, foarte mare	mic, mediu
Precizie cinematică	scăzută	scăzută	scăzută	bună	scăzută	bună	bună, foarte bună	scăzută
Distanţă dintre axe	mică,medie mare, foarte mare	mică, medie	foarte mică mică	mică, medie	mică, medie	mică, medie, mare	foarte mică	foarte mică
Nivel de zgomot	foarte mic	mic, mediu	foarte mic	mic, mediu	mic, mediu	mare, foarte mare	mediu, mare	mediu, mare
Nivel de vibraţii	foarte mic, mic	mediu	foarte mic	mediu	mediu	mare, foarte mare	mediu	mediu
Temperatură	mică, normală, medie	mică, normală, medie	normală, medie	mică, normală, medie, mare	mică, normală, medie	foarte mică mică,normală,medie mare, foarte mare extrem de mare	foarte mică mică,normală,medie mare, foarte mare extrem de mare	mică, normală, medie, mare
Poluare mediu cu praf	mică	mică	foarte mică	mică	mică	mică, medie, mare	mică, medie, mare, foarte mare	mică, medie, mare
Umiditate mediu	foarte mică	mică	foarte mică	mica, medie	mică, medie	mică, medie, mare	mică, medie, mare, foarte mare	mică, medie, mare
Poluare cu petrol	nu	nu	nu	nu	nu	da	da	da
Poluare cu acizi	nu	nu	nu	da	nu	nu	da	nu
Gabarit	mare	mare	mic	mediu	mediu	mare	mic	mediu
Răspândire	mare	mare	mică	mare	medie	medie	foarte mare	mică
Preţ	mic, mediu	mic	mic	mediu, mare	mediu, mare	mediu	mediu, mare, foarte mare	mare, foarte mare
Intreţinere	simplă	simplă	simplă	simplă	simplă	dificilă	medie	medie
Masa specifică	mică	mică	mică	mică	mică	mare, foarte mare	mare	mare

T4.2

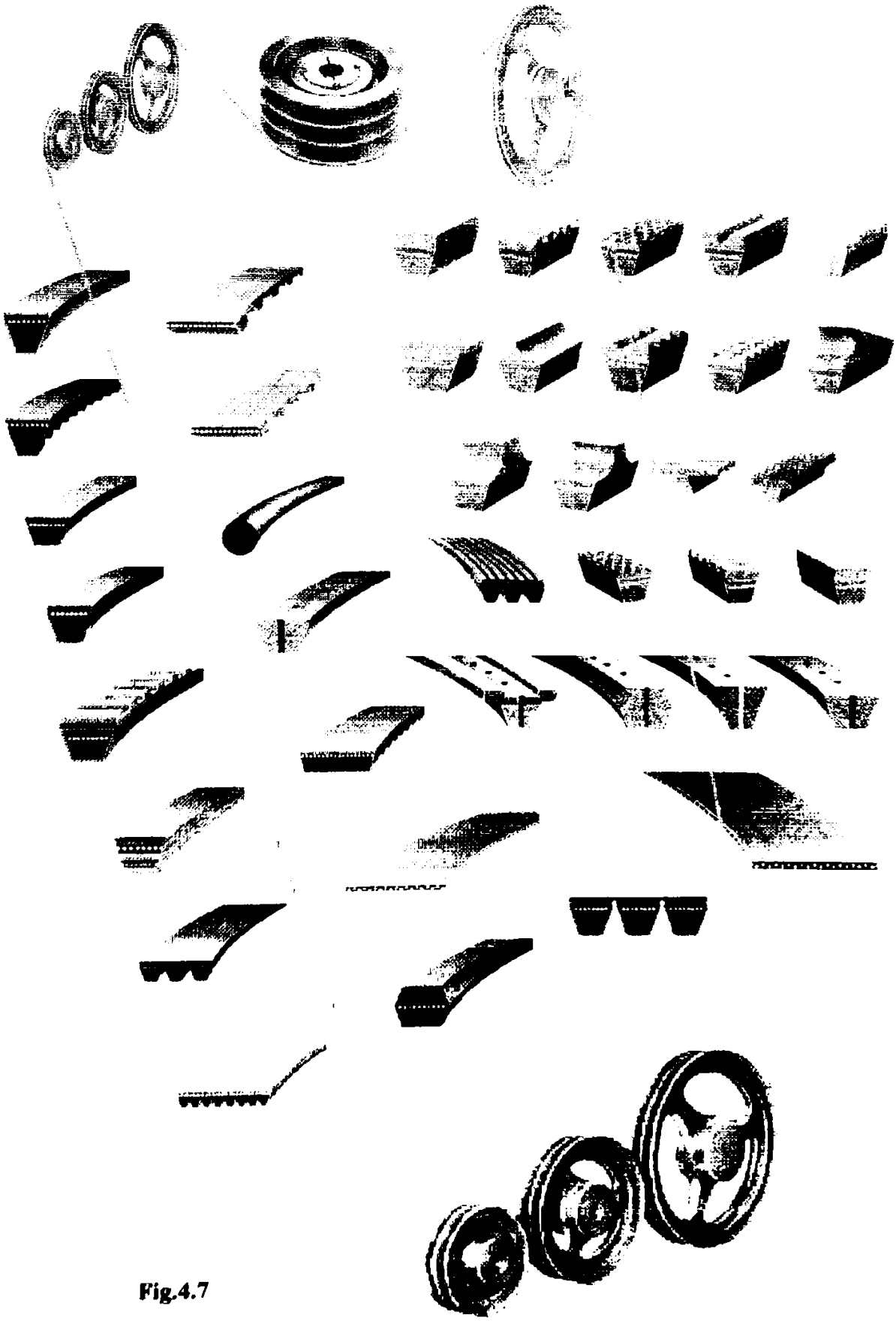


Fig.4.7

Procesul de inferenţă conduce alegerea pe nivelele inferioare ale structurii, până la specimen. In Fig.4.7 este prezentat un obiect generalizat al transmisiei prin curea, ce conţine principale profile, care se utilizează astăzi.

Alegerea specimenului, este o operaţie, care în mod tradiţional, este inclusă în aşa numita etapă de dimensionare. De exemplu, dimensionarea unei transmisii prin curele începe prin alegerea dimensiunii, codului, profilului. Algoritmii clasici, cuprinşi în toate cataloagele de curele de transmisie, conţin nomograme de alegere a dimensiunii, ca în Fig.4.8. O astfel de nomogramă, poate fi extrem de uşor reprezentată prin fapte.

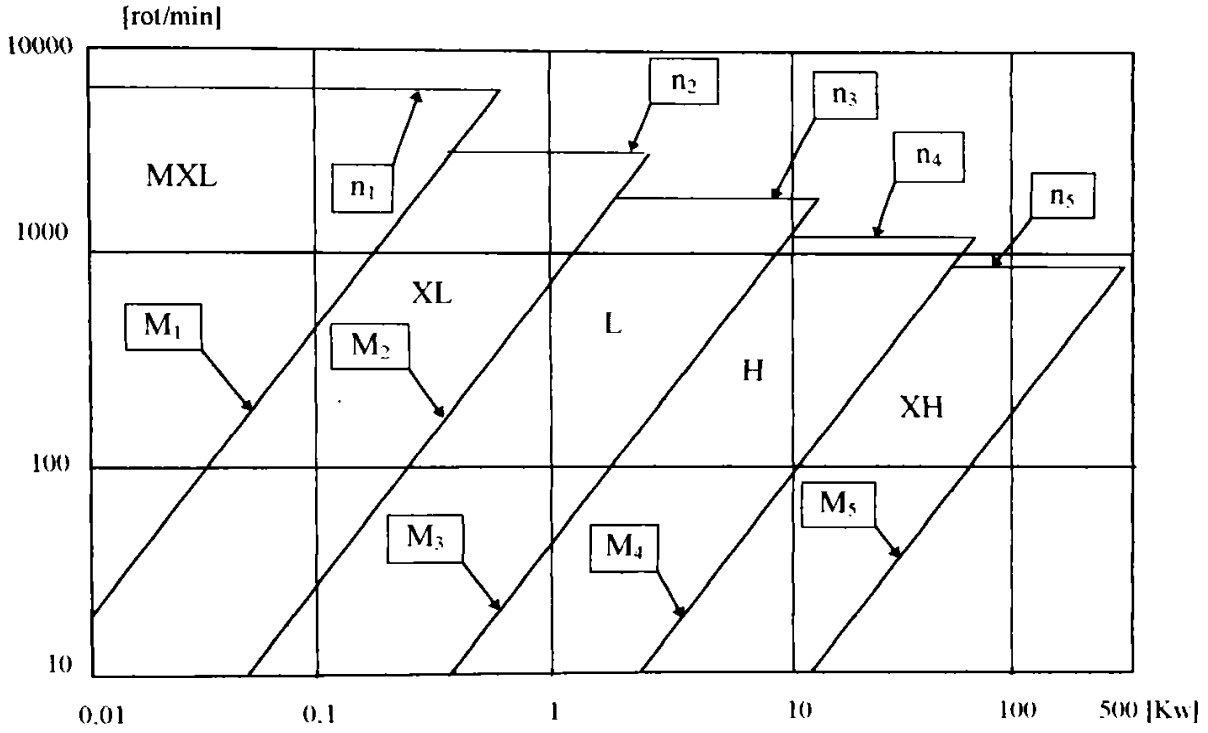


Fig.4.8

Faptele care descriu complet toate zonele nomogramei sunt:

MXL :- $M > M_0$, $M \le M_1$, $N \le n_1$.

XL :- $M > M_1$, $M \le M_2$, $N \le n_2$.

L :- $M > M_2$, $M \le M_3$, $N \le n_3$.

H :- $M > M_3$, $M \le M_4$, $N \le n_4$.

XH :- $M > M_4$, $M \le M_5$, $N \le n_5$.

Exprimarea concretă se face cu predicatul specimen, care are următoarea structură:

specimen (MXL, M_0 ,M_1 , n_1).

specimen (XL, M_1 ,M_2 , n_2).

specimen (L, M_2 ,M_3 , n_3).

specimen (H, M_3 ,M_4 , n_4).

specimen (XH, M_4 ,M_5 , n_5).

Căutarea specimenului, în regim de backtraking, se face cu predicatul caută_specimen, pentru care se scrie următoarea clauză:

caută_specimen(S) :- read(M), read(N), specimen(M1, M2, Nr), M > M1,
M ≤ M2, N ≤ Nr.

Predicatul returnează codul specimenului, care este mărimea profilului de curea, obținut prin instanțiere și backtraking. Restul calculelor de dimensionare se pot efectua, fie în limbajul declarativ, fie prin apelarea la un cod scris, într-un limbaj procedural. Mediul de programare PROLOG (toate dialectele), acceptă o interfață cu limbajele C, respectiv PASCAL.

Etapele procesului de proiectare, care conțin calcule de dimensionare și calcule de verificare, se recomandă să fie implementate în module de program, scrise în limbaje procedurale, ce pot apelate de Sistemul Expert.

Datele de catalog, normele și standardele pot fi introduse în baze de date specifice mediului PROLOG, sau pot fi introduse în baze de date consacrate. Acest lucru este posibil, deoarece, mediul PROLOG admite importul de date din fișierele bazelor de date consacrate.

4.5. ALGORITM DE CALCUL PENTRU TRANSMISIILE PRIN CUREA

Efectuarea unui set de calcule, care urmează, în proiectarea tradițională, așa numitul mers de calcul, nu este proprie unui limbaj declarativ. Dacă totuși este nevoie de așa ceva, atunci se apelează la o procedură scrisă într-un alt limbaj (evident procedural), compatibil cu cel al sistemului. Limbajul PROLOG acceptă interfața cu limbajele C, respectiv PASCAL.

Rezolvarea algoritmului de calcul propus în [96] se poate face în succesiunea de operații care este prezentată în continuare. Prima etapă a calculului, cea de introducere a datelor inițiale, își propune să stabilească care este valoarea puterii de calcul.

$$P_B = P_N * C_B \; ; \; P_N = P_1 = P_2 / \eta_{12} \; ; \; P_1 = T_1 * \omega_1$$

$$C_B = 1 + C_{Typ} (0{,}075\, C_{ab} + 0{,}1\, C_{an} + 0{,}1\, C_t)$$

unde: P_B , P_N - puterea de calcul și nominală la arborele motor (de la Berchnungsleistung)

C_B - factor de regim (de la Betriebsfaktor)

C_{Typ} - factor de tip

C_{ab} - factorul mașinii antrenate (de la Abtriebsmaschine)

C_{an} - factorul mașinii de acționare (de la Antriebsmaschine)

C_t - durata zilnică de funcționare (de la Betriebsdauer je Tag - zi)

$P_{1,2}$ - puterile nominale la arborii motor și antrenat;
η_{12} - randamentul transm;
T_1 - momentul motor;
ω_1 - viteza unghiulară a arborelui motor.

Implementarea în limbaj declarativ a primei etape folosește 7 predicate și este următoarea:

stabilește_putere (Putere_de_calcul):- read(Putere_nominală),factor_de_regim(Factor),
Putere_de_calcul = Putere_nominală * Factor.

factor_de_regim(F):-alege_factor_de_tip(C_tip),alege_abtrieb(C_ab),alege_antrieb(C_an),
alege_durată(C_t), F = 1+ C_tip * (0.075 * C_ab + 0.1 * C_an + 0.1 * C_t).

alege_factor_de_tip(T):-alege_tip(Tip),caută_în_listă_f_tip(Tip,T).

caută_în_listă_f_tip(Tip,T):- Tip="lată",T=1;Tip="trapezoidală",T=1;Tip="multiplă",Tip=1.

caută_în_listă_f_tip(Tip,T):- Tip="sincronă cu profil trapez",T=1.6.

caută_în_listă_f_tip(Tip,T):- Tip="sincronă cu profil rotund",T=2.

alege_durată(C_t):-.....

alege_a.....(C_...):-.....

alege_antrieb(C_an):-.....

C_{Tip}	Tip de profil
1	lată, trapezoidală, multiplă
1.6	sincronă cu profiltrapez
2	sincronă cu profil rotund

C_t	Durata zilnică de funcţionare
0	< 10 h
1	10 la 16 h
2	> 16 h

C_{an}	Regim	Exemple
0	uniform	Electromotoare cu moment de pornire mic (< 1.5 T_N), turbine, motoare cu ardere internă cu mai mult de 8 cilindri
1	neuniform	Electromotoare cu moment de pornire normal, (1.5-2.5 T_N), motoare cu ardere internă cu 4 la 6 cilindri
2	foarte neuniform	Electromotoare cu moment de pornire mare (> 2.5 T_N), motoare cu ardere internă cu mai puţin de patru cilindri

C_{ab}	Regim	Exemple
0	foarte uşor	maşini de scris, tehnică -audio,-foto, -film, aparatură electro-casnică uşoară, etc.
1	uşor	maşini pentru prelucrarea lemnului ventilatoare, pompe, etc.
2	mediu	maşini-unelte cu solicitări medii, ventilatoare mari, compresoare, maşini textile, etc.
3	greu	pompe cu piston, ventilatoare, ventilatoare radiale cu moment de pornire mare, calandre, extrudere, etc.
4	foarte greu	toate maşinile cu moment de pornire mare, concasoare, laminoare, etc.

Continuarea calculului se face prin folosirea unui predicat de apelare a unei proceduri scrise în limbaj C. Predicatul trebuie să fie global.

global predicates

calculează(symbol, real, real, real, real, real, real, real, int, ...):-(i,i,i,i,i,i,i,o,o,.....) language c

simbol profil

$(d_w / d_p)_{min}$; z_{min}

puterea P_B

raportul i

e_0

n_{an}

d_{ab}

d_{an}

$l_{w,0}$

z

t_c

l_f /2

F_c

void calculează_0(real putere, real raport, real *

{

<< corpul procedurii>>

$$L_w = 2a_0 \cos\alpha + \frac{d_g}{2}\beta_g + \frac{d_k}{2}\beta_k \in \{L_w\}_{STAS}$$

$$L_w = 2a_0 \cos\alpha + \frac{\pi}{2}(d_g + d_k) + \alpha(d_g - d_k)$$

$d_{an} \geq (d_p)_{min} \in \{d_i\}_{STAS}$

$d_{ab} = i * d_{an} \in \{d_i\}_{STAS}$

$d_{an} = z_{an} * t / \pi$

$d_{ab} = z_{ab} * t / \pi$

$$\alpha = \arcsin\left(\frac{d_g - d_k}{2a_0}\right)$$

$\beta_k = \pi - 2\alpha$

$\beta_g = \pi + 2\alpha$

$v = \pi \; n_{an} \; d_{an} \; 10^{-3} / 60$

$$f_B = \frac{2v}{L_w} 10^3$$

<<corpul procedurii>>

$$F_{u,B} = \frac{P_B 10^3}{v}$$

$$b_{erf} = \frac{F_{u,B}}{f_{u,zul}} \in \{b\}_{STAS}$$

$f_{u,zul,p} = 1/2\ d_k\ p_{zul}\ C_w$
$f_{u,zul,z} = h_z\ (\sigma_{zul} - \sigma_F - \sigma_B)\ C_w$

$$C_w = 1 - e^{-\mu_r \beta_k}$$

$$\sigma_F = \frac{\rho v^2 h_g}{1000 h_z^2}$$

$$\sigma_B = \frac{h_z E}{d_k + h_g}$$

$E=1000\ N/mm^2$ $\sigma_{zul} = 42\ N/mm^2$
$p_{zul} = 0{,}42\ N/mm^2$ $\rho = 1{,}1\ g/cm^2$
$\mu = 0{,}5$ $h_g = h_z + 1.5$

$$z_{erf} = \frac{F_{u,B}}{F_{u,zul}} \quad ; \quad z > z_{erf}$$

$F_{u,zul} = K(F_{u,b} + \Delta F_{u1} + \Delta F_{u2})\ 10^{-3}$
$F_{u,b} = 2[C_1 - C_2/d_p - C_3(2v)^2 - C_4 \log(2v)]$

$$\Delta F_{u1} = 2C_4 \log\left(\frac{2}{1 + 10^{Q}}\right)$$

$$Q = \frac{C_2}{C_4} \frac{d_k - d_g}{d_g d_k}$$

$$\Delta F_{u2} = 2C_4 \log\left(\frac{L_w}{L_{wref}}\right)$$

$$K = 1{,}25(1 - 5^{(\beta_k/\pi)})$$

$$b_{erf} = b_{bez}\left(\frac{F_{u,B}}{b_{bez} k_z f_u}\right)^{(1/1{,}14)}$$

$k_z = 1 - 0.2\ (6 - z_e)$ pentru $z_e < 6$ $k_z = 1$
$z_e = z_k\ \beta_k / 2\pi$
$b_{st} \geq b_{erf}$
$F_{u,b} + b_{st}\ m'\ v^2 \leq b_{st}\ f_u$

<<corpul procedurii>>

Pretensionarea curelelor late

$$F_{t,0} = \frac{F_{u,B}}{2} \frac{e^{\mu_r \beta_k} + 1}{e^{\mu_r \beta_k} - 1} + \frac{h_g b_{erf} v^2}{1000}$$

$$F_{w,0} = 2F_{t,0} \sin\left(\frac{\beta_k}{2}\right)$$

$$\varepsilon_0 = \frac{F_{t,0}}{E h_z b_{erf}} 100\%$$

Pretensionare curelelor trapezoidale şi multiple

$$F_{r,0} = \frac{F_{u,B}}{2}\left(\frac{2{,}5}{K} - 1\right) + m v^2$$

$$F_{w,0} = 2F_{t,0} \sin(\beta_k / 2)$$

Pretensionarea curelelor sincrone

$$F_{t,0} = \frac{F_{u,B}}{2} + m_1' v^2$$

$$F_{w,0} = 2F_{t,0} \sin(\beta_k / 2)$$

<<corpul procedurii>>

Calculul săgeții de pretensionare

$$t_e = \frac{l_f}{100}\left(K_1 - K_2 \log F_t\right)$$

$$l_f = \frac{a}{\cos\alpha}$$

Măsurarea alunecării

$$s = \left(1 - \frac{i\, n_{ab}}{n_{an}}\right) 100\%$$

Profil	F_e [N]	K_1	K_2	Domeniu tensiune [N]
SPZ, XPZ	25	10.34	3.47	200-500
SPA, XPA	50	12.84	3.92	300-1000
SPB, XPB	75	17.53	5.45	400-1200
SPC, XPC	125	17.10	4.82	400-1400
Z/10, ZX/X10	25	6.92	2.33	140-330
A/13, AX/X13	25	8.65	2.72	180-450
B/17, BX/X17	50	11.62	3.42	200-630
C/22, CX/X22	100	15.11	4.22	360-1040
D/32	150	14.96	3.98	350-1500
PH	3	6.14	2.81	20-70
PJ	5	8.15	3.38	30-100
PK	8	9.44	3.77	40-185
PL	10	9.84	3.82	50-200
PM	25	11.10	3.67	200-550

}

Predicatul *calculează* transmite procedurii *calculează 0*, scrisă în limbaj C, toate valorile argumentelor necesare derulării calculului procedural. Rezultatele sunt preluate de același predicat *calculează*. Distincția dintre parametri de intrare și cei de ieșire este efectuată în declarația globală a predicatului, prin *flow pattern*.

Datele de ieșire din procedură sunt folosite în continuare de către modulul de program, care a apelat procedura.

Notațiile folosite în algoritmul de calcul sunt conforme cu VDI 2758 și cu fig.4.9. Rularea modulului de program pentru alegerea și dimensionarea transmisiei mecanice (Anexa 2) este prezentată în Anexa 4.

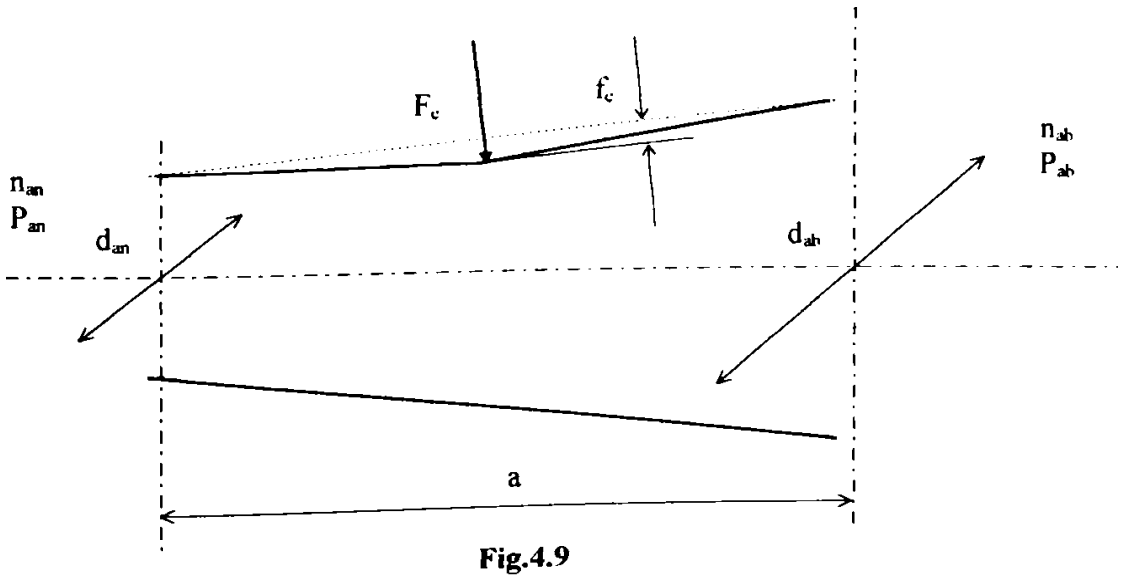


Fig.4.9

5. METODA DE PROIECTARE A MEDIULUI SISTEM EXPERT

5.1. ALEGEREA LIMBAJULUI ADECVAT SCOPULUI

5.1.1. Etapele proiectării mediului software

Dezvoltarea unui Sistem Expert, dedicat unei aplicaţii particulare, poate să urmeze două căi distincte. Una dintre căi constă în proiectarea completă a mediului şi a bazelor de cunoştinţe necesare funcţionării acestuia. Pe această cale au fost realizate toate sistemele mari, consacrate, aflate în funcţie, începând cu DENDRAL şi MYCIN. Proiectarea completă a unui Sistem Expert cere un volum considerabil de muncă realizat de o echipă multidisciplinară de specialişti. Efortul financiar şi uman cerut de această cale de rezolvare i-a determinat pe proiectanţii de software să găsească o cale mai simplă.

A doua cale de dezvoltare a sistemului, porneşte de la un mediu Shell şi obţine produsul dorit, prin introducerea unui set de cunoştinţe specifice aplicaţiei concrete. In ambele cazuri sistemul evoluează prin adăugarea de cunoştinţe noi.

Motivul, pentru care lucrarea a abordat prima cale de rezolvare, este lipsa unui mediu Shell potrivit. Proiectarea ţine cont de toate atributele cerute unui Sistem Expert folosit în proiectare, prezentate în paragraful 2.6 şi debutează cu alegerea limbajului cel mai adecvat scopului.

Proiectarea mediului software are următoarele etape:

- Stabilirea atributelor necesare unui Sistem Expert folosit în proiectare;
- Stabilirea cerinţelor impuse limbajului de programare;
- Analiza comparativă a unui grup de limbaje de programare;
- Alegerea limbajului de programare adecvat;
- Conceperea sau adoptarea schemei de structură a mediului Sistem Expert;
- Proiectarea modulelor de program care realizează funcţiile blocurilor din schema de structură;
- Ordonarea cunoştinţelor;
- Introducerea cunoştinţelor în baza de cunoştinţe;
- Dezvoltarea bazei de cunoştinţe.

5.1.2. Stabilirea cerinţelor impuse limbajului de programare

Descrierea realităţii, cu ajutorul calculatoarelor, parcurge mai multe nivele de abstractizare. La creearea oricărei teorii, clasele de elemente (fenomene şi obiecte) sunt obţinute prin abstragerea detaliilor, care individualizează fiecare membru. Această pierdere poate fi compensată prin puterea descriptivă şi predicativă a unei teorii valide. Abstractizarea, este un

instrument esențial pentru înțelegerea complexității lumii şi pentru organizarea universului cunoaşterii, ca structură validă. Abstractizarea este un proces continuu, care se desfăşoară pe mai multe nivele, aşa cum se observă în Fig.5.1.

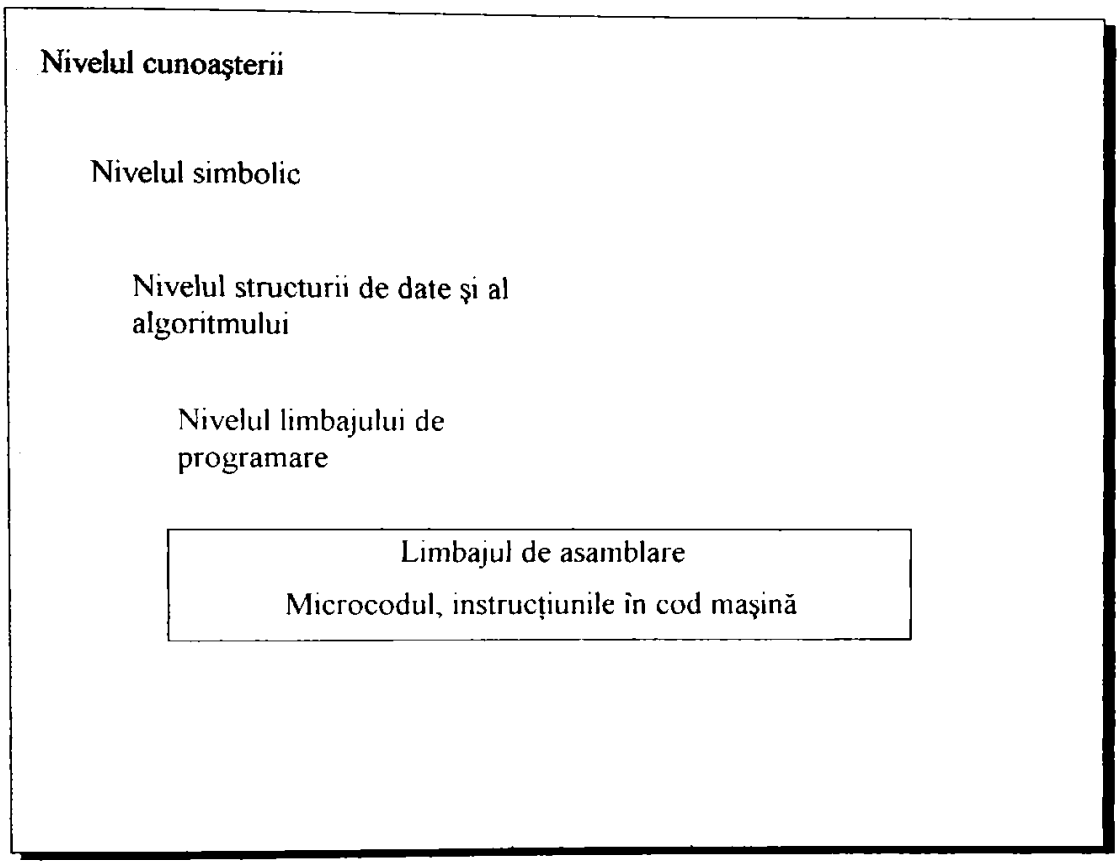


Fig.5.1.

Nivelul cunoaşterii conține elementele cel mai puțin abstractizate; el este adiacent necunoscutului. Un limbaj trebuie să fie adecvat implementării structurilor de nivel înalt şi să fie un instrument eficient pentru dezvoltarea programelor. Calitățile cerute unui limbaj, pentru a fi bun la dezvoltarea Sistemelor Expert sunt:

- să admită calculul simbolic;
- să permită un control flexibil;
- să admită metodele programării exploratorii;
- să admită amânarea propagării şi conectării restricțiilor;
- să posede o semantică bine definită şi clară.

Există mai multe căi de organizare a cunoaşterii la nivel simbolic, toate acestea fiind implementate ca acțiuni asupra cadrelor (conținutului) simbolurilor. Ipotezele sistemului simbolic caracterizează, pe de o parte, tendința generală a cercetării, în domeniul acestor limbaje, iar pe de altă parte, necesitatea unor limbaje, care să simplifice implementarea operațiilor simbolice. Cea mai importantă cerință a unui limbaj, pentru Inteligența Artificială, este capacitatea de a simplifica creerea structurilor de simboluri arbitrare şi a operațiilor

efectuate pe aceste structuri. Un limbaj de programare pentru Inteligența Artificială trebuie să ofere, printre altele, următoarele facilități:

- modularitatea codului (interacțiunile dintre componentele programului trebuie să fie limitate și clar definite);
- extensibilitatea (abilitatea de a defini construcții lingvistice noi, cu maximum de libertate și flexibilitate);
- structurile de nivel înalt;
- suportul pentru execuția rapidă a prototipului;
- programul sursă lizibil;
- rularea, atât în mod interpretor, cât și în mod compilator (rularea în mod interpretor este utilă în faza de dezvoltare a programului, pentru economisirea timpului necesar recompilărilor succesive, cerute de modificările care sunt introduse);
- suportul software pentru programare exploratorie (un mediu de programare modern trebuie să posede facilități de depanare importante ca: break point, trace, etc).

5.1.3. Analiza comparativă a unui grup de limbaje de programare

O comparație între patru limbaje, care își dispută supremația pentru dezvoltarea Inteligenței Artificiale, este prezentată în tabelul T5.1. De remarcat că, limbajele orientate pe obiecte au fost luate în considerare împreună în coloana numită OOL (Oriented Object Language). Această grupă se referă la limbajele VISUAL C, respectiv VISUAL BASIC.

LIMBAJUL LISP

LISP este primul limbaj destinat Inteligenței Artificiale. A fost creat în anul 1960, de către John McCarthy. Are la bază teoria funcțiilor recursive. Puterea limbajului constă în capacitatea de procesare a listelor. La început LISP a fost foarte mic și simplu. Un program era controlat de un mecanism recursiv și de o condiție simplă. Limbajul a fost completat ulterior cu funcții mai complicate. Cele mai bune dintre ele au devenit parte integrantă a limbajului. LISP a cunoscut o mare diversificare, putând fi menționate mai multe dialecte INTERLISP, MACLISP, FRANZ LISP, ZETA LISP, XLISP.

In anul 1983 *the Defense Advanced Research Projects Agency* a propus un dialect standard al limbajului numit **COMMON LISP**. Dar COMMON LISP a fost criticat pentru faptul că, este excesiv de complicat. Dialectul XLISP este compact și permite programarea orientată pe obiecte.

LIMBAJUL PROLOG

PROLOG este un limbaj neprocedural de programare, dedicat programării logice și constituie un mediu adecvat pentru dezvoltarea sistemelor de Inteligență Artificială. Acesta este centrat pe un mic set de mecanisme fundamentale, format dintr-un model de joc, o structură arborescentă pentru date și un automat de backtracking. Acest set mic conferă mediului de programare o putere și flexibilitate surprinzătoare. PROLOG este specializat pe probleme care implică obiecte, structuri de obiecte și relații între acestea. Este ușor pentru un programator, care folosește acest limbaj, să implementeze reguli generale, de tipul silogismului. PROLOG este astfel creeat încât, să raționeze pe baza regulilor date în spațiul problemei.

T5.1

Caracteristica	PASCAL	LISP	PROLOG	OOL
Modelul de limbaj	Programare procedurală	Programare funcţională	Programare logică	Programare orintată pe obiecte
Tipurile de date	Scalari, matrici, înregistrări	Atomi simbolici şi numerici liste şi structuri de liste proprietăţi şi liste de asociere	Atomi simbolici şi numerici, predicate, liste şi structuri de liste	Atomi simbolici şi numerici, clase de valori şi sloturi, instanţe
Manipularea datelor	Atribuirea; transferul parametrilor prin valoare şi prin referinţă; returnarea valorilor funcţiilor	Returnarea valorilor funcţiilor, transferul parametrilor prin valoare; legarea variabilelor locale şi globale	Legarea variabilelor prin unificare	Transferul de mesaje; instanţierea valorilor locaţiilor în obiecte
Controlul programului	Secvenţial; întreruperi; salturi; recursivitate	Evaluarea funcţiilor evaluarea condiţională, recursivitatea	Căutare tipizată, recursivitate	Invocare a transferului de mesaje
Structura programului	Structură bloc; definiţii de proceduri locale	Funcţii într-un mediu global, blocuri	Reguli şi fapte într-o bază de date	Obiecte într-o ierarhie bazată pe clase, moştenire
Statutul variabilelor	Variabile globale şi variabile locale	Variabile legate, variabile lexicale, variabile libere, domeniul stabilit dinamic	Variabile cu domeniul o singură regulă sau un singur fapt	Valorile legate de locaţiile unui obiect valorile vizibile pentru subclase şi instanţe asupra moştenirii
Modul de interacţiune	Interpretor sau compilator	Interpretor interactiv, compilator	Interpretor interactiv, compilator	Interpretor interactiv, compilator

Prima versiune oficială de PROLOG a fost creeată în anul 1972, de Alain Colmerauer, de la Universitatea din Marsilia. Apoi, a fost dezvoltat de Robert Kowalski şi Maarten van Emden, la Edinburgh. Popularitatea PROLOG-ului se datorează, în mare măsură, eficientelor aplicaţii realizate la Edinburgh de către David Warren, la mijlocul deceniului opt. In anul 1983 Japonia a publicat planurile unui ambiţios program naţional de proiectare şi producţie al calculatoarelor de generaţia a cincea, program pentru care limbajul fundamental a fost ales PROLOG-ul.

Prima implementare remarcabilă a PROLOG-ului pe calculatoare PC a fost varianta 2.0 TURBO PROLOG, realizată de firma Borland. Această variantă dispune de un compilator şi un link-editor, care creează programe compilate, în varianta ".EXE", rapide şi uşor de utilizat.

Incepând din 1984, dezvoltarea limbajului PROLOG a fost preluată de către *Prolog Development Center*, din Copenhaga, care a elaborat varianta PDC 3.20. In ultimul deceniu, dezvoltarea limbajului PROLOG a luat o mare amploare, ajungându-se la dialectul al şaselea, varianta a patra

SICStus PROLOG, V4.0 September 1996; © 1993-1996 Rob Lucas and Keylink, Keylink Computers Ltd., 2 Woodway House, Common Lane Kenilworth, Warwickshire, CV8 2ES, United Kingdom

5.1.4. Alegerea propriu-zisă a limbajului de programare

Caracteristicile cerute SE au fost evaluate de specialiştii în domeniu IA şi au fost prezentate sintetic în capitolul 2. In tabelul T2.8. am concentrat acele atribute, care se referă la SE cuprinse în categoria generică 2, intitulată: *Analiză, Proiectare*. Concluziile de adecvare ale limbajului la aceste cerinţe sunt prezentate în tabelul T5.2.

T5.2

Atributele principale ale SE de proiectare	Punctaj	PROLOG	PASCAL	OOL
Legătura cu limbaje 3g, 4g	4.3	* * *	*	* * *
Legătura cu baze de date	4.3	* * *	* *	* * *
Legătura cu soft-ul special ad hoc	4.3	*	*	* *
Prototip rapid	4.2	* * *	*	* * *
Portabilitate	4.1	* * *	* *	* * *
Implantabilitate	4.1	* * *	* *	* * *
Reguli de producţie	4.1	* * *	-	*
Acces la limbajul de bază	4.1	* * *	*	* * *
Editare; depanare; instruire	4.0	* *	* * *	* * *
Căutare înapoi	4.0	* * *	-	*
Valoare ponderată		111.9	53.9	104

1. Analizând valorile ponderate din tabelul T5.2, rezultă că alegerea limbajului PROLOG pentru dezvoltarea SE este adecvată principalelor cerinţe enunţate de specialiştii în domeniu, respectiv de utilizatorii produselor soft de acest gen.
2. Conjugând concluziile din tabelul T5.2, cu cele din tabelul T5.1, rezultă că majoritatea calităţilor cerute unui limbaj pentru dezvoltarea unui SE sunt deţinute de PROLOG, împreună cu limbajele notate generic cu OOL.

5.2. ELEMENTE DE PROGRAMARE IN LIMBAJUL PROLOG

Programarea în PROLOG foloseşte conceptele specifice programării logice, unificarea şi rezoluţia. Inexistenţa atribuirii este posibilă prin faptul că, transmiterea de valori pentru variabile se face pe baza unor legături temporare, care în final, după terminarea lanţului deductiv, sunt dezlegate. PROLOG-ul este, în esenţă, un demonstrator de fapte.

Un program în PROLOG este o descriere a unei probleme, prin descrierea unor **obiecte** şi a **relaţiilor** dintre ele. Descrierea este alcătuită din trei componente:

- nume şi structuri de obiecte implicate în problemă;
- nume de relaţii între obiecte;
- fapte şi reguli care descriu aceste relaţii.

5.2.1. Structura tip a unui program

Structura completă, a unui program în PROLOG cuprinde mai multe secțiuni, după cum urmează:

- **compiler options** opțiuni compilator (sunt date la începutul programului);
- **constants** secțiunea constante (poate conține 0 sau mai multe definiții);
- **domains** este secțiunea în care utilizatorul definește tipul obiectelor și domeniile nestandard folosite în program. (poate conține 0 sau mai multe definiții);
- **database** este secțiunea în care programatorul definește o bază de cunoștințe dinamică, predicatele și tipurile argumentelor. In această bază de cunoștințe se pot memora numai faptele care provin de la aceste predicate. (poate conține 0 sau mai multe predicate database);
- **predicates** un predicat este declarat prin nume și prin domeniile argumentelor sale (poate conține 0 sau mai multe definiții);
- **goal** se definește scopul programului (poate conține 0 sau 1 scop);
- **clauses** se înscriu faptele și regulile (poate conține 0 sau mai multe clauze);

In mod obișnuit, un program necesită cel puțin o secțiune de predicate și una de clauze. Secțiunea **goal** poate să lipsească, dar pentru generarea unui program executabil, de sine stătător, aceasta trebuie să existe în textul sursă. In majoritatea cazurilor, în secțiunea **domains** este necesar să se declare liste, obiecte compuse și nume.

Tehnica programării modulare necesită declarații globale, astfel secțiunile, **domains, predicates** și **database** pot primi prefixul **global**, indicând faptul că declarațiile respective, afectează global mai multe module de program.

Un program poate conține mai multe secțiuni **domains, predicates, database** sau **clauses** cu condiția respectării următoarelor restricții:

- **Constants, domains** și **predicates** trebuie definite înainte de utilizare.
- In decursul compilării trebuie întâlnit un singur **goal** (scop), acesta poate fi plasat oriunde după secțiunea **predicates**.
- Toate clauzele, care descriu același predicat, trebuie grupate într-o singură secvență.
- Toate declarațiile globale trebuie plasate înaintea declarațiilor locale.
- Secțiunea **database** poate fi numită, dar numele dat poate să apară o singură dată în program, deoarece numele implicit este **dbasedom**; în progam poate exista numai o singură secțiune **database** fără nume.

5.2.2. Definirea domeniilor *Standard Domains*

Atât mediul TURBO PROLOG, cât și mediul PDC operează cu șase tipuri standard de domenii, care au următoarele definiții:

- **char** tipul caracter, se scrie inclus între apostroafe, ex: 's', 'w' etc.;
- **integer** tipul întreg, cuprins între -32768 şi 32767;
- **real** tipul real, care se scrie cu punct zecimal sau se poate scrie exponenţial, ex: 43e -12, -5e-10, 1.25; domeniul permis este între $+/-1e^{-307}$ şi $+/-1e^{308}$. Intregii sunt convertiţi automat în numere reale, când este necesar;
- **string** tipul şir de caractere; este o secvenţă de caractere scrisă între ghilimele;
- **symbol** tipul simbol; este o secvenţă de litere, cifre şi caracterul de subliniere, care începe întotdeauna cu o literă mică. Tipurile **string** şi **symbol** sunt interschimbabile cu observaţia că sistemul operează diferit cu ele;
- **file** tipul fişier.

Domeniul *List Domains*

Este un domeniu de tip listă de elemente. Exemplu: lista_clase = componente* unde lista_clase este o listă de elemente din domeniul componente. Acest domeniu, poate fi definit de utilizator, sau poate fi standard.

Domeniul *File Domain*

Acest domeniu trebuie definit, atunci când, este necesară referirea prin nume simbolice la fişiere, altele decât cele predefinite. Un program poate avea un singur domeniu de acest tip, care trebuie să se numească **file**.

Domenii speciale predefinite

- **dbasedom** domeniu generat pentru termenii din bazele de date globale;
- **bt_selector** selector de arbore binar;
- **db_selector** selector de bază de date exterioară, definită de utilizator;
- **place** precizează locul ocupat de fişier: in_memory, in_file, in_ems;
- **accesmode** precizează modul de acces: read, readwrite;
- **denymode** precizează modul interzis de acces: denywrite, denyall;
- **ref** domeniu generat pentru numerele de referinţă ale bazelor de date;
- **file** nume simbolic de fişier;
- **reg** registre folosite cu bios: reg(AX, BX, CX, DX, SI, DI, DS, ES);
- **bgi_ilist** listă de întregi folosiţi în predicatele BGI.

Domeniul *Reference Domains*

Un domeniu de referinţă este acela, care poate purta variabile nelegate, ca argumente de intrare. Pentru a declara un domeniu de referinţă, se aplică prefixul **reference**, declaraţiilor de domeniu. Atunci când se declară de referinţă un domeniu compus, toate subdomeniile sale devin automat domenii de referinţă.

Ex: domains

refint = reference integer

term = reference int(refint); symb(refsymb).

5.2.3. Declararea predicatelor

Introducerea definiţiilor de predicate se face prin cuvântul cheie **predicates**. Cu excepţia predicatelor predefinite, toate celelalte trebuie definite ca formă. TURBO PROLOG şi PDC admit predicate multiple (cu mai mult de două argumente). Sunt admise declaraţii multiple pentru acelaşi predicat. In cazul declaraţiilor multiple, acestea trebuie să fie grupate, iar predicatul să aibă acelaşi număr de argumente, în toate declaraţiile. Predicatele pot fi declarate deterministe, cu prefixul **determ**, sau nedeterministe, cu prefixul **nondeterm**.

5.2.4. Definirea clauzelor

Regulile şi faptele sunt cuprinse în secţiunea **clauses** şi constituie baza de date a programului. La această descriere preliminară se pot adăuga şi următoarele consideraţii:

1. Programele în PROLOG pot fi extinse prin simpla adăugare de noi clauze.
2. Clauzele sunt de trei tipuri: fapte, reguli şi întrebări.
3. Faptele declară lucruri, care sunt întotdeauna şi necondiţionat adevărate.
4. Regulile declară lucruri, care sunt adevărate, dependent de condiţiile date.
5. Prin intermediul întrebărilor, utilizatorul întreabă programul care lucruri sunt adevărate.
6. Clauzele PROLOG sunt alcătuite dintr-un cap şi un corp. Corpul este o listă de scopuri separate prin virgule. Virgulele sunt înţelese ca şi conjuncţii.
7. Faptele sunt clauze care au corp vid. Intrebările au numai corp. Regulile au şi cap şi corp.
8. Pe parcursul calculului o variabilă poate fi substituită cu un alt obiect. Operaţia se numeşte instanţiere.
9. Variabilele sunt asumate ca fiind cuantificate universal. Acestea sunt citite astfel: "Pentru toate...".
10. Obiectele simple în PROLOG sunt atomii, variabilele şi numerele. Obiectele structurate, numite structuri, sunt folosite pentru reprezentarea obiectelor cu mai multe componente.
11. Structurile sunt construite prin intermediul functorilor. Fiecare functor este definit prin nume şi domeniu.
12. Tipul obiectului este recunoscut în întregime prin propria forma sintactică.
13. Domeniul lexical al variabilelor este o singura clauză. De aceea, acelaşi nume de variabilă, în două clauze diferite, înseamnă două variabile diferite.
14. Structurile pot fi reprezentate ca arbori. PROLOG-ul este limbajul care deţine o mare abilitate în procesarea structurilor arborescente.
15. Operaţia de joc ia doi termeni şi încearcă să-i identifice prin instanţierea variabilelor din ambii termeni.
16. Semantica procedurală a PROLOG-ului constă în, satisfacerea unei liste de **goal**-uri, în contextul unui program dat. Procedura dă la ieşire valoarea de adevărat sau fals a **goal**-urilor din listă şi variabilele de instanţiere corespunzatoare. Procedura realizează automat **backtracks** pentru examinarea alternativelor
17. Înţelesul declarativ al programelor nu depinde de ordinea clauzelor şi nici de ordinea **goal**-urilor în clauze.
18. Înţelesul procedural depinde de ordinea **goal**-urilor şi clauzelor. Ordinea afectează eficienţa unui program, iar uneori o ordine neadecvată poate conduce, chiar, la un proces recursiv infinit.

19. Tehnica generală de elaborare a unui program presupune să se dea programului, întâi un conținut declarativ corect, iar apoi, prin schimbarea ordini clauzelor și **goal**-urilor, să se obțină eficiența și robustețea necesare.

5.2.5. Prelucrarea *Listelor.*

Recursivitatea este o proprietate principală a listelor. Ea este folosită în două situații:

- relațiile sunt descrise cu ajutorul listelor;
- obiectele compuse sunt părți ale altor obiecte compuse.

Listele sunt *structuri de date fundamentale* în programele scrise în PROLOG. Limbajul oferă un mod facil de reprezentare a lor, ca obiecte compuse. Elementele unei liste sunt separate cu virgule și sunt plasate între paranteze drepte "[" și "]". In TURBO PROLOG toate elementele dintr-o listă trebuie să aparțină aceluiași domeniu. PDC admite liste cu elemente de tipuri diferite. De exemplu, o listă care conține întegi și caractere se declară astfel:

```
domains
        element = b(integer) ; k(char)
        list = element*
        [b(52), k(' i '), k(' j '), k(' u '), b(82), b(1)]
```

Oricare element dintr-o listă poate fi inclus în alte liste. TURBO PROLOG procesează o listă prin divizarea ei în două părți, un cap și o coadă. Separarea celor două părți se face cu bara verticală "|". Fiind un obiect compus, lista poate fi reprezentată printr-o arborescență. Ex.: [a, b, c, d, e].

Procesarea listei constă în prelucrarea elementelor, până la epuizarea ei. Lista este eminamente un obiect recursiv [27]. **Procedura generală de prelucrare a listei** este următoarea:

```
procedură ( [ ] ) :- !.
procedură ( H | Tail ) :- fă_ceva ( H ), procedură ( Tail ).
```

Căutarea în listă.

Lista fiind o structură, inspectarea ei se face într-o anumită manieră. Pentru a afla, dacă un nume se găsește într-o listă, trebuie ca programul să exprime o relație între două obiecte, un nume și o listă de nume, ceea ce corespunde predicatului **member** (name,namelist). Prima clauză investigează capul listei. Dacă, capul este egal cu "name", numele căutat, atunci se trage concluzia că "name" este un membru al listei. In această fază nu interesează coada listei, fapt indicat prin variabila anonimă " _ ". Dacă, capul listei este diferit de "name", atunci trebuie investigat unde poate fi găsit "name", în coada listei. Cu alte cuvinte "name" este un membru al listei, dacă "name" este un membru al cozii listei. Exprimarea în limbajul PROLOG a regulii enunțate se regăsește în exemplul de mai jos:

```
domains
        namelist = name*          ---> Definirea listei și tipului elementelor
        name = symbol
predicates
        member (name,namelist)    ---> Predicatul member nu este un predicat standard, numele lui este păstrat de convenienţă
clauses
        member (Name,[Name| _ ]).
```

```
        member (Name,[ _ | Tail]) :- member (Name,Tail).
goal
        member (şurub,[piuliță, şaibă, şurub, ştift]).
```

La acest **goal** programul va răspunde "yes". Programul de mai sus poate să funcționeze în două moduri diferite. Să luăm în considerare din nou cele două clauze:

```
        member (Name, [Name| _ ]).
        member (Name, [ _ | Tail]) :- member (Name,Tail).
```

Aceste două clauze pot fi privite din două puncte de vedere. Din punct de vedere declarativ, spunem că pentru o listă dată, Name este un membru al listei, dacă capul acesteia este Name; dacă nu, atunci Name este un membru al listei, dacă este un membru al cozii. Din punct de vedere procedural, cele două clauze pot fi interpretate astfel: să se caute un membru al listei, cercetînd capul, iar în caz contrar, cercetând coada. Aceste două puncte de vedere corespund următoarelor **goal**-uri:

```
        member (şurub,[piuliță,şaibă,şurub,ştift]).
```

respectiv

```
        member (X,[piuliță,şaibă,şurub,ştift]).
```

In timp ce efectul primului **goal** este de a descoperi valoarea de adevăr, al doilea cere să se găsească toți membri listei. De multe ori se întâmplă, să se construiască predicatele dintr-un punct de vedere, iar ele să funcționeze pentru celălalt.

Pentru a obține lista elementelor trebuie să facem o modificare şi anume, să indicăm în program ca scrierea să înceteze, atunci când lista este goală. Programul este următorul:

```
domains
        namelist = name*
        name = symbol
predicates
        member (namelist)
clauses
        member ( [ ] ):- !.
        member ( [Head |Tail] ) :-
        write (Head), nl, member (Tail).
goal
        member ( [piuliță,şaibă,şurub,ştift] )
```

Prima expresie a clauzei testează vidarea listei pentru încetarea acțiunii recursive.

A doua expresie procesează recursiv lista. Se observă că nu este necesar să se cunoască lungimea acesteia.

Programul va afişa conținutul listei "namelist".

Concatenarea listelor

Conform cu aceeaşi convenție aplicată predicatului **member**, definim un predicat de concatenare **append** (List1, List2, List3). Operația constă din combinarea *List1* cu *List2*, pentru a forma *List3*. Si de această dată se foloseşte recursivitatea.

Raționamentul este următorul. Dacă *List1* este goală, rezultatul concatenării *List1*, cu *List2*, este *List2*. In TURBO PROLOG se scrie:

```
append ( [ ], List2, List2] ).
```

In caz contrar, combinăm listele *List1* şi *List2* pentru a forma *List3*, punând capul listei *List1*, să fie capul listei *List3*. Restul listei *List3* se obţine punând împreună restul listei *List1*, cu întreagă listă *List2*. In TURBO PROLOG:

```
append ( [X|L1]), List2, [X|L3] ) :-
append (L1, List2, L3).
```

Predicatul **append** operează în felul următor: Regula recursivă transferă din *List1*, în *List3*, element cu element, până când *List1* este goală. Apoi se agaţă *List2*, la spatele lui *List3*.

```
domains
        integerlist = integer*
predicates
        append (integerlist,integerlist,integerlist)
        writelist (integerlist)
clauses
        append ( [ ],List,List).
        append ( [X|L1], List2, [X|L3] ) :-
        append ( L1,List2,L3 ).
        writelist ( [ ] ).
        writelist ( [Head|Tail] ) :-
        write (Head," "),
        writelist ( Tail ).
```

Prima expresie a clauzei testează vidarea listei *List1*

A doua expresie a clauzei realizează recursivitatea

Clauza de afişare a conţinutului listei

Rulând programul cu următorul goal:

```
goal
        append ( [9,8,7],[1,2,3,4],L) , writelist (L).
```

Rularea programului cu un scop strict de adăugare

vom obţine rezultatul: 9 8 7 1 2 3 4 L=[9,8,7,1,2,3,4] .

Se pot rula mai multe **goal**-uri, ca exemplu:

```
goal
        append (L1,L2,[1,2,3]).
```

Rularea programului cu un scop de reconstituire a tuturor variantelor posibile de liste, care prin adăugare să dea lista precizată în enunţ

Se obţine:

```
L1= [ ],        L2=[1,2,3]
L1= [1],        L2=[2,3]
L1= [1,2],      L2=[3]
L1= [1,2,3]     L2=[ ].
```

```
goal
        append ( L1,[1,2,3],[5,6,7,1,2,3] ).
```

Rularea programului pentru reconstituirea *uneia* dintre listele termen

Se obţine: L1=[5,6,7].

5.2.6. Forţarea şi prevenirea procedurii *backtracking*.

Procedura **backtracking** este principalul instrument de căutare, al limbajului PROLOG. In prelucrarea recursivă a listelor, procedura este automată. Sunt unele situaţii, când este necesară forţarea procedurii, iar altele când se impune prevenirea ei. In cele două cazuri, se folosesc două predicate standard destinate forţării, respectiv prevenirii procedurii **backtracking**. Secvenţa de program de mai jos foloseşte predicatul **fail**, pentru forţarea trecerii la expresia a doua a clauzei captează_date. Acest predicat înseamnă: introducerea unui eşec artificial.

```
captează_date:-shiftwindow(1),defineşte_clasa,clearwindow,fail.
captează_date:-shiftwindow(1),
write("\nContinuaţi ? d/n "),readchar(R),
upper_lower(R,Rp),Rp='d',!,captează_date.
```

Folosirea predicatului "!" scurtează căutarea, care încetează, atunci când, răspunsul utilizatorului este diferit de "d". Predicatul standard **upper_lower( )**, aduce toate răspunsurile la literă mică, pentru a preveni erorile cauzate de starea *caps lock on*, neobservată de către utilizator.

Procedeul tăieturii se foloseste şi pentru a preveni procedura **backtracking**, spre clauza următoare. Un exemplu elocvent este acela de înlăturare a nedeterminismului, din clauzele predicatului **member**. Transformarea clauzelor nedeterministe, în clauze deterministe, se face prin folosirea procedeului tăieturii. Versiunea deterministă a lui **member** este următoarea:

```
verify_member (X,[X|_]) :-!.
verify_member (X,[_|Y]) :- verify_member(X,Y).
```

Verify_member poate fi utilizat pentru a căuta, dacă un element este membru al unei liste date, dar nu poate fi folosit, pentru a lega o variabilă, de membri unei liste.

5.2.7. Predicate standard pentru intrări-ieşiri

Predicatele folosite pentru introducerea datelor, deci pentru citire, sunt următoarele:

- **readln** (line) Domeniul pentru variabila *line* trebuie să fie de tip şir sau simbol. Variabila *line* trebuie să fie liberă, înaintea invocării ei de către readln. Readln citeşte până la 150 caractere pentru tipul symbol, până la 64k pentru tipul şir, şi până la limită, pentru alte dispozitive.
- **readint** (X) Domeniul pentru variabila *X* trebuie să fie de tip integer, iar variabila să fie liberă, înainte de apelare.
- **readreal**(X) Domeniul pentru variabila *X* trebuie să fie de tip real, iar variabila să fie liberă, înainte de apelare.
- **file_str**(Filename,X) Domeniul pentru *filename* şi *X* trebuie să fie de tipul simbol sau şir (preferabil de tip şir, pentru că textele lungi încetinesc derularea tabelului cu simboluri). File_str citeşte caractere din fişier, până la întâlnirea caracterului end-of-file (Ctrl+Z). Contextul fişierului *Filename* este transferat variabilei *X*.
- **readchar**(CharParam)*CharParam* trebuie să aparţină domeniului de tip **char** şi să fie liberă,

înainte de invocare. Predicatul citeşte un singur caracter de la dispozitivul de intrare.

-**readterm**(domeniu,termen) Predicatul poate citi un termen din orice domeniu. Este folosit pentru accesarea faptelor plasate în fişierele de pe disc.

Predicatul standard de scriere este **write**. Acesta are următoarea formă generală: write(Arg1, Arg2, Arg3, ...). Argumentele pot fi constante sau variabile, cu precizarea esențială că, variabilele trebuie să fie legate înainte de manipulare.

5.2.8. Sistemul de fişiere TURBO PROLOG

Predicatele standard, pentru citire şi scriere, sunt elegante şi eficiente întrucât, cu o singură comandă, ieşirea poate fi dirijată spre un fişier, în loc să fie afişată pe ecran. TURBO PROLOG face disponibile (utilizabile), un dispozitiv curent de citire, de la care este citită intrarea şi un dispozitiv curent de scriere, căruia îi este trimisă ieşirea. Dispozitivul curent de citire este în mod normal claviatura, iar dispozitivul curent de scriere este ecranul, dar mai pot fi precizate şi alte dispozitive. De exemplu, intrarea ar putea fi citită dintr-un fişier, care este înregistrat pe disc, iar ieşirea ar putea fi trimisă la o imprimantă. Chiar mai mult, este posibil să se reatribuie dispozitivele curente de intrare şi ieşire, în timp ce programul rulează. Fără a pierde din vedere la ce sunt folosite dispozitivele de citire şi scriere, citirea şi scrierea în interiorul unui program TURBO PROLOG sunt manevrate în mod identic. Pentru a putea fi accesat, un fişier trebuie întâi deschis. Un fişier poate fi deschis în patru moduri:

- pentru citire
- pentru scriere
- pentru adăugare
- pentru modificare.

Un fişier deschis, pentru orice altă activitate decât cititul, trebuie închis atunci când activitatea s-a încheiat, ori se abandonează modificările dedicate fişierului. Pot fi deschise mai multe fişiere simultan, astfel încât şi intrarea şi ieşirea să poată fi repede dirijate între fişierele deschise. Din contra, durează mult mai mult să se deschidă şi să se închidă un fişier, decât să se redirecţioneze datele între fişiere.

Când este deschis un fişier, TURBO PROLOG conectează un nume simbolic la numele actual al fişierului, folosit de către DOS. Acest nume simbolic este folosit de către TURBO PROLOG când, intrarea şi ieşirea sunt redirecţionate. Numele simbolic de fişier trebuie să înceapă cu o literă mică şi trebuie declarat ca fişier. Ex.: file = fila1; fila2; sursa.

In fiecare program este permis numai un singur domeniu fişier şi patru simboluri referitoare la dispozitivele de intrare ieşire. Printer, keyboard, screen, com1 sunt definite automat la începutul domeniului fişier şi nu trebuie să apară explicit în declaraţia fişier; printer se referă la portul paralel de imprimantă, iar com1 se referă la portul serial de comunicări. Predicatele standard sunt următoarele:

- **openread (numesimbolic, numedefişierDOS)**

Fişierul "numedefişierDOS" este deschis pentru citire. După deschidere fişierul este apelat cu numele simbolic. Dacă fişierul nu este găsit, predicatul eşuează. Dacă numele DOS este ilegal, se afişează un mesaj de eroare.

- openwrite (numesimbolic, numedefişierDOS)

Fişierul cu numele DOS este deschis pentru scriere. Dacă fişierul deja există, acesta este şters.

- openappend (numesimbolic, numedefişierDOS)

Fişierul este deschis pentru adăugare. Dacă nu este găsit, se afişază un mesaj de eroare şi se opreşte execţia programului.

- openmodify (numesimbolic, numedefişierDOS)

Fişierul este deschis, atât pentru citire, cât şi pentru scriere. Predicatul **openmodify** poate fi folosit corelat cu predicatul standard **filepos**, pentru a actualiza accesul aleator la fişier.

- filemode (numesimbolic, filemod)

Fişierul filemode este deschis cu specificaţia *filemode 0*, pentru fişier text, *filemod 1*, pentru fişier binar. Filemode poate fi folosit pentru a accesa fişiere binare.

- closefile (numesimbolic)

Predicatul închide fişierul şi are succes, chiar dacă fişierul cu pricina, nu a fost deschis.

- readdevice (numesimbolic)

Predicatul redesemnează dispozitivul curent de citire, oferind *numesimbolic*, care este conectat la fişierul ce a fost deschis pentru citire. Dacă *numesimbolic* este liber, cererea va fi conectată la dispozitivul de citire.

- writedevice (numesimbolic)

Predicatul redesemnează dispozitivul curent de scriere, oferind pentru aceasta fişierul indicat, care a fost deschis pentru scriere, sau adăugare. De exemplu, în programul următor se deschide un fişier numit *mydata.fil*, pentru scriere şi se direcţionează toate ieşirile produse de clauzele dintre cele două **writedevice**. Codul următor, cu excepţia fişierului, este asociat cu numele simbolic de fişier *destination*, care apare în declaraţia din *domains*.

```
domains
        file = destination
goal
        openwrite (destination, mydate.fil"),
        writedevice (destination),
                |
                |
        writedevice (screen).
```

La fel ca şi exemplul de mai sus, cel care urmează, redirecţionează ieşirea produsă de clauzele plasate între cele două **writedevice**, spre dispozitivul imprimantă. Imprimanta este un dispozitiv care nu necesită deschidere explicită, aceasta fiind făcută de sistem. Precauţia ce trebuie luată, este aceea că, dacă nu este conectată nici o imprimantă, sistemul se agaţă, caz în care se foloseşte combinaţia de taste *Ctrl + Break* pentru a permite reluarea controlului de catre TURBO PROLOG.

```
        writedevice (printer),
        writedevice (screen),
```

Dispozitivul com1 este deschis automat. In cazul când, computerul nu are ieşire serială, dacă se foloseşte *com1*, în timpul rulării programului se va semnala eroare. Fişierul este închis la apăsarea tastei #. In timpul introducerii textului, fiecare cod ASCII primeşte de la tastatură câte un *retur car* şi trimite în fişier două coduri ASCII, unul de *retur car* şi altul de schimbare linie (*feed line*). Comanda DOS *TYPE*, cere să fie prezente ambele caractere, în această ordine, pentru a genera o lista proprie a conţinutului fişierului.

```
domains
        file = fişier
predicates
        start
        readin(char)
goal
        start
clauses
        start:-
        write("Acest program citeşte valorile de intrare \nde la testatură şi \n
        le scrie în EXEMPLUL. UNU.\n"),
        write("Apăsaţi # pentru introducere :\n"), openwrite(fişier,"exemplu.unu"),
        readchar(X), readin(X),
        closefile(fişier),
        writedevice(screen),
        write("Intrarea a fost transferată în fişier").
        readin( '#' ):-!.
        readin('\13'):-!,writedevice(fişier),
        write("\13\10"),writedevice(screen),
        write("\13\10"),readchar(X),readin(X).
        readin( X ):- writedevice(fişier),
        write(X),writedevice(screen),
        write(X),readchar(Y),readin(Y).
```

- filepos (numesimbolic, poziţieînfişier, mod)

Acest predicat poate schimba poziţia de citire şi scriere într-un fişier identificat prin numesimbolic, care a fost deschis cu ajutorul predicatului **openmodify**, sau poate returna poziţia curentă în fişier dacă este apelat cu argumentul *poziţieînfisier*, fără nici o valoare (ca variabilă) şi mod *0*. Argumentul *mod* este un întreg şi specifică în ce mod să fie interpretată valoarea din *poziţieînfişier*.

```
domains
        file = intrare
predicates
```

```
        start
        inspectare_poziție
goal
        start.
clauses
  start:-
        write("Cu ce fișier doriți să lucrați ? "),
        readln(Nume_fișier),
        openread(intrare,Nume_fișier),
        inspectare_poziție.
  inspectare_poziție :-
        readdevice(keyboard),nl,write("Număr de poziție :"),
        readreal(X),
        readdevice(intrare),filepos(intrare,X,0),readchar(Y),
        write(Y),inspectare_poziție.
```

- eof (numesimbolic)

Predicatul cercetează când poziția, în timpul unei operații de citire, a ajuns la sfârșitul fișierului. La sfârșitul fișierului predicatul are succes, iar în interior eșec.

```
domains
        file = intrare
predicates
        start
        afișează_conținut
goal
        start.
clauses
  start:-
        clearwindow,
        write("Cu ce fișier doriți să lucrați ? "),
        readln(Nume_fișier),
        openread(intrare,Nume_fișier),
        readdevice(intrare),
        afișează_conținut.
  afișează_conținut :-!,
        not(eof(intrare)),readchar(Y),char_int(Y,T),
```

```
        write(T," "), afişează_conţinut.
afişează_conţinut :-!,
        nl,readdevice(keyboard),
        write("\nVă rog să apăsaţi spacebar"),readchar(_).
```

- flush (numesimbolic)

Predicatul forţează scrierea conţinutului *buffer*-ului intern în fişierul numit cu *numesimbolic*. **Flush** este util atunci când, ieşirea este direcţionată spre un port serial şi devine necesar, să se trimită datele spre port, înainte ca registrul tampon (*buffer*-ul) să fie plin. In mod normal, trasmiterea se face automat, când se umple registrul.

- exitfile (numedefisierDOS)

Predicatul eşuează dacă numele nu apare în director, sau numele este un nume de fişier invalid, inclusiv *.*. Predicatul este folosit la verificarea existenţei unui fişier, înaintea încercării de deschidere a lui.

```
open (fişier, nume) :-
        exitfile (nume), !, openread (fişier, nume).
open (_, nume) :-
        write ("Eroare, fişierul "nume" nu este de găsit").
```

- deletefile (numedefişierDOS)

Predicatul şterge fişierul cu nume DOS. Acţiunea se întâmplă dacă *numedefişierDOS* este un nume valid de fişier. In caz contrar apare eroare.

- renamefile (numedefişierDOSvechi, numedefişierDOSnou)

Predicatul redenumeşte un fişier DOS schimbând numele vechi, cu unul nou. Dacă numele vechi nu există, sau cele două nume sunt invalide, predicatul eşuează.

5.2.9. Procesarea *Sirurilor*

Următoarele caractere speciale sunt înţelese de TURBO PROLOG, atunci când foloseşte şiruri, astfel:

```
" \n "   linie nouă
" \t "   tabelare
" \b "   spaţiu înapoi
" \\ "   \
```

Backslash este un caracter de control. Tiparirea lui necesită 2 *backslash*-uri. Ex.: A:\\myfile.pro. Predicatele descrise mai jos sunt folosite, sau pentru divizarea unui şir într-o listă de caractere individuale, sau într-o listă de simboluri de legătură (corespondenţă)

- frontchar (sirul1, parametrucaracter, şirul2)

Dacă şirul1 este vid predicatul va eşua. Sirul1 = (concatenarea lui parametrucaracter cu şirul2)

In progamul de mai jos predicatul **frontchar** este folosit pentru a transforma un şir, într-o listă de caractere.

```
domains
        listă_de_caractere=char*
predicates
        conversie_şir (şir_de_caractere, listă_de_caractere)
clauses
        conversie_şir ("",[ ]).
        conversie_şir (S,[H|T]):-
        frontchar(S,H,S1),
        conversie_şir (S1,T).
```

- fronttoken (sirul1, parametrusimbol, rest)

Dacă **fronttoken** este apelat cu şirul1 legat (un şir anume), predicatul descoperă primul simbol al şirului1, care este apoi returnat în *parametrusimbol*. Ce rămâne din şir este returnat în al treilea parametru *rest*. Blancurile care preced şirul sunt ignorate.

O secvenţă de caractere este grupată ca un simbol, când se constituie astfel:

- ca un nume acordat cu sintaxa TURBO PROLOG
- ca un număr (semnul care îl precede este returnat ca un simbol)
- un caracter.

Predicatele următoare pot fi folosite pentru a determina natura unui simbol returnat **isname, str_int** şi **str_ln**, demonstrat în următorul program, care ilustrează divizarea unei propoziţii într-o listă de nume.

```
domains
        numelistă = nume*
        nume  = symbol
predicates
        conversie_propoziţie (string,numelistă)
clauses
        conversie_propoziţie (S,[H|T]):-
        fronttoken(S,H,S1),!,string_numelistă(S1,T).
        conversie_propoziţie (_,[ ]).
```

Un alt exemplu va defini predicatul **scanner**, care va transforma un şir într-o listă de simboluri, de data aceasta, clasificat prin asocierea fiecărui simbol cu un functor. Ex:

```
domains
        tok = număr(integer);caracter(char);nume(string)
        tokl = tok*
predicates
        scanner(string,tokl)
```

```
        maketok(string,tok)
clauses
        scanner("",[ ]) :!.
        scanner(Str,[Tok|Rest]):-
        fronttoken(Str,Sym,Str1),
        maketok(Sym,Tok),
        scanner(Str1,Rest).
        maketok(S,nume(S)):- isname(S),!.
        maketok(S,număr(N)):- str_int(S,N),!.
        maketok(S,caracter(C)):- str_Char(S,C),!.
```

- concat (sirul1, sirul2, sirul3)

Sirul3 se obține prin concatenarea şirului1, cu sirul2. Doi dintre parametri trebuie să fie legați (constante), căci predicatul este determinist şi oferă o singura soluție. Ex: concat ("croco", "dile", Animal) leagă Animal de crocodile.

- frontstr (numardecaractere, şirul1, startstr, şirul2)

Predicatul desparte *şirul1* în două părți. Prima parte de lungime *numardecaractere* va alcătui *startstr*, iar restul va fi *şirul2*. Primii doi parametri trebuie să fie legați, iar ultimii doi trebuie să fie liberi (variabile).

- str_len (stringparam, lungime)

Predicatul returnează lungimea şirului *stringparam*, sau testează dacă, *stringparam*, are lungimea dată.

- isname (sir)

Testează şirul, pentru a verifica, dacă există un nume corelat cu sintaxa TURBO PROLOG, dacă acesta începe cu o literă a alfabetului, urmată de orice număr de litere, digiți şi caractere de subliniere. Spațiile de precedare şi succedare sunt ignorate.

5.2.10. Predicate standard de conversie

Predicatele standard efectuează conversia între caracterele ASCII şi valorile acestora, între un şir şi un caracter, un şir şi un întreg, un şir şi un real, o literă mare şi o literă mică. Predicatele sunt:

- char_int (uncaracter, unintreg)

- str_char (uncaracterinsir, uncaracter)

- str_int (unsir, unintreg)

- str_real (unsir, unreal)

- upper_lower (caracteremari, caracteremici)

Conversia între domeniile *symbol* şi *string*, respectiv *integer* şi *real*, sunt realizate în mod automat, atunci când, se folosesc predicatele standard, cât şi în timpul evaluării expresiilor aritmetice. Această conversie implicită este realizată în mod automat, când un predicat este apelat, la fel ca în exemplul următor:

```
predicates
        p(integer)
clauses
        p(X):-write("Valoarea întreagă este ",X),nl.
```

cele două goal-uri

```
X=1.345,p(X).
X=1,p(X).
```

au acelaşi efect. In exemplul următor conversia este dată explicit:

```
predicates
        int_real(integer,real)
        str_symbol(string,symbol)
clauses
        int_real(X,Y):- X=Y.
        str_symbol(X,Y):- X=Y.
```

Conversia este efectuată de către predicatul standard =.

- findall (numedevariabila, predicat(...), paramlista)

Predicatul este folosit pentru a colecta valorile obținute prin **backtracking**, într-o listă. **Findall** este apelat cu trei parametri: primul parametru specifică, care variabilă din predicat desemnează valorile ce urmează a fi colectate în listă; al doilea este un predicat, care dă valori multiple prin **backtracking**; iar al treilea paramatru este o variabilă, care reține lista de valori oferite de **backtracking**. Trebuie să existe definit un domeniu utilizator, de care să aparțină valorile din *paramlista*.

```
domains
        nume,tip,stare,componente_pr=symbol
        comp_p=symbol*
database - componente
        comp(string)
predicates
        compune(comp_p)
        citeşte
clauses
        compune(Lista):-retractall(comp(_)),NOT(citeşte),findall(C,comp(C),Lista),
        retractall(comp(_)).
        citeşte:-write("\ncomponenta: "),readln(C),C<>"".!,
        assertz(comp(C)),citeşte.
```

Creează lista *Listă* pornind de la conținutul bazei de date dinamice *comp()*, care a fost încărcată cu ajutorul predicatului **citeşte**

- random (numarreal)

Predicatul returnează un număr real *X* care satisface condiția 0<=X<1. In exemplul de mai jos, cu ajutorul lui *random()*, se generează o *semnătură* unicat.

```
predicates
        întrebarea1(string,string)
clauses
        întrebarea1(Pren,Nume):-write("Va rog să vă prezentați\nPrenumele: "),readln(Pren),
                write("  Numele: "),readln(Nume),date(An,Luna,Zi),
                time(Ora,Min,Sec,Sut),Ra=Sec*Sut+1,randominit(Ra),
                random(Semn),Semnatura=Semn*(Ora+Min+Sec*Sut)/(5000+Luna*Zi),
                retractall(_,db2),assertz(semn(Semnatura)),
                consult("\\pdc\\index1.dat",db1),
```

5.3. MODULE DE PROGRAM ALE SISTEMULUI

5.3.1. Modulul de introducere al coeficienților Fuzzy

Acest modul de program are ca scop dezvoltarea unei baze de date, care să conțină coordonatele funcțiilor de apartenență Fuzzy, necesare în deciziile Sistemului Expert. Fiecare înregistrare conține: numele parametrului, numele atributului și cele patru coordonate ale funcției de apartenență corespunzătoare. Atributele sunt exprimări lingvistice de nuanțare a stării parametrului pe care îl însoțesc. Coordonatele sunt abscisele funcției de apartenență trapezoidală. Programul percepe funcția trapezoidală, ca pe un caz general, în care se includ, prin restricții adecvate, funcțiile triunghiulară și dreptunghiulară.

Fazele principale parcurse de program sunt prezentate simplificat în Fig.5.2. Reprezentarea exactă și completă a tuturor fazelor, nu este posibilă, deoarece codul este scris într-un limbaj neprocedural (PROLOG). Varietatea legăturilor stabilite este dependentă de bogăția bazei de cunoștințe.

Modulul de program, el însuși, scop parțial pentru programul monitor, urmărește atingerea a trei scopuri parțiale de rang inferior. Acestea sunt: introducerea de înregistrări noi, modificarea selectivă a înregistrărilor existente și vizualizarea conținutului bazei de cunoștințe. Elementele, oarecum procedurale sunt: definițiile de argumente și predicate, construirea ferestrelor de dialog și validarea ciclărilor.

Programul testează pre-existența în baza de date a parametrilor, care se dorește să fie introduși și efectuează validări ale coordonatelor funcției de apartenență. In caz de nevalidare, se reia introducerea valorii neconforme.

Codul modulului de program este prezentat și explicat în tabelul T5.3. Programul, deși este scris în limbajul PROLOG, are un pronunțat caracter procedural. Efectul procedural a fost obținut prin folosirea bazelor de date dinamice și a perechii de predicate predefinite, **assert-retract**. In acest mod, este posibil să se transfere date între clauze și să se forțeze, în privința datelor, o anumită procedură. Efectul procedural este solicitat de caracterul de modul de

introducere al datelor. In schema din Fig.5.2 este reprezentat aspectul introductiv, de definire, precum şi cel procedural.

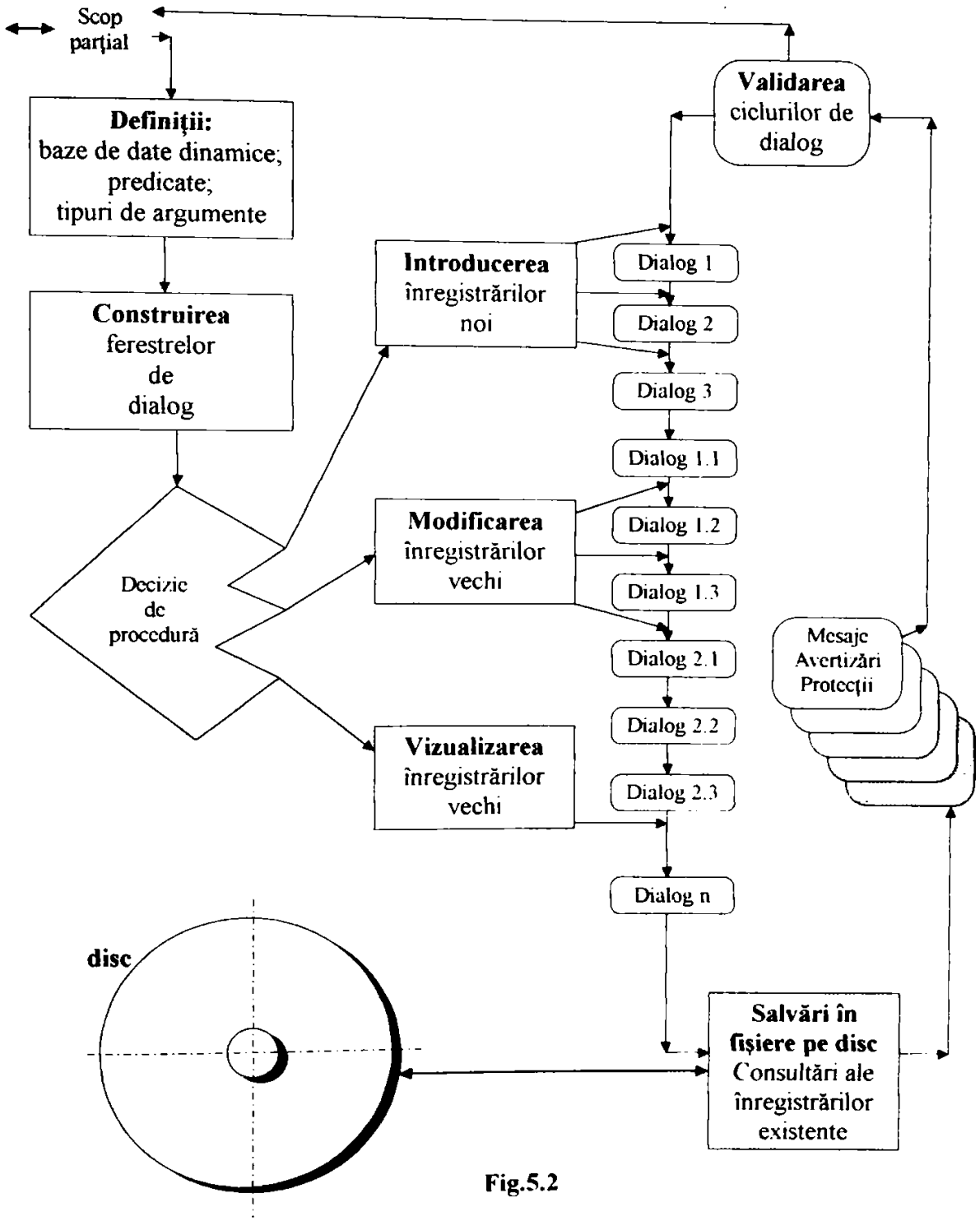


Fig.5.2

Elementele de originalitate ale modulului de program sunt: predicatul de transformare a unei propoziţii, într-un element de tip *symbol*, insensibilizarea *case*, utilizarea unei baze de date, de tip dinamic, pentru efectuarea rocadelor de date, necesare în fazele de corectare a datelor existente, înregistrate în fişier pe disc.

Programul testează intenția utilizatorului de continuare, respectiv de întrerupere a introducerii atributelor şi coordonatelor funcțiilor de apartenență Fuzzy. Programul verifică corectitudinea introducerii coordonatelor, pentru cazul funcțiilor triunghiulare, trapezoidale şi dreptunhiulare. In caz de nerespectare a ordinii pentru aceste trei cazuri, utilizatorul este avertizat şi obligat să reia faza de introducere a datelor.

Bucla dialogului este realizată prin recurența predicatului de captură a datelor. Ieşirea din buclă se petrece la nevalidarea, de către utilizator, a continuării procesului. Părăsirea buclei este însoțită de salvarea pe disc a ultimei variante de bază de date şi afişarea conținutului. Operația de modificare afişează succesiv toate înregistrările existente, pentru a putea fi şterse sau păstrate.

Program sursă	Comentariu
domains **nume_parametru, atribut_parametru=symbol** **t1,t2,t3,t4=real** **atribut=atribut(nume_parametru,atribut_parame** **tru,t1,t2,t3,t4)**	S-a definit un obiect compus numit atribut, care conține două elemente de tip "symbol", pentru stocarea numelui şi a atributului şi patru elemente de tip "real", pentru stocarea coordonatelor Fuzzy.
database - stoc_parametri **stoc1(atribut)**	S-a definit o bază de date de tip dinamic, principală, necesară manipulării obiectului atribut, între memorie şi fişierul înregistrat pe disc (stoc_parametri).
database - stoc_provizoriu **stoc2(atribut)** **database - lucru** **stoc3(string)** **database - lucru2** **stoc4(string,string)** **database - cercetare** **stoc5(atribut)** **database - stoc_ultim stoc6(atribut)**	S-au definit trei baze de date de tip dinamic, pentru operațiile de rocadă, necesare verificării în timpul introducerii datelor, a existenței unei înregistrări cu același parametru şi același atribut, precum şi a rocadei necesare în cazul corectării datelor din fişierul existent (stoc_provizoriu, cercetare, stoc_ultim). S-au definit două baze de date tip dinamic, pentru transferul datelor între clauzele de verificare (lucru, lucru2).
predicates **actualiz_param** **ferestre** **umple** **umple2** **umple3** **umple4** **decizie1** **decizie2** **incarc** **conversie** **modific** **modific_date** **captura** **captura_date** **afiseaza** **repeat** **verificare** **verif1**	Definirea predicatelor: - predicatul scop al modulului de program; - predicatul care defineşte ferestrele; - predicat de manevră al datelor; - predicat de manevră al datelor; - predicat de manevră al datelor; - predicat de manevră al datelor; - două predicate de decizie; - predicat de încărcare al datelor din fişier; - trei predicate necesare la modificarea datelor exitente în fişier; - două predicate pentru introducerea datelor; - predicat pentru afişare; - predicat pentru conversia unei expresii de tip şir de caractere, într-un element de tip "symbol"; - două predicate pentru efectuarea verificării existenței unei înregistrări similare.
clauses **actualiz_param :- ferestre,decizie1, readchar (_),** **removewindow(2,1),removewindow(4,1),removewin** **dow(5,1),!.** **actualiz_param :- decizie2,readchar(_),removewin** **dow(2,1),removewindow(4,1),removewindow(5,1),!.** **actualiz_param :- incarc,shiftwindow(3), afiseaza,** **removewindow(2,1),removewindow(4,1),removewin**	Scopul modulului de program este actualizarea parametrilor. Definiția lui cuprinde această clauză. Ea are trei expresii pentru cele trei subscopuri componente: achiziția parametrilor, modificarea bazei de date, respectiv vizualizarea conținutului acesteia. Cele trei expresii determină, pe acest nivel, o căutare pe orizontală. Ele se termină cu închiderea a trei ferestre şi "cut".

dow(5,1),!.	
umple :- retractall(_,stoc_parametri), retractall (_,stoc_provizoriu), consult("\\prol\\ pdc\\myprog\\ param.dat",stoc_parametri),umple2.	Această clauză golește cele două baze de date, iar apoi încarcă baza "stoc_parametri" cu conținutul fișierului "param.dat".
umple2 :- retract(stoc1(atribut(A,B,C,D,E,F)), stoc_parametri),assertz(stoc2(atribut(A,B,C,D,E,F)),stoc_provizoriu),assertz(stoc6 (atribut (A,B,C, D,E,F)),stoc_ultim), !,umple2.	Clauza, încarcă bazele "stoc_provizoriu" și "stoc_ultim", cu conținutul fișierului "param.dat". Manevrele se efectuează cu perechea de predicate predefinite"retract - assertz".
umple3 :- retract(stoc6(atribut(A,B,C,D,E,F)),stoc_ ultim),assertz(stoc2(atribut(A,B,C,D,E,F)),stoc_pro vizoriu),assertz(stoc5(atribut(A,B,C,D,E,F)), cercetare),!,umple3.	Clauza, asigură încărcarea cu date, a bazelor "stoc_povizoriu" (încărcarea nu este redundantă, căci predicatul "umple3" se apelează după golirea bazei de date implicată) și "cercetare".
umple4 :- retract(stoc5(atribut(A,B,C,D,E,F)), cercetare),assertz(stoc6(atribut(A,B,C,D,E,F)),stoc_ ultim),!,umple4.	Clauza "umple4" îndeplinește o funcție asemănătoare cu cea îndeplinită de clauza "umple3'.
decizie1:-shiftwindow(4),write("Introduceti date ? d/n "), readchar(D), upper_lower(D, Dc), Dc='d', NOT(umple),incarc,captura,!.	Clauza "decizie1" selectează prima ramură a programului, cea de completare a bazei de date.
decizie2 :- shiftwindow(4),nl,write("Modificati date le ? d/n "), readchar(M), upper_lower(M,Mc),Mc ='d',incarc, retract all(_,stoc_provizoriu) ,modific,!.	Clauza "decizie2" selectează a doua ramură, cea de modificare a datelor existente în fișier.
incarc :- shiftwindow(5),readln(P),P="pa", consult("\\prol\\pdc\\myprog\\param.dat",stoc_parametri).	Clauza, încarcă parametri pentru completare și solicită parola pentru protecția datelor.
captura :- captura_date,write("Continuati sa intro duceti date ? d/n\n"),readchar(D), upper_lower(D,Dc),Dc='d',!,captura.	Clauza "captură", prima expresie, controlează ciclul de introducere al datelor.
captura :- save("\\prol\\pdc\\myprog\\param.dat", stoc_parametri),shiftwindow(3),afiseaza,!.	Clauza "captură", a doua expresie, salvează varianta completată a setului de parametri.
captura_date :- shiftwindow(2),write("Care este numele parametrului ? "), readln(Nume_p), assertz(stoc3(Nume_p)),NOT(repeat),retract(stoc3(Nume_param)),upper_lower(Nume_param,Parame tru),write("Care este atributul parametrului ",Parametru," ?\n"),readln (Atrib_param),assertz (stoc3(Atrib_param)),NOT(repeat),retract(stoc3(At ribut_parametru)),upper_lower(Atribut_parametr u,Atr),assertz(stoc4(Parametru,Atr)),NOT(verifica re),retractall(_,lucru2),retractall(_,cercetare),NOT(umple3),NOT(umple4),write("\nValoare p1 :"), readreal(P1),write("Valoarep2(p2>p1):"), readreal (P2), P2>P1,write("Valoare p3 (p3>=p2) :"), readreal(P3),P3>=P2,write ("Valoare p4 (p4>p3) :"), readreal(P4),P4>P3,!, assertz(stoc1 (atribut (Parametru, Atr,P1,P2,P3, P4)),stoc_parametri).	"captura_date" este clauza principală de achiziție a datelor. Ea afișează mesaje, apelează clauzele de conversie și de verificare a datelor, afișează datele curente și verifică condiția de validare a coordonatelor funcției de apartenență Fuzzy. Validarea constă în verificarea condiției de succesiune a punctelor, pentru funcția trapezoidală, respectiv triunghiulară.
captura_date :- write("\nAtentie la valori!p1 <p2 <=p3<p4 Continuati?? d/n"),readchar(C), upper_ lower(C,Cc),Cc='d',!,clearwindow,captura_date.	A doua expresie a clauzei, avertizează operatorul asupra nerespectării succesiunii și condiționează continuarea introducerii datelor.
repeat:-retract(stoc3(Atrib)),fronttoken(Atrib ,X1, X2),assertz(stoc3(X1)),!,X2<>"",retract(stoc3(X1)), concat(X1,"_",X11),fronttoken(X2,Cap,Coada),con cat(X11,Cap,X),concat(X,Coada,Z),assertz(stoc3(Z)),fronttoken(Z,_,Z1),!,Z1<>"",repeat.	Clauza "repeat"transformă expresia șir de caractere, introdusă de la tastatură, într-un element de tip "symbol", compatibil cu inferența, în care se va folosi și în primă fază, cu câmpul în care se va înregistra.
verificare :- NOT(verif1),retract(stoc5(atribut (P1,A1,V1,V2,V3,V4)),cercetare),!,P1><"",write("\ nmai aveti ",P1," ",A1," ",V1," ",V2," ",V3," ",V4),verificare.	Clauza, declanșează ciclul de verificare al existenței, în baza de date, a celei ce se intenționează să se introducă.
verif1 :- retract(stoc4(P,A)),retract(stoc2(atribut(P1,A1,V1,V2,V3,V4)),stoc_provizoriu),!,P1><"",ass ertz(stoc4(P,A)),assertz(stoc5(atribut(P1,A1,V1,V2,	"verif1" este clauza propriu-zisă de verificare.

V3,V4)),cercetare),concat(P,A,C1),concat(P1,A1,C2),!,C1><C2,retractall(_,cercetare), verif1.	
afiseaza :- retract(stoc1(atribut(A,Aa,B,C, D,E)), stoc_parametri),write("\nParam:",A,"Atributul: ", Aa,"p1=",B," p2=",C," p3=",D," p4=",E), A="",!, afiseaza. **afiseaza :-!.**	Clauza de afişare a rezultatelor.
ferestre :-makewindow(1,0,0,"",0,0,25,80), make window(2,111,111," Hello ! ",15,0, 10,59), make window(3,48,48," Rezultate ", 0,0,15,80), make window(4,32,32,"Informatii suplimentare ",9, 45, 16,35),makewindow(5,0,3,"PAROLA",12,50,3, 20).	Clauza de definire a ferestrelor de lucru.
modific :- modific_date,!,modific. **modific :- retractall(_,stoc_parametri), conversie, save("\\prol\\pdc\\myprog\\param.dat",stoc_param etri),nl,write("ultima varianta"), shiftwindow(3), nl,nl,afiseaza,!.**	Clauza de modificare a datelor.
conversie :- retract(stoc2(atribut(A,Aa,B, C,D,E)), stoc_provizoriu),assertz(stoc1(atribut(A,Aa,B,C,D, E)),stoc_parametri),A<>"",Aa<>"",!,conversie. **conversie:- !.**	Clauza de rocadă a datelor.
modific_date :- retract(stoc1(atribut(A,Aa,B,C,D,E)),stoc_parametri),shiftwindow(3), nl, write("Para metru :",A,"Atributul :",Aa,"p1=",B,"p2=",C, "p3 =",D,"p4=",E),shiftwindow(2),nl, write ("Este bun ?? d/n"),readchar(Bun), upper_lower (Bun, Bunc), Bunc='d',!,assertz(stoc2(atribut(A,Aa,B,C,D,E)), stoc_provizoriu).	Clauza de selecţie a datelor, care se doreşte să fie păstrate în baza de date. Datele neselectate vor fi eliminate.
goal **actualiz_param.**	Scopul modulului de program.

T5.3

5.3.2. Modulul de definire al entităţilor

Funcţia principală a acestui modul este aceea de a creea o bază de cunoştinţe şi de a creea un mediu de interogare-explicitare cu funcţionare extensivă, care să simuleze curiozitatea nesfârşită. Evident, modulul dă şi posibilitatea curmării dialogului interogativ. Dezvoltarea curiozităţii nu este absolut extensivă, ci ţine cont de cunoştinţele pre-existente, peste care trece, considerându-le ca atare. Dezvoltarea şirului de aşteptare al întrebărilor este mai dinamică decât ritmul răspunsurilor. Astfel că, şi în cazul în care utilizatorul manifestă apatie, în ce priveşte entităţile noi, sistemul are ce să întrebe.

Si acest modul efectuează verificări ale cunoştinţelor pre-existente, dar caracterul programului este mai puţin procedural. Codul programului şi explicaţiile sunt prezentate în tabelul T5.4.

Program sursă	Comentariu
domains **nume,tip,stare,componente_pr=symbol** **comp_p=symbol*** **gen,functie_princ,comp_esenta,domeniu=string**	S-a definit tipul elementelor folosite ca argumente în predicate şi în bazele de date dinamice, din acest modul de program.
database - stoc_clase **entitate(nume,tip,gen,functie_princ,comp_p,comp_e senta,stare,domeniu)** **database - componente** **comp(string)**	S-au definit trei baze de date de tip dinamic, una principală, numită "stoc_clase", pentru stocarea definiţiilor entităţilor introduse şi alte două "comp" şi "rel", pentru transferul datelor între clauze.

database - reciteste1 **re1(string)**	
predicates **capteaza** **ferestre** **capteaza_date** **defineste_clasa** **spune(string)** **alege_tip(string,string)** **cauta_tip(integer,string,symbol)** **compune(comp_p)** **citeste** **cauta_stare(symbol)** **controlc(tip,gen,functie_princ,comp_p,** **comp_esenta,stare,domeniu,string)** **sterge**	Definirea predicatelor: - predicat scop al modulului de program; - predicat care defineşte ferestrele; - predicat de captare al datelor; - predicat de definire al claselor; - predicat de interogare modală; - predicat de selecţie a tipului de entitate; - predicat auxiliar de selecţie a tipului de entitate; - predicat de alcătuire a listei de componente; - predicat care provoacă formarea unui şir de comp; - predicat care verifică starea de singularitate din BD; - predicat care cercetează existenţa noii entităţi în baza de date şi în caz afirmativ, îi afişează definiţia; - predicat pentru vidarea globală a BD dinamice.
clauses **capteaza:-ferestre,sterge,capteaza_date.** **/*==========*/** **ferestre:- makewindow(1,96,96,"Dialog",0,0,13,80),** **makewindow(2,7,7," Avertizari ",13,0,12,50),** **makewindow(3,23,23," Observatii ",13,50,12,30).**	Clauza scop. Clauza de construire a ferestrelor.
capteaza_date:-shiftwindow(1), **defineste_clasa,clearwindow,fail.** **capteaza_date:-shiftwindow(1),** **write("\nContinuati ? d/n "),readchar(R),** **upper_lower(R,Rp),Rp='d',!,capteaza_date**	Clauza de captură a datelor are două secţiuni, una pentru definirea datelor şi una pentru a crea o recursivitate opţională a utilizatorului (cu protecţie "lower case").
defineste_clasa :- **consult("\\prol\\pdc\\myprog\\clase.sto",stoc_clase),** **write("\nDati numele noii entitati :"),** **spune(C),clearwindow,** **assertz(comp(C)),controlc(Ti,Ge,F_prin,** **Com,Comp_esent,Star,Do,Mesaj),** **shiftwindow(2),clearwindow,** **write(Mesaj," ca ",Ti," apartinind ",Ge,** **"\nsi are functia ",F_prin,"\nAlcatuit din** **:", Com, \n Componenta principala:",** **Comp_esent, "\nStarea :", Star , "\nDome** **niu :",Do),** **Mesaj="",!,clearwindow,shiftwindow(1),**	"Defineste_clasa" este clauza principală a modulului de program. Ea efectuează mai multe operaţii după cum urmează: - întreabă numele noii entităţi; - verifică existenţa acesteia în BD memorată pe disc; - afişează un mesaj;
write("\nCe tip de entitate este ",C," ? "), **alege_tip(Tip,Art),** **write("\nCare este genul proxim declarat** **al ",C," ? "),readln(Gen),** **write("\nCare este functia principala a ",** **Art," ",C," : "),** **readln(F_princ),assertz(re1(F_princ)),** **write("\nCare sunt componentele** **principale ale ",C," ? "),** **compune(Comp),** **write("\nCare este componenta esentiala** **fara de care\n",C," nu poate avea functia** **de ",F_princ," ? "),** **readln(Comp_esenta),assertz(re1(Comp_es** **enta)),** **retractall(comp(_)),assertz(comp(Comp_es** **enta)),cauta_stare(Stare),** **write("\nCare este domeniu de care** **apartine ?"),readln(Dom),**	- cere interlocutorului un set de detalii, referitoare la noua entitate, introdusă în baza de cunoştinţe - cercetează conţinutul bazei entitate pentru acordarea atributului de stare a noii entităţi: 1.singulară; 2.numele clasei existente, în care se include; 3.numele noii clase; - delimitează domeniul de existenţă al entităţii;

retractall(entitate(_,_,_,_,_,_,_,_)), consult("\\prol\\pdc\\myprog\\clase.sto", stoc_clase), asserta(entitate(C,Tip,Gen,F_princ,Comp, Comp_esenta,Stare,Dom)), save("\\prol\\pdc\\myprog\\clase.sto",stoc_ clase), retractall(entitate(_,_,_,_,_,_,_,_)).	- completează baza de cunoştinţe: - salvează ultima variantă.
compune(Lista):- NOT(citeste),findall(C,comp(C),Lista), retractall(comp(_)). citeste:-write("\ncomponenta: "),readln(C),C<>"",!, assertz(comp(C)),citeste.	Clauza compune lista componentelor necesară memorării
alege_tip(Tip,Art):- write("\n1. Tip obiect simplu\n2. Tip ansamblu\n3. Tip functie\n4. Tip relatie cauzala\n5. Tip actiune\n"), readint(N),cauta_tip(N,Tip,Art). cauta_tip(1,"obiect_simplu","obiectului"). cauta_tip(2,"obiect_compus","ansamblului "). cauta_tip(3,"relatie_functionala","functiei "). cauta_tip(4,"relatie_cauzala","cauzei"). cauta_tip(5,"actiune","actului\n").	Clauza. alege dintr-un set tipul entităţii şi îl transferă clauzei de definire.
cauta_stare(Stare):- consult("\\prol\\pdc\\myprog\\clase.sto",stoc_clase), retract(comp(Esenta)),retract(entitate(_,_, _,_,_,Comp,_,_)), Esenta=Comp,!,Stare="prezenta". cauta_stare(Stare):- Stare="Individualitate", retractall(entitate(_,_,_,_,_,_,_,_)).	Clauza. verifică existenta în BD a componentei principale şi dă atributul de singularitate. după caz.
controlc(Ti,Ge,F_prin,Com,Comp_esent,Star,Do, Mesaj):- retract(comp(C)), retract(entitate(Cc,Ti,Ge,F_prin,Com,Com p_esent,Star,Do)), C=Cc,!,Mesaj="Entitatea exista deja...", retractall(entitate(_,_,_,_,_,_,_,_)). controlc(A,B,C,D,E,F,G,Mesaj):- retractall(entitate(_,_,_,_,_,_,_,_)), A="",B="",C="",D=[],E="",F="",G="", Mesaj="",!.	Clauza. verifică existenţa în BD a noii entităţi şi afişează un mesaj. Clauza are două expresii.
spune(C):-shiftwindow(3),write("\nIntroduceti un concept nou ? d/n"), readchar(V),upper_lower(V,Va),Va<>'d',!, write("\nExplicitari la conceptele anterioare"), retract(re1(Dat)),Dat<>"",!,C=Dat. spune(C):-clearwindow,shiftwindow(1), retractall(re1(_)), readln(C).	Clauza. citeşte datele de intrare şi verifică opţiunea de menţinere a recursivităţii. Clauza are două expresii.
sterge:-retractall(entitate(_,_,_,_,_,_,_,_)), retractall(comp(_)),retractall(re1(_)).	Clauza pentru golirea "centralizată" a bazelor de date dinamice.
goal capteaza.	Scopul modulului de program.

T5.4

5.3.3 Modulul de dialog preliminar

Modulul de dialog, prezentat în tabelul T5.5, implementează prima etapă de interogare a utilizatorului. Intrebările conținute în modulul de program sunt cele prezentate în paragraful 4.2.3. şi urmează ordinea din schema reprezentată de Fig.4.6.

Program sursă	Comentariu
include "\\prol\\pdctools\\tdoms.pro" include "\\prol\\pdctools\\tpreds.pro" include "\\prol\\pdctools\\mouse\\mouse.pre" include "\\prol\\pdctools\\mouse\\mouse.pro" include "\\prol\\pdctools\\mouse\\mustmous.pro"	Instrucţiuni compilator necesare pentru identificarea locaţiei programelor, care urmează să fie compilate împreună cu programul curent
domains prenume,nume,mediu=symbol data=data(integer,integer,integer) ora=ora(integer,integer,integer,integer) index_nume=index_nume(prenume,nume, semnatura,data,ora) semnatura,t_min,t_max,umid=real stare=stare(integer,symbol) stari=stare*	Definirea domeniului argumentelor
database - db1 index(index_nume) database - db2 semn(real) database - db3 conditii(t_min,t_max,umid,mediu) database - db4 nr(integer)	Definirea bazelor de date dinamice Bază de date pentru stocarea numelor utilizatorilor Bază de date pentru stocarea codului de identificare Bază de date pentru stocarea parametrilor preliminari Bază de date operativă
predicates rezolvare mediu conversatie conversatie2 intrebarea1(string,string) intrebarea2 intrebarea2a intrebarea2b intrebarea2c(real) intrebarea2d(real) intrebarea2e(real) intrebarea2f(symbol) afiseaza(stari) afla(symbol,char) raspunsuri iesire repetare	Definirea predicatelor Numele predicatelor sugerează funcţiile pe care acestea le îndeplinesc
clauses rezolvare :- mediu,mouse_on,conversatie, iesire,mouse_off,!.	Clauza principală, care conţine predicatul scop al modulului de program
mediu :- makewindow(1,0,0,"",0,0,25,80),shiftwindo w(1),clearwindow, makewindow(2,79,64," dialog ",0,0,12,80), makewindow(3,32,32," observatii ",12,0, 13,50), makewindow(4,12,12," avertizare ",12,50, 13,30).	Clauza care defineşte ferestrele de lucru

conversatie :- conversatie2,shiftwindow(4),write ("\nContinuati?d/n"),readchar(Xi), upper_ lower(Xi,X),X='d',clearwindow, !,conversatie	Clauza care întreţine recursivitatea dialogului cu utilizatorul; mai conţine opţiunea de ieşire şi protecţia "case"
conversatie2 :- shiftwindow(3),clearwindow, intrebarea1(Pren,Nume),clearwindow, write("Bun venit si\n MULT SUCCES ",Pren," ",Nume),intrebarea2.	Clauza care apelează întrebările
intrebarea1(Pren,Nume):-write("Va rog sa va prezentati\nPrenumele: "),readln(Pren), write("Numele:"),readln(Nume),date(An, Luna,Zi),time(Ora,Min,Sec,Sut),Ra= Sec* Sut+1,randominit(Ra),random(Semn), Semnatura=Semn*(Ora+Min+Sec*Sut)/(50 00+Luna*Zi),retractall(_,db2),assertz(semn (Semnatura)),consult("\\prol\\pdc\\ myprog\\index1.dat",db1),assertz(index(in dex_nume(Pren,Nume,Semnatura,data(An, Luna,Zi),ora(Ora,Min,Sec,Sut))),db1), save("\\prol\\pdc\\myprog\\index1.dat",db1),retractall(_,db1).	Prima clauză interogativă. Ea conţine interogaţia şi generarea aleatoare a codului de identificare personală a utilizatorilor
intrebarea2 :- intrebarea2a,clearwindow, NOT (repetare).	Clauza de apelare recursivă a întrebării următoare
intrebarea2b :- write("\nCare este temperatura minima de lucru ?"),intrebarea2c(Tmin),shiftwindow (2),write("\nTemperarura minima :",Tmin ," 0C"), write("\nCare este temperatura maxima de lucru ?"),intrebarea2d(Tmax), shiftwindow(2),write("\n Temperatura maxima :",Tmax," 0C"),write("\nCare este umiditatea relativa :"),intrebarea2e (Umid),shiftwindow(2),write("\n Umiditatea relativa este",Umid," %"), write("\nCare este starea de poluare a mediului :"), intrebarea2f(Praf), assertz(conditii(Tmin,Tmax,Umid,Praf)).	Clauza de interogare a utilizatorului privind parametri preliminari, necesari în prima fază a procesului de inferenţă
intrebarea2c(Tmin):- shiftwindow(3),clearwindow,write("valoar ea in [0]C sau "),readreal(Tmin),!. intrebarea2c(Tmin):-write("nu stiu ?? "),readln(R) ,frontchar(R,R1,_),upper_lower(R1,R2),R2 ='n',Tmin=4.	Clauza de interogare referitoare la valoarea temperaturii minime de funcţionare
intrebarea2d(Tmax):- shiftwindow(3),clearwindow,write("valoar ea in [0]C sau "),readreal(Tmax),!. intrebarea2d(Tmax):-write("nu stiu !! "),readln(R) ,frontchar(R,R1,_),upper_lower(R1,R2),R2 ='n',Tmax=60.	Clauza de interogare referitoare la valoarea temperaturii maxime de funcţionare
intrebarea2e(Umid):- shiftwindow(3),clearwindow,write("umidi tatea in % sau "),readreal(Umid),!. intrebarea2e(Umid):-write("nu stiu !! "),readln(R) ,frontchar(R,R1,_),upper_lower(R1,R2),R2 ='n',Umid=50.	Clauza de interogare referitoare la valoarea umidităţii mediului de funcţionare
intrebarea2f(Praf):-shiftwindow(3),clearwindow, afiseaza([stare(1,neglijabila),stare(2,mica), stare(3,medie),stare(4,mare),stare(5,excesi va)]),write("Care este alegerea ? "),	Clauza de interogare referitoare la nivelul de poluare al mediului de funcţionare

readchar(Q),afla(Praf,Q).	
afiseaza([]):-!. **afiseaza([Titlu|Rest]):-Titlu=stare(A,B),write(A,".",B,"\n"), afiseaza(Rest).**	Clauza care afişează ofertele de situaţie
afla(N,C):- C='1',N="neglijabila",!. **afla(N,C):- C='2',N="mica",!.** **afla(N,C):- C='3',N="medie",!.** **afla(N,C):- C='4',N="mare",!.** **afla(N,C):- C='5',N="excesiva".**	Clauzele de situaţie
repetare :- retract(nr(Nr)),intrebarea2b,Nr1=Nr-1,Nr1>0,!,assertz(nr(Nr1)),repetare.	Clauza de recursivitate la numărul de utilizatori
intrebarea2a:-shiftwindow(4),retract(semn(S),db2), write(S),shiftwindow(2),clearwindow,write("Transmisii mecanice"),write("\nVa cer cateva informatii despre problema dumneavoastra:"),write("\nCati consuma tori de energie mecanica aveti ? "),readint(Nr1),assertz(nr(Nr1)),assertz(semn(S),db2),!.	Clauză de interogare pentru deschiderea dialugului
iesire :- shiftwindow(4),readchar(X),upper_lower(X,Xl),Xl='y',write(Xl),removewindow(2,1),removewindow(3,1).	Clauză de ieşire
goal **rezolvare.**	Scopul modulului de program

T5.5

5.3.4. Modulul pentru alegerea şi dimensionarea transmisiei mecanice

Acest modul efectuează interogarea utilizatorului (pe patru nivele de interogare), conform cu scemele de organizare a cunoştinţelor de transmisii mecanice, prezentate în lucrare. Modulul efectuează alegerea familiei, tipului şi specimenului şi efectuează un set de calcule de dimensionare. Programul sursă al modulului este prezentat în Anexa 2.

5.3.5. Modulul pentru introducerea datelor referitoare la transmisii

Acest modul este un modul auxiliar al modulului precedent, care îndeplineşte funcţia de a ajuta la încărcarea bazelor de date. Bazele de date conţin informaţii necesare funcţionării modulului de la paragraful 5.3.4. Programul sursă al modulului este prezentat în Anexa 3.

6. CONCLUZII FINALE

6.1. INDEX DE CONTRIBUTII PERSONALE

Contribuţiile personale au fost clasate pe câteva grupe de activităţi: proiectare, organizarea proiectării, Inteligenţa Artificială, definiţii, strategia dezvoltării softului, teoria cunoaşterii, dezvoltarea Sistemului Expert şi ordonarea cunoştinţelor. Contribuţiile au fost sortate pe grupe.

Nr. crt.	Ordinea de apariţie	Descrierea contribuţiei	Domeniul de apartenenţă al contribuţiei	Cap.	Pag.
1.	8.	Selectarea celor mai reprezentative enunţuri definitorii ale Sistemului Expert	definiţii	2	20
2.	18.	Definiţia procedurală a Inteligenţei Artificiale (d1)	definiţii	3	40
3.	26.	Definirea parametrului precizie cinematică şi a modului în care acesta se poate constitui în criteriu, pentru selecţia tipului de transmisie mecanică	definiţii	3	70
4.	29.	Definirea funcţiilor de apartenenţă Fuzzy, pentru temperatura mediului de funcţionare al transmisiilor mecanice	definiţii	3	73
5.	30.	Reprezentarea grafică a domeniului funcţiilor de apartenenţă, pentru întreg domeniul de temperaturi de funcţionare al transmisiilor mecanice	definiţii	3	74
6.	31.	Definirea complexităţii constructive a transmisiilor mecanice	definiţii	3	76
7.	32.	Definirea complexităţii roţilor de trasmisie	definiţii	3	76
8.	34.	Definirea funcţiei principale a transmisiei mecanice	definiţii	3	80
9.	35.	Definirea transformării cinematice	definiţii	3	80
10.	37.	Definiţia transmisiei mecanice	definiţii	3	81
11.	38.	Definiţia transmisiei de putere	definiţii	3	81
12.	39.	Definiţia sistemului tehnic	definiţii	3	82
13.	40.	Definiţia sistemului tehnic mobil	definiţii	3	82
14.	41.	Definiţia transmisiei mecanice cu element de transmitere 0	definiţii	3	82
15.	42.	Definiţia transmisiei mecanice cu element de transmitere 1	definiţii	3	82
16.	45.	Enunţul postulatului simplicităţii	definiţii	4	85
17.	46.	Definirea fazelor de punere în practică a principiului simplicităţii	definiţii	4	85
18.	47.	Definirea arborelui motor cel mai adecvat maşinii de lucru	definiţii	4	85
19.	48.	Enunţul postulatului descompunerii rezolvării în etape	definiţii	4	86
20.	49.	Enunţul postulatului inconsistenţei	definiţii	4	87
21.	50.	Enunţul regulilor de evoluţie a inconsistenţei unei baze de cunoştinţe	definiţii	4	88
22.	52.	Definirea scopului global al Sistemului Expert şi a scopurilor parţiale	definiţii	4	89
23.	10.	Descoperirea celor mai importante atribute ale Sistemului Expert, folosit în domeniul proiectării constructive a maşinilor, prin prelucrarea răspunsurilor specialiştilor şi	dezvoltarea sistemului expert	2	27

44.	71.	Programul generic pentru forţarea procedurii back tracking, modificarea caracterelor şi "cut"	expert dezvoltarea sistemului	5	113
45.	72.	Programele pentru utilizarea predicatelor de accesare a discului şi pentru operaţiile de scriere-citire a fişierelor	expert dezvoltarea sistemului	5	117
46.	73.	Programul de construire a unui scaner de sintaxă, necesar pentru dezvoltarea unui analizor gramatical, parser	expert dezvoltarea sistemului	5	120
47.	74.	Programul de generare al unui cod riguros nerepetabil, necesar la înregistrarea utilizatorilor	expert dezvoltarea sistemului	5	122
48.	75.	Algoritmul şi schema logică a modulului de program pentru introducerea coeficienţilor Fuzzy	expert dezvoltarea sistemului	5	123
49.	76.	Modulul de program pentru introducerea coeficienţilor Fuzzy, scris în limbaj PROLOG	expert dezvoltarea sistemului	5	124
50.	77.	Modulul de program pentru definirea entităţilor, scris în limbaj PROLOG	expert dezvoltarea sistemului	5	126
51.	78.	Modulul de program pentru dialogul preliminar, scris în limbaj PROLOG	expert dezvoltarea sistemului	5	129
52.	79.	Modulul de program pentru alegerea şi dimensionarea transmisiei mecanice, scris în limbaj PROLOG;	expert dezvoltarea sistemului	5	131
53.	80.	Modulul de program pentru introducerea datelor referitoare la transmisii, scris în limbaj PROLOG;	expert dezvoltarea sistemului	5	131
54.	7.	Incadrarea Sistemului Expert în domeniul Inteligenţei Artificiale	inteligenţa artificială	2	16
55.	9.	Detalierea claselor de aplicaţii ale Sistemelor Expert	inteligenţa artificială	2	26
56.	24.	Stabilirea claselor de criterii necesare la alegerea tipului de transmisie mecanică	ordonarea cunoştinţelor	3	66
57.	25.	Prezentarea sintetică a principalilor parametri de performanţă funcţională a tipurilor de transmisii mecanice	ordonarea cunoştinţelor	3	68
58.	27.	Prezentarea sintetică a modului de utilizare a preciziei cinematice, ca şi criteriu de selecţie, a tipului de transmisie mecanică	ordonarea cunoştinţelor	3	70
59.	28.	Prezentarea sintetică a parametrilor de performanţă funcţională, a diferitelor tipuri de transmisii mecanice	ordonarea cunoştinţelor	3	71
60.	33.	Prezentarea sintetică a parametrilor de performanţă constructivă şi tehnologică, a diferitelor tipuri de transmisii mecanice	ordonarea cunoştinţelor	3	77
61.	36.	Descoperirea diferenţei specifice, la nivelul funcţiei principale, pentru diferite tipuri de transmisii	ordonarea cunoştinţelor	3	81
62.	43.	Structura ce cuprinde sistemele tehnice, clasele de transmisii şi principalele specimene ale acestora	ordonarea cunoştinţelor	3	82
63.	4.	Algoritmul general al proiectării constructive a maşinilor.	organizarea proiectării	1	8-12
64.	17.	Precizarea tipurilor de soft folosite la asistarea proiectării.	organizarea proiectării	2	36
65.	1.	Explicitarea condiţiilor necesare dezvoltării unui Sistem Expert	proiectare	1	5
66.	2.	Schema conexiunilor activităţilor de proiectare	proiectare	1	6
67.	3.	Stabilirea locului şi a relaţiilor activităţii de proiectare a maşinilor, în cadrul sistemului de producţie	proiectare	1	7
		Clasificarea tipologică a proiectării			

68.	5.	Schema simplificată a procesului de proiectare constructivă a maşinilor	proiectare	1	9
69.	6.	Analiza tipologică a asistării procesului de proiectare	proiectare	1	13
70.	12.	Analiza răspândirii Sistemelor Expert după complexitate, pe mapamond, în Europa, în America şi în Japonia	strategia dezvoltării softului	2	33
71.	13.	Analiza domeniilor preferenţiale pentru dezvoltarea Sistemelor Expert	strategia dezvoltării softului	2	34
72.	14.	Reprezentarea grafică a răspândirii Sistemelor Expert, pe domenii de aplicaţii, pe mapamond şi pe cele trei continente	strategia dezvoltării softului	2	35
73.	15.	Reprezentarea grafică a răspândirii Sistemelor Expert, pe cele patru grupe de complexitate, pe mapamond şi pe cele trei continente	strategia dezvoltării softului	2	35
74.	16.	Reprezentarea grafică a răspândirii Sistemelor Expert, mai complexe decât media, pe cele trei continente	strategia dezvoltării softului	2	35
75.	19.	Precizarea rolului pe care îl are contextul, la stabilirea gradului de utilizare al unei piese de cunoaştere	teoria cunoaşterii	3	49
76.	20.	Structura şi reprezentarea grafică a unei ierarhii de concepte, care stă la baza construirii unei reţele semantice	teoria cunoaşterii	3	52
77.	21.	Reprezentarea grafică a unui model de editor ontologic, util la completarea bazei de date a specimenelor	teoria cunoaşterii	3	54
78.	22.	Formularea principiilor necesare la construirea unei ontologii	teoria cunoaşterii	3	55
79.	23.	Relevarea diferenţelor dintre logica bivalentă şi logica vagă	teoria cunoaşterii	3	62
80.	64.	Reprezentarea grafică ce ilustrează evoluţia gradului de abstractizare, în funcţie de nivelul la care se face reprezentarea	teoria cunoaşterii	5	103
81.	81.	Concluziile finale	toate domeniile	6	135

6.2. CONCLUZII FINALE

1. Acţiunea de dezvoltare a unui Sistem Expert este posibilă dacă, iniţiatorul poate să definească exact problema care trebuie rezolvată, domeniul de care aparţine şi scopul urmărit. Iniţiatorul programului trebuie să deţină o bază materială solidă, hardware şi software, precum şi o putere financiară suficientă pentru alcătuirea unei echipe pluridisciplinare cu o componenţă şi o calificare, adecvate obiectivului urmărit.

2. Iniţiatorul dezvoltării Sistemului Expert trebuie să fie o persoană suficient de influentă pentru a putea să atragă pe cei mai buni specialişti în domeniu şi să-i convingă să furnizeze informaţiile necesare dezvoltării sistemului.

3. Iniţiatorul trebuie să demonstreze că Sistemul Expert este necesar, iar folosirea lui aduce un avantaj şi constituie un progres tehnico-stiinţific.

4. Cunoştinţele din domeniul temei, trebuie să fie bine structurate şi să existe metode de control a consistenţei bazei de cunoştinţe. Activitatea aceasta trebuie efectuată de un specialist în teoria cunoaşterii.

5. Extragerea cunoştinţelor de dezvoltare a sistemului se efectuează din mai multe surse: din experienţa de proiectare, din experienţa de producţie, din experienţa de programare a

calculatoarelor, din experienţa utilizării calculatoarelor, din experienţa matematică în domeniu, din experienţa economică a societăţilor comerciale.

6. Sistemele Expert au fost creeate să rezolve o serie de probleme din domenii ca : medicină, matematică, inginerie, chimie, geologie, ştiinţa calculatoarelor, afaceri, drept, apărare şi educaţie. O lista restrânsă a categoriilor de probleme rezolvabile de către Sistemele Expert este următoarea :

- **Interpretare** - formarea concluziilor sau descrierilor de nivel înalt, despre colecţii de date native.
- **Predicţie** - predeterminarea consecinţelor probabile pentru situaţii date.
- **Diagnoză** - stabilirea stării de funcţionare a unui sistem, prin măsurarea şi evaluarea unui set accesibil de parametri.
- **Proiectare** - stabilirea configuraţiei unui sistem, care să atingă sigur performanţele cerute şi în acelaşi timp, să satisfacă un set de restricţii.
- **Planificare** - programarea unei secvenţe de acţiuni, care va completa un set de scopuri date, ca şi condiţii sigure de start.
- **Supraveghere** - compararea comportamentului efectiv al unui sistem, cu cel prognozat.
- **Depanare** - prescrierea şi implementarea remediilor, în caz de proastă funcţionare.
- **Instruire** - detectarea şi corectarea deficienţelor de înţelegere a unui subiect, de către utilizatori.
- **Control** - conducerea comportamentului mediilor complexe.

7. Alegerea unui Shell, de către o organizaţie, trebuie să ţină seama de următorii factori:

- caracteristicile aplicaţiei;
- caracteristicile proiectului;
- capacităţile Sistemelor Expert Shell;
- sofisticarea utilizatorilor;
- uşurinţa de integrare cu soft-ul existent;
- susţinerea vânzării.

8. Cele mai importante atribute ale Sistemelor Expert destinate proiectării sunt:

- existenţa legăturilor cu limbajele de generaţia a 3-a şi a 4-a;
- legătura cu bazele de date consacrate;
- legătura cu soft-ul special de proiectare;
- capacitatea de a scoate rapid un prototip;
- portabilitatea, implantabilitatea;
- accesul la limbajul de bază.

9. Fabricaţia este domeniul cel mai bine dotat cu Sisteme Expert.

10. Problema fundamentală a Inteligenţei Artificiale este cea de definire a unor metode pentru reprezentarea unei mari cantităţi de cunoştinţe, metodă care să permită stocarea şi utilizarea eficientă a acesteia. Metodele de reprezentare ale cunoaşterii pot fi grupate în

- metode logice;

- metode relaționale;
- metode procedurale.

11. Calitățile cerute unui limbaj, pentru a fi bun la dezvoltarea Sistemelor Expert sunt:

- să admită calculul simbolic;
- să permită un control flexibil;
- să admită metodele programării exploratorii;
- să admită amânarea propagării şi conectării restricțiilor;
- să posede o semantică bine definită şi clară.

12. Organizarea pe clase a domeniului transmisiilor mecanice este necesară pentru construirea bazei de cunoştințe, ca parte componentă a Sistemului Expert. Preîntâmpinarea apariției inconsistenței bazei de cunoştințe cere, printre altele, şi o riguroasă organizare a cunoştințelor din domeniu. Mijlocul de realizare al organizării unui domeniu pentru pregătirea dezvoltării unui Sistem Expert, constă în folosirea elementelor de organizare, furnizate de o ontologie evoluată.

13. Lucrarea şi-a propus să găsească o metodă adecvată de organizare a cunoştințelor şi de dezvoltare a Sistemelor Expert, dedicate activităților de proiectare, în general şi a proiectării transmisiilor mecanice, în particular. Structura lucrării, principiile enunțate, analizele efectuate, programele experimentate şi concluziile desprinse, costituie un ghid de încredere pentru continuarea efortului de dezvoltare al domeniului Inteligenței Artificiale.

ANEXA 1

Mulţimi Fuzzy

Mulţime = colecţie de elemente care aparţin unui anumit domeniu (în teoria clasică)

$x \in X$; L - domeniul de discuţie

$x \in A$; L - mulţime inclusă în X <=> $A \subseteq X$

Un element poate aparţine submulţimii A sau poate să nu aparţină submulţimii A.

$x \subseteq A$<=> "1" ; $x \not\subset A$ <=> "0"

1. A poate fi descrisă prin enumerarea elementelor cuprinse în ea.
2. A poate fi definită prin expresie analitică . Ex. : $A = \{ x \in X \mid x > 5 \}$
3. A poate fi definită prin funcţia caracteristică .

Mulţime Fuzzy Dacă notăm cu x o mulţime de obiecte aparţinând universului X, o mulţime vagă $\tilde{A} \subseteq X$, se defineşte ca o colecţie de perechi ordonate de forma :

$$\tilde{A} = \{ x, \mu_{\tilde{A}}(x) \mid x \in X \},$$

unde $\mu_{\tilde{A}}(x)$ este funcţia de apartenenţă a elemenelor mulţimii X la $\tilde{A}$

Pentru o anumită valoare a lui x, $\mu_{\tilde{A}}(x)$ poartă numele de grad de apartenenţă. Funcţia de apartenenţă poate să ia valori pozitive, cuprinse între 0 şi $\sup_x (\mu_{\tilde{A}}(x))$. In practică, se utilizează funcţii de operare normate, pentru care $\mu_{\tilde{A}}(x) \in [0, 1]$

Normalizarea funcţiei de operare se face împărţind cu sup ($\mu_{\tilde{A}}(x)$).

Importul unei mulţimi vagi S ($\tilde{A}$) este o mulţime nevagă a elementelor cuprinse în $\tilde{A}$ şi caracterizate prin : $\mu_{\tilde{A}}(x) > 0$

Mulţimi de nivel α sau tăietura α

Dacă se consideră o mulţime vagă $\tilde{A}(x)$, mulţimea de nivel α se defineşte ca o mulţime nevagă a elementelor cu grad de apartenenţă $\mu_{\tilde{A}}$ cel puţin egal cu α; $\alpha \in [0, 1]$

$$A_\alpha = \{ x \in X \mid \mu_{\tilde{A}}(x) \geq \alpha \}$$

Mulţime de grad α netă (strong) sau tăietura α netă

$$A_\alpha = \{ x \in X \mid \mu_{\tilde{A}}(x) > \alpha \} ; \ \alpha = [0, 1]$$

Convexitatea unei mulţimi vagi

se defineşte în raport cu funcţia de apartenenţă (forma ei). $\tilde{A}$ este o mulţime vagă, convexă, dacă este îndeplinită condiţia :

$$\mu_{\tilde{A}}(\lambda x_1 + (1 - \lambda)x_2) \geq \min (\mu_{\tilde{A}}(x_1), \mu_{\tilde{A}}(x_2)) ; \text{ unde } \lambda = [0, 1]$$

Cardinalitatea unei mulțimi vagi

mod de notare - | Ã | sau card Ã

Cardinalitatea servește la compararea a două mulțimi vagi. Ea se definește tot în raport cu funcția de apartenență, considerând o mulțime finită :

$$| \tilde{A} | = \sum_{i=1}^{n} \mu_A(x_i) \quad , \quad x \in X \text{ , unde n este numărul elementelor mulțimii vagi}$$

Cardinalitatea relativă se definește ca și raportul dintre cardinalitatea lui Ã și cea a universului în care este definită Ã. Card | X | este egală (în cazul unei mulțimi finite) cu numărul de elemente cuprinse în X.

Operații fundamentale cu mulțimi vagi

1. Intersecția

Fie două mulțimi vagi : Ã și Ñ , atunci Õ = Ã ∩ Ñ se definește astfel

$$\tilde{O} = \{ x, \mu_{A \cap N}(x) \mid x \in X \}$$

$\mu_{A \cap N}(x)$ se definește punctual, având valoarea egală cu valoarea minimă a gradului de apartenență pentru x dat, corespunzător celor două mulțimi Ã și Ñ.

$$\mu_{A \cap N}(x) = \min (\mu_A(x), \mu_N)$$

2. Reuniunea

Fie două mulțimi vagi : Ã și Ñ , atunci Õ = Ã ∪ Ñ se definește astfel:

$$\tilde{O} = \{ x, \mu_{A \cup N}(x) \mid x \in X \}; \text{ unde}$$

$$\mu_{A \cup N}(x) = \max (\mu_A(x), \mu_N)$$

3. Complementul unei mulțimi vagi

Se numește complementul unei mulțimi vagi mulțimea $C\tilde{A} = \{x, \mu_{C\tilde{A}}(x)\}$,

unde $\mu_{C\tilde{A}}(x) = 1 - \mu_{\tilde{A}}(x), \quad x \in X$

4. Produsul cartezian

Fie $\tilde{A}_1, \tilde{A}_2, \ldots, \tilde{A}_n$ definite în raport cu $x_1, x_2, \ldots, x_n$. Produsul cartezian este o mulțime vagă definită în spațiul produs $x_1 * x_2 * \ldots * x_n$ și are funcția de apartenență

$$\mu_{(\tilde{A}1 * \tilde{A}2 * \ldots * \tilde{A}n)}(x) = \min \{ \mu_{\tilde{A}i}(x) \mid x \quad (x_1, x_2, \ldots, x_n), \; x_i \in X_i \}$$

ANEXA 2

```
domains
        varianta=char
        cod,cod_profil=symbol
        gabarit,d_arb,l_arb,n_max,d_min,l_min,l_max=integer
        p3000,p1500,p1000,p750,m_min,m_max,po,ecart,f,hmin=real
        gabarite=gabarite(cod,gabarit,p3000,p1500,p1000,p750,d_arb,l_arb)
        trapez=trapez(cod_profil,varianta,m_min,m_max,n_max,d_min,po,l_min,l_max,ecart,
        hmin)

database - db1
        param_fam(symbol,real,real,real,real,real)
database - db2
        param_tip(symbol,symbol,char,symbol)
database - db3
        param_spec(symbol,symbol,char,symbol)
database - db4
        motor(gabarite)
database - db5
        t_profil(trapez)
database - db6
        diametre(real)
database - db7
        lungimi(integer)
database - db8
        c2(real,real)

predicates
        rezolvare
        mediu
        conversatie
        alege_familie(symbol)
        parametri_familie(symbol)
        qestionar_familie(real,real,real,real,real)
 nondeterm tip_parametri_f(symbol,real,real,real,real,real)
        alege_tip(symbol)
        parametri_tip(symbol)
        qestionar_tip(symbol,char,symbol)
 nondeterm tip_parametri_tip(symbol,symbol,char,symbol )
 nondeterm alege_vib(integer,symbol)
 nondeterm alege_cost(integer,symbol)
        alege_specimen(symbol)
        parametri_spec(symbol)
        qestionar_spec(symbol,char,symbol)
```

```
nondeterm tip_parametri_spec(symbol,symbol,char,symbol)
nondeterm alege_vib2(integer,symbol)
nondeterm alege_gabarit(integer,symbol)
        calculeaza
        utilizare(integer)
        select_utiliz(integer)
        ia_factor(integer,integer,real)
        motor_profil(real,symbol,integer,integer,integer,real,integer,integer,real)
nondeterm cauta_motor(integer,integer,real,symbol,integer,real,integer,integer)
        cauta_profil(real,integer,cod_profil,d_min,po,l_min,l_max,ecart,f,hmin)
        varianta(char,char)
        cauta_d(real,real)
        cauta_lungime(real,integer)
        cauta_c1(real,real)
        sterge

clauses
        rezolvare :- mediu,conversatie,write("\n\nReluati dialogul? d/n "),
                readchar(D),D<>'n',!,sterge,rezolvare.

        mediu :- makewindow(1,32,32,"",0,0,25,80),shiftwindow(1),clearwindow,
                makewindow(5,31,31," Oferte ",3,5,19,70),
                makewindow(2,79,64," dialog ",0,0,12,80),
                makewindow(3,32,32," observatii ",12,0,13,50),
                makewindow(4,12,12," avertizare ",12,50,13,30).

conversatie:- shiftwindow(2),clearwindow,alege_familie(Familie),shiftwindow(3),
                write("Se recomanda familia:\n",Familie),shiftwindow(2),
                clearwindow,alege_tip(Tip),shiftwindow(3),
                write("\nSe recomanda ",Familie," de tipul\n",Tip),shiftwindow(2),
                clearwindow,alege_specimen(Spec),shiftwindow(3),clearwindow,
                write("Se recomanda :\n",Tip," cu ",Spec),shiftwindow(2),clearwindow,
                calculeaza.
/************************ Alegere familie ********************************/
alege_familie(Familie):- parametri_familie(Familie).
parametri_familie(Tip_familie) :- qestionar_familie(P,Nmin,Nmax,Dist,Ra),
        tip_parametri_f(Tip_familie,P,Nmin,Nmax,Dist,Ra),
        asserta(param_fam(Tip_familie,P,Nmin,Nmax,Dist,Ra)),!.
qestionar_familie(Puterea,Nmin,Nmax,Dist,Ra):-
        write("Care este puterea consumata de masina de lucru ? [Kw]: "),
        readreal(Puterea),
        write("Care este turatia minima a masinii de lucru ? [rot/min]: "),
        readreal(Nmin),
        write("Care este turatia maxima a masinii de lucru ? [rot/min]: "),
        readreal(Nmax),
        write("\nCare este distanta aproximativa dintre masina de lucru\nsi masina motoare ?
        [mm]: "),
        readreal(Dist),
        write("\nCare este randamentul minim acceptat ? [%] "),
        readreal(Ra),clearwindow.
```

```
tip_parametri_f(Tip,P,N1,N2,D,R):-Tip="transmisie mecanica",P<1000,
        N1>30,N2<50000,D<=2000,R>=90.
tip_parametri_f(Tip,P,N1,N2,D,R):-Tip="transmisie mecanica",P<1500,
        N1>100,N1=N2,D<=2000,R>=90.
tip_parametri_f(Tip,P,N1,N2,D,R):-Tip="transmisie eletrica",P>=1000,
        N1>30,N2<50000,D>500,R<95,R>50.
tip_parametri_f(Tip,P,N1,N2,D,R):-Tip="transmisie hidraulica",P>10,
        N1>0,N2<3000,D>500,R<=80.
/*---------------------------------- Alege tip ----------------------------------*/
alege_tip(Tip):- retract(param_fam(Mec,A,B,C,D,E)),Mec="transmisie mecanica",
                !,asserta(param_fam(Mec,A,B,C,D,E)), parametri_tip(Tip).
parametri_tip(Tip) :- qestionar_tip(Vib,Pr,Cost),
        tip_parametri_tip(Tip,Vib,Pr,Cost),
        asserta(param_tip(Tip,Vib,Pr,Cost)),!.
qestionar_tip(Vib,Pr,Cost):-
        write("\nCare este nivelul acceptat al vibratiilor produse de transmisie ?\n1.
        Neglijabil\n2. Scazut\n3. Mediu\n"),
        readint(N_vib),N_vib<4,
        write("\nTransmisia sa coste\n1. Moderat \n2. Costul nu conteza !\n"),
        readint(N_cost),N_cost<3,
        alege_vib(N_vib,Vib),
        write("\nDoriti protectie la suprasarcini ? d/n "),
        readchar(Pr),write(Pr),alege_cost(N_cost,Cost),!.
qestionar_tip(Vib,Pr,Cost):-shiftwindow(4),
        write("\nAtentie la indici !!! "),shiftwindow(2),
        qestionar_tip(Vib,Pr,Cost).
alege_vib(1, "vib_neglijabil").
alege_vib(2, "vib_scazut").
alege_vib(3, "vib_mediu").
alege_cost(1,"cost_moderat").
alege_cost(2,"cost_mare").
tip_parametri_tip(Tip,Vib,Pr,Cost):-Tip="transmisie prin curea",
                Vib="vib_neglijabil",Pr='d',Cost="cost_moderat"
tip_parametri_tip(Tip,Vib,Pr,Cost):-Tip="transmisie prin curea",
                Vib="vib_scazut",Pr='d',Cost="cost_moderat".
tip_parametri_tip(Tip,Vib,Pr,Cost):-Tip="transmisie prin lant",
                Vib="vib_mediu",Pr='n',Cost="cost_moderat".
tip_parametri_tip(Tip,Vib,Pr,Cost):-Tip="transmisie prin angrenaje",
                Vib="vib_neglijabil",Pr='n',Cost="cost_mare".
tip_parametri_tip(Tip,Vib,Pr,Cost):-Tip="transmisie prin angrenaje",
                Vib="vib_scazut",Pr='n',Cost="cost_moderat"
tip_parametri_tip(Tip,Vib,Pr,Cost):-Tip="transmisie prin angrenaje",
                Vib="vib_scazut",Pr='n',Cost="cost_mare"
tip_parametri_tip(Tip,Vib,Pr,Cost):-Tip="transmisie prin curea sincrona",
                Vib="vib_neglijabil",Pr='n',Cost="cost_moderat"
/********************* Alege specimen ***********************************/
alege_specimen(Spec):- retract(param_tip(Tip,Vib,Pr,Cost)),Tip="transmisie prin curea",
                !,asserta(param_tip(Tip,Vib,Pr,Cost)), parametri_spec(Spec).
parametri_spec(Spec) :- qestionar_spec(Vib,Sc,Gaba),
                        tip_parametri_spec(Spec,Vib,Sc,Gaba).
```

```
                    asserta(param_spec(Spec,Vib,Sc,Gaba)),!.
qestionar_spec(Vib,Sc,Gaba):-
        write("\nSe doreste o functionare ?\n1. Deosebit de linistita\n2. Cu vibratii mici\n3. Cu
        vibratii medii\n"),
        readint(N_vib),N_vib<4,alege_vib2(N_vib,Vib),
        write("\nDoriti miscare sincrona la arborii transmisiei ? d/n "),
        readchar(Sc),write(Sc),
        write("\n\nDoriti gabarit radial ?\n1. Mic \n2. Moderat\n"),
        readint(N_gaba),N_gaba<3,alege_gabarit(N_gaba,Gaba),!.
qestionar_spec(Vib,Sc,Gaba):-shiftwindow(4),
        write("\nAtentie la indici !!! "),shiftwindow(2),
        qestionar_tip(Vib,Sc,Gaba).
alege_vib2(1, "foarte_lin").
alege_vib2(2, "lin").
alege_vib2(3, "tremolo").
alege_gabarit(1,"g_mic").
alege_gabarit(2,"g_moderat").
tip_parametri_spec(Tip,Vib,Sc,Gaba):-Tip="profil trapezoidal",
                Vib="lin",Sc='n',Gaba="g_moderat".
tip_parametri_spec(Tip,Vib,Sc,Gaba):-Tip="profil trapezoidal",
                Vib="tremolo",Sc='n',Gaba="g_moderat".
tip_parametri_spec(Tip,Vib,Sc,Gaba):-Tip="profil lat",
                Vib="foarte_lin",Sc='n',Gaba="g_mic".
tip_parametri_spec(Tip,Vib,Sc,Gaba):-Tip="profil lat",
                Vib="lin",Sc='n',Gaba="g_mic".
tip_parametri_spec(Tip,Vib,Sc,Gaba):-Tip="sincrona",
                Vib="lin",Sc='d',Gaba="g_mic".
tip_parametri_spec(Tip,Vib,Sc,Gaba):-Tip="sincrona",
                Vib="tremolo",Sc='d',Gaba="g_mic".
/*------------------------------ Detalii --------------------------------*/
calculeaza :- utilizare(Index),clearwindow,
        select_utiliz(Grupa),ia_factor(Index,Grupa,Factor),
        write("\nCoef. de functionare: ",Factor),shiftwindow(3),
        shiftwindow(2),motor_profil(Factor,Cod,Gabarit,Nmin,Nm,X,D,L,Pc),
        shiftwindow(4),clearwindow,shiftwindow(3),clearwindow,
        write("\nCod motor: ",Cod,"\nGabarit: ",Gabarit," [mm]","\nTuratie ",Nm,"
        [rot/min]","\nPutere : ",X," [kW]","\nDiametru arbore ",D," [mm]",
        "\nLungime arbore: ",L," [mm]","\nPutere de calcul : ",Pc," [kW]"),
        shiftwindow(2),cauta_profil(Pc,Nm,Cod_profil,Dmin,Po,Lmin,Lmax,Ecart,F,Hmin),
        Rap=Nm/Nmin,D2=Dmin*Rap,cauta_d(D2,Dmax),
        write("\nDistanta dintre axe ? [mm] :"),readint(Axe),
        Lw=2*Axe+1.57*(Dmin+Dmax)+(Dmax-Dmin)*(Dmax-Dmin)/(4*Axe),
        cauta_lungime(Lw,Lefectiv),
        Fr=(Lefectiv-3.14159*(Dmin+Dmax)/2)/4,
        Aef=Fr+sqrt(Fr*Fr-(Dmax-Dmin)*(Dmax-Dmin)/8),
        Cc1=(Dmax-Dmin)/Aef,
        cauta_c1(Cc1,C1),Z=round(Pc/(Po*C1)),shiftwindow(5),
        write("Se recomanda profilul : ",Cod_profil),
        write("\nDiametrul minim de roata : ",Dmin," [mm]"),
        write("\nPuterea unitara : ",Po," [kW]"),
```

```
        write("\nLungimea minima a curelei : ",Lmin," [mm]"),
        write("\nLungimea maxima a curelei : ",Lmax," [mm]"),
        write("\nDistanta dintre canale : ",Ecart," [mm]"),
        write("\nDistanta de la marginea coroanei : ",F," [mm]"),
        write("\nAdincimea minima a canalului : ",Hmin," [mm]"),
        write("\nRaportul de transmitere : ",Rap),
        write("\n        D1 = ",Dmin," [mm]"),
        write("\n        D2 = ",Dmax," [mm]"),
        write("\nLungimea necesara a curelei : ",Lw," [mm]"),
        write("\nLungimea efectiva a curelei : ",Lefectiv," [mm]"),
        write("\nDistanta efectiva dintre axe : ",Aef," [mm]"),
        write("\nNumarul de curele : ",Z).
utilizare(Index):- write("Cite ore functioneaza pe zi ?\n1. Mai putin de 8 ore \n2. Intre 8 si 16
                ore \n3. Intre 16 si 24 ore\n"),readint(Index).
select_utiliz(Grupa) :- shiftwindow(1),clearwindow,
        write("In care dintre grupele de utilizatori va incadrati ?\n"),
write("\n1. Pompe, compresoare, ventilatoare, generatoare electrice usoare\n   Site usoare,
        separatoare, transportoare cu banda\n"),
write("\n2. Pompe cu piston, generatoare electrice, masini de frezat\n   Strunguri revolver,
        Masini pentru industriile:\n   alimentara, textila si hirtie. Site grele, cuptoare
        rotative\n"),
write("\n3. Ventilatoare grele, dezintegratoare, transportoare elicoidale\n   Masini de rabotat,
        mortezat si polizat\n   Prese cu surub si cu excentric. Masini de tesut.\n"),
write("\n4. Masini de ridicat, escavat si dragat. Foarfeci mecanice\n   Mori cu bile, mori cu
        pietre, mori cu valturi\n   Concasoare, malaxoare\n"),readint(Grupa),shiftwindow(3).
ia_factor(1,1,1.1).
ia_factor(1,2,1.1).
ia_factor(1,3,1.2).
ia_factor(1,4,1.3).
ia_factor(2,1,1.1).
ia_factor(2,2,1.2).
ia_factor(2,3,1.3).
ia_factor(2,4,1.4).
ia_factor(3,1,1.2).
ia_factor(3,2,1.3).
ia_factor(3,3,1.4).
ia_factor(3,4,1.5).
motor_profil(Factor,Cod,Gabarit,Nmin,Nm,X,D,L,Pc1):-
retract(param_fam(_,P,Nmin,_,_,Ra)),
        Pc=P*100/Ra,Pc1=Pc*Factor,consult("c:\\prol\\pdc\\myprog\\d_motor.dat",db4),
        cauta_motor(Nmin,Nm,Pc,Cod,Gabarit,X,D,L),!.
cauta_motor(Nmin,Nm,Pc,Cod,Gabarit,X,D,L):- Nmin>1500,Nm=3000,
                motor(gabarite(Cod,Gabarit,X,_,_,_,D,L)),
                X>0,((X-Pc)/(abs(X-Pc))-(Pc-X)/(abs(Pc-X)))=2.
cauta_motor(Nmin,Nm,Pc,Cod,Gabarit,X,D,L):- Nmin>1000,Nm=1500,
                motor(gabarite(Cod,Gabarit,_,X,_,_,D,L)),
                X>0,((X-Pc)/(abs(X-Pc))-(Pc-X)/(abs(Pc-X)))=2
                X>0,((X-Pc)/(abs(X-Pc))-(Pc-X)/(abs(Pc-X)))=2.
cauta_motor(Nmin,Nm,Pc,Cod,Gabarit,X,D,L):- Nmin>750,Nm=1000,
                motor(gabarite(Cod,Gabarit,_,_,X,_,D,L)),
                X>0,((X-Pc)/(abs(X-Pc))-(Pc-X)/(abs(Pc-X)))=2.
```

```
cauta_motor(Nmin,Nm,Pc,Cod,Gabarit,X,D,L):- Nmin<=750,Nm=750,
                motor(gabarite(Cod,Gabarit,_,_,_,X,D,L)),
                X>0,((X-Pc)/(abs(X-Pc))-(Pc-X)/(abs(Pc-X)))=2.
cauta_profil(Pc,Nm,Cod_profil,Dmin,Po,Lmin,Lmax,Ecart,F,Hmin):-
        write(" Pentru ce varianta optati ?\n\n 1. Profil trapezoidal clasic\n\n 2. Profil
        trapeziodal ingust\n"),
        readchar(Nv),varianta(Nv,V),Mtors=Pc*9550/Nm,
        consult("\\prol\\pdc\\myprog\\d_curea.dat",db5),
        t_profil(trapez(Cod_profil,V,Mmin,Mmax,Nmax,Dmin,Po,Lmin,Lmax,Ecart,F,Hmin)),
        Mtors>=Mmin,Mtors<Mmax,Nm<=Nmax,!.
varianta('1','c').
varianta('2','i').
cauta_d(D2,Dmax):-consult("\\prol\\pdc\\myprog\\diam.dat",db6),
        diametre(D),D>=D2,!,D=Dmax.
cauta_lungime(Lw,Lreal):-consult("\\prol\\pdc\\myprog\\lungimi.dat",db7),
        lungimi(L),L>=Lw,!,L=Lreal.
cauta_c1(Cc1,C1):-consult("\\prol\\pdc\\myprog\\c2.dat",db8),
        c2(C,V),C>=Cc1,!,V=C1.
sterge:-retractall(param_fam(_,_,_,_,_,_)),retractall(param_tip(_,_,_,_)),
        retractall(param_spec(_,_,_,_)),
        retractall(motor(gabarite(_,_,_,_,_,_,_,_,_))),
        retractall(t_profil(trapez(_,_,_,_,_,_,_,_,_,_,_,_,_,_))),
        retractall(diametre(_)),
        retractall(lungimi(_)),retractall(c2(_,_)).

goal
        rezolvare.
```

ANEXA 3

```
domains
        varianta=char
        cod,cod_profil=symbol
        gabarit,d_arb,l_arb,n_max,d_min,l_min,l_max=integer
        p3000,p1500,p1000,p750,m_min,m_max,po,ecart,f,hmin=real
        gabarite=gabarite(cod,gabarit,p3000,p1500,p1000,p750,d_arb,l_arb)
        trapez=trapez(cod_profil,varianta,m_min,m_max,n_max,d_min,po,l_min,l_max,ecart,f,
        hmin)

database - db1
        motor(gabarite)
database - db2
        t_profil(trapez)
database - db3
        diametre(real)
database - db4
        lungimi(integer)
database - db5
        c2(real,real)

predicates
        rezolvare
        mediu
        conversatie
        introd_motor
        invirte
        confirm
        invirte2
        confirm2
        introd_curea
        introd_diam
        invirte3
        confirm3
        introd_lung
        invirte4
        confirm4
        introd_c2
        invirte5
        confirm5

clauses
        rezolvare :- mediu,mouse_on,conversatie,mouse_off,!
```

```
        mediu :- makewindow(1,0,0,"",0,0,25,80),shiftwindow(1),clearwindow,
                 makewindow(2,79,64," dialog ",0,0,12,80),
                 makewindow(3,32,32," observatii ",12,0,13,50),
                 makewindow(4,12,12," avertizare ",12,50,13,30).

conversatie:- shiftwindow(2),NOT(introd_motor),NOT(introd_curea),
              NOT(introd_diam),NOT(introd_lung),NOT(introd_c2).
/************************** Motor ******************************/
introd_motor:-clearwindow,write("Introduceti date despre motor? d/n"),
        readchar(D),D='d',!,consult("\\prol\\pdc\\myprog\\d_motor.dat",db1),
        NOT(invirte),save("\\prol\\pdc\\myprog\\d_motor.dat",db1).
invirte:-write("\nCodul motorului: "),readln(Cod),
        write("\Gabaritul: "),readint(Gaba),
        write("\P3000 : "),readreal(P3000),
        write("P1500 : "),readreal(P1500),
        write("P1000 : "),readreal(P1000),
        write("P750 : "),readreal(P750),
        write("\Diametrul arborelui : "),readint(D_arb),
        write("Lungimea arborelui : "),readint(L_arb),
        assertz(motor(gabarite(Cod,Gaba,P3000,P1500,P1000,P750,D_arb,L_arb))),
        confirm,!,invirte.
confirm:-write("\nContinuati ? d/n "),readchar(V),V<>'n'.
/***************************** Curea ****************************/
introd_curea:-clearwindow,write("Introduceti date despre curea? d/n"),
        readchar(D),D='d',!,consult("\\prol\\pdc\\myprog\\d_curea.dat",db2),
        NOT(invirte2),save("\\prol\\pdc\\myprog\\d_curea.dat",db2).
invirte2:-write("\nCodul curelei: "),readln(Cod_profil),
        write("c pentru clasic i pentru ingust : "),readchar(Var),
        write("Cuplul minim: "),readreal(Mmin),
        write("Cuplul maxim: "),readreal(Mmax),
        write("Turatia maxima: "),readint(Nmax),
        write("Diametrul minim : "),readint(Dmin),
        write("Puterea unitara : "),readreal(Po),
        write("Lungimea minima : "),readint(Lmin),
        write("Lungimea maxima : "),readint(Lmax),
        write("Ecartul dintre canale : "),readreal(Ec),
        write("Ecartul de la margine : "),readreal(F),
        write("Adincimea canalului : "),readreal(Hmin),
        asserta(t_profil(trapez(Cod_profil,Var,Mmin,Mmax,Nmax,Dmin,Po,Lmin,Lmax,Ec,F,
        Hmin))),
        confirm2,!,invirte2.
confirm2:-write("\nContinuati ? d/n "),readchar(V),V<>'n'.
/*---------------------------------- Diametre si lungimi ----------------------------------*/
introd_diam:-clearwindow,write("Introduceti diametre ? d/n"),
        readchar(D),D='d',!,consult("\\prol\\pdc\\myprog\\diam.dat",db3),
        NOT(invirte3),save("\\prol\\pdc\\myprog\\diam.dat",db3).
invirte3:-write("Diametrul : "),readreal(Diam),
        assertz(diametre(Diam)),
        confirm3,!,invirte3.
confirm3:-write("\nContinuati ? d/n\n"),readchar(V),V<>'n'.
```

```
introd_lung:-clearwindow,write("Introduceti lungimi ? d/n"),
        readchar(D),D='d',!,consult("\\prol\\pdc\\myprog\\lungimi.dat",db4),
        NOT(invirte4),save("\\prol\\pdc\\myprog\\lungimi.dat",db4).
invirte4:-write("Lungimea : "),readreal(Lung),
        assertz(lungimi(Lung)),
        confirm4,!,invirte4.
confirm4:-write("Continuati ? d/n\n"),readchar(V),V<>'n'.
/*---------------------------------- Coeficienti ----------------------------------------*/
introd_c2:-clearwindow,write("Introduceti C2 ? d/n"),
        readchar(D),D='d',!,consult("\\prol\\pdc\\myprog\\c2.dat",db5),
        NOT(invirte5),save("\\prol\\pdc\\myprog\\c2.dat",db5).
invirte5:-write("\n(Dmax-Dmin)/a : "),readreal(Val),
        write(" C2 : "),readreal(Coef),
        assertz(c2(Val,Coef)),
        confirm5,!,invirte5.
confirm5:-write("Continuati ? d/n\n"),readchar(V),V<>'n'.

goal
        rezolvare.
```

ANEXA 4

Dialog

Care este puterea consumată de maşina de lucru ? [kW]	4
Care este turaţia minimă a maşinii de lucru ? [rot/min]	1050
Care este turaţia maximă a maşinii de lucru ? [rot/min]	1050
Care este distanţa aproximativă dintre maşina de lucru şi maşina motoare ? [mm]	900
Care este randamentul minim acceptat ? [%]	92

Observaţii

Avertizări

Dialog

Care este nivelul acceptat al vibraţiilor produse de transmisie ?
1. Neglijabil
2. Scăzut
3. Mediu

2. (Scăzut)

Transmisia poate să coste ?
1. Moderat
2. Costul nu contează la selecţie

1.(Moderat)

Observaţii

Se recomandă familia:
transmisie mecanică

Avertizări

Dialog

Doriţi protecţie la suprasaccini ? d/n — d

Se doreşte o funcţionare ?
1. Deosebit de liniştită
2. Cu vibraţii mici — 2.(cu vibraţii mici)
3. Cu vibraţii medii

Observaţii

Se recomandă familia:
transmisie mecanică

Se recomandă transmisia de tipul:
transmisie prin curea

Avertizări

Dialog

Doriţi mişcare sincronă la arborii transmisiei ? d/n — n

Doriţi gabarit radial ?
1. Mic
2. Moderat — 2.(Moderat)

Câte ore funcţionează transmisia pe zi ?
1. Mai puţin de 8 ore
2. Intre 8 şi 16 ore — 2.(8 - 16 ore)
3. Intre 16 şi 24 ore

Observaţii

Se recomandă:
transmisia prin curea cu profil trapezoidal

Avertizări

Dialog

In care dintre grupele de utilizatori vă încadraţi ?

1. Pompe, compresoare, ventilatoare, generatoare electrice uşoare. Site uşoare, separatoare, transportoare cu bandă.

2. Pompe cu piston, generatoare electrice, maşini de frezat. Strunguri revolver, Maşini pentru industriile: textilă, alimentară, şi hârtie. Site grele. Cuptoare rotative. 2.(pompe cu piston)

3. Ventilatoare grele, dezintegratoare, trasportoare elicoidale, maşini de rabotat, mortezat şi polizat. Prese cu şurub şi cu excentric.

4. Maşini de ridicat, escavat şi dragat. Foarfeci mecanice. Mori cu bile. Mori cu pietre. Mori cu valţuri. Concasoare. Malaxoare.

Dialog

Pentru ce variantă optaţi ?

1. Profil trapezoidal clasic

2. Profil trapezoidal îngust 2.(profil îngust)

Distanţa dintre axe ? [mm] 850

Observaţii

Cod motor :	132 S
Gabarit :	132 [mm]
Turaţie :	1500 [mm]
Putere :	5.5 [kW]
Diametrul arborelui :	38 [mm]
Lungimea arborelui :	80 [mm]
Puterea de calcul :	5.21 [kW]

Avertizări

Dialog

Oferte

Se recomandă profilul	SPZ
Diametru minim de roată	63 [mm]
Lungimea minimă a curelei	630 [mm]
Lungimea maximă a curelei	3550 [mm]
Distanța dintre canale	12 [mm]
Distanța de la marginea coroanei	8 [mm]
Adâncimea minimă a canalului	11 [mm]
Raportul de transmitere	1,428
D_1 : 63 [mm]	
D_2 : 90 [mm]	
Lungimea necesară a curelei	1949 [mm]
Lungimea efectivă a curelei	2000 [mm]
Distanța efectivă dintre axe	879 [mm]
Numărul de curele	5

Reluați dialogul ? d/n d

Dialog

Care este puterea consumată de mașina de lucru ? [kW]	10
Care este turația minimă a mașinii de lucru ? [rot/min]	800
Care este turația maximă a mașinii de lucru ? [rot/min]	800
Care este distanța aproximativă dintre mașina de lucru și mașina motoare [mm]	1500
Care este randamentul minim acceptat ? [%]	94

Observații

Avertizări

BIBLIOGRAFIE

1.	35.	Aristotel	*Organon*	Editura IRI, Bucureşti	1997
2.	59.	Boroş E., Opriş D.	*Introducere în Optimizarea Liniară*	Editura Facla. Timişoara	1979
3.	31.	Bratko Ivan	*PROLOG Programming for Artificial Intelligence*	Addison-Wesley Publi shing Comp.. England	1988
4.	71.	Bundy A.	*Catalogue of Artificial Intelligence Tools*	Springer Verlag Berlin	1984
5.	43.	Burdescu D.	*Algoritmi şi Structuri de Date*	Editura Mirton. Timişoara	1992
6.	75.	Buzdugan Gh.	*Vibraţii Mecanice*	EDP. Bucureşti	1979
7.	97.	Buzdugan Gheorghe s.a.	*Măsurarea Vibraţiilor*	Editura Academici. Bucureşti	1979
8.	39.	Căprariu Vlad	*Turbo C 2.0. Ghid de Utilizare*	Editura Microinformatica. Cluj-Napoca	1991
9.	15.	Cârstoiu Dorin	*Sisteme Expert*	Editura ALL. Bucureşti	1995
10.	38.	Catrina Octavian. Cojocaru Iuliana	*Turbo C++*	Editura Teora. Bucureşti	1993
11.	79.	Chişiu Al.	*Organe de Maşini*	EDP. Bucureşti	1981
12.	62.	de Palma Raoul	*Algebra Binară a lui Boole şi Aplicaţiile ei în Informatică*	Editura Tehnică. Bucureşti	1976
13.	58.	Deacu L.	*Vibraţii la Maşini Unelte*	Editura Dacia. Cluj-Napoca	1977
14.	81.	Dehelean Nicolae Ioanovici Fr.jr. Dreuceam Mircea	*Instalaţie Experimentală pentru Studiul Caracteristicilor unei Transmisii cu Arbore Flexibil*	Simpozion Naţional de Roboţi MTM Cluj-Napoca	1988
15.	82.	Dehelean Nicolae Gheorghiu Nicolae Dreucean Mircea	*Predimensionarea Roţilor Dinţate Cilindrice*	Simpozion Naţional de Roboţi MTM Cluj-Napoca	1988
16.	83.	Dehelean Nicolae Luchin Milenco Ioanovici Fr.jr.	*Schematizarea Geometrică a Roţilor de Fricţiune cu Contact Punctual pentru Evaluarea Tensiunilor de Contact*	Simpozion Naţional de Roboţi MTM Cluj-Napoca	1988
17.	84.	Dehelean Nicolae Luchin Milenco Ioanovici Fr.jr.	*Analiza Morfologică pentru Conceptia unei Frâne Hidraulice Miniatură*	Simpozion Naţional de Roboţi MTM Cluj-Napoca	1988
18.	85.	Dehelean Nicolae Dreucean Angela Argeşanu Vera	*Metodă şi Dispozitiv pentru Debitarea la Lăţime a Curelelor Sincrone*	Simpozion Naţional de Roboţi MTM Cluj-Napoca	1988
19.	86.	Dehelean Nicolae	*Alegerea Oţelurilor pentru Roţile Dinţate*	Simpozion Naţional MTM Timişoara	1992
20.	87.	Dehelean Nicolae	*Predimensionarea Angrenajelor Melcate*	Simpozion Naţional MTM Timişoara	1992
21.	88.	Dehelean Nicolae Dreucean Mircea Radu Călin	*Program de Simulare al Obturatoarelor cu Perdea*	Simpozion Naţional MTM Timişoara	1992
22.	89.	Dehelean Nicolae	*Predimensionarea Angrenajelor Conice*	Simpozion Naţional MTM Timişoara	1992
23.	90.	Dehelean Nicolae Dreucean Mircea Radu Călin	*Program de Testare pentru Laboratorul de Organe de Maşini*	Simpozion Naţional MTM Timişoara	1992
24.	91.	Dehelean Nicolae	*Algoritm pentru Proiectarea Transmisiilor cu Lanţ*	Simpozion Naţional MTM Timişoara	1992

25.	92.	Dehelean Nicolae Radu Călin	*Variante Constructive de Reductoare Navale*	Sesiunea de Comunicări Universitatea Constanţa	1993
26.	93.	Dehelean Nicolae Gheorghiu Nicolae	*Bază de Date Dedicată Proiectării Transmisiilor cu Element Flexibil*	Simpozion Naţional cu participare Internaţiona lă PRASIC '94, Braşov	1994
27.	94.	Dehelean Nicolae	*Organizarea Cunoştinţelor şi Simplificarea Căutării în Proiectare*	Simpozion Naţional cu participare Internaţ. MTeM'95, Cluj-Napoca.	1995
28.	95.	Dehelean Nicolae	*Complexitatea Tehnologică şi de Cunoaştere a Sistemelor Expert*	PSCM of "Aurel Vlaicu" University Arad	1996
29.	13.	Dehelean Nicolae Mircea	*Evaluarea Sistemelor Expert*	Simpozion MTM Timişoara	1996
30.	14.	Dehelean Nicolae Mircea	*Baza de Cunoştinţe pentru Sisteme Expert Multi-Domeniu*	Simpozion MTM Timişoara	1996
31.	51.	Dehelean Nicolae Mircea	*Stadiul Actual al Conceptului de Sistem Expert. Referatul Nr.1.*		1993
32.	52.	Dehelean Nicolae Mircea	*Particularităţile Proiectării Asistate de Calculator în Construcţia de Maşini. Referatul Nr.2.*		1994
33.	45.	Dima Gabriel, Dima Mihai	*FOXPRO*	Editura Teora, Bucureşti	1993
34.	73.	Drăghici I., ş.a.	*Indrumar de Proiectare în Construcţia de Maşini. Vol.I*	Editura Tehnică, Bucureşti	1981
35.	74.	Drăghici I., ş.a.	*Indrumar de Proiectare In construcţia de Maşini. Vol.II*	Editura Tehnică, Bucureşti	1982
36.	54.	Drăghici Ioan	*Indrumar de Proiectare în Construcţia de Maşini*	Editura Tehnică, Bucureşti	1982
37.	50.	Ermine Jean-Louis	*Systems Experts*	Technicque et Documentation - Lavoisier, France	1989
38.	76.	Florea Viorel ş.a.	*Organe de Maşini. Vol.I*	Lito Sibiu	1984
39.	56.	Gafiţanu Mihai	*Organe de Maşini. Vol.II*	Editura Tehnică, Bucureşti	1983
40.	33.	Georgescu Stefan	*Teoria Cunoaşterii Stiinţifice*	Editura Academici, Bucureşti	1982
41.	112.	Gheorghiu N., ş.a.	*Transmisii Mecanice. Proiectare.*	Editura Felix, Timişoara	1997
42.	57.	Gheorghiu N., Ionescu N.	*Organe de Maşini. Transmisii Mecanice*	Lito IPTV Timişoara	1982
43.	65.	Goel K. Ashok, B.Chandrasekaran	*Case - Based Design: A Task Analysis*	Academic Press Inc.	1992
44.	34.	Gundberg Ludwig	*Ontologia Umanului*	Editura Academici, Bucureşti	1989
45.	101.	Hague M.J., Taleb-Bendiab A. & Brandish M.J.	*An Adaptive Machine Learning System for Computer Supported Conceptual Engineering Design*	Arizona University	1997
46.	78.	Handra-Luca Viorel	*Organe de Maşini şi Mecanisme*	EDP, Bucureşti	1975
47.	70.	Hayes-Roth Fr., Waterman D.A., Lanat D.B	*Building Expert Sistems*	Addison-Wesley Publi shing Comp., England	1983
48.	36.	Heidegger Martin	*Repere pe Drumul Gândirii*	Editura Politică, Bucureşti	1988
49.	63.	Hristev Ryurick	*Introducere în PROLOG un Limbaj al Inteligenţei Artificiale*	Editura APH, Bucureşti	1992
50.	64.	Huhns N. M. Acosta D. Ramon	*Argo: An Analogical Reasoning System for Solving Design Problems*	Academic Press Inc.	1992
51.	48.	Jula Aurel	*Proiectarea Angrenajelor Evolventice*	Editura "Scrisul Romă nesc", Craiova	1989
52.	102.	Keat L.B., Tan C.L. & Mathur K.	*Design Model: Towards an Integrated Representation for Design Semantics and*	Arizona University	1997

			Syntax		
53.	25.	Klaus Georg	*Logica Modernă*	Editura Stiinţifică şi Enciclopedică, Bucureşti	1977
54.	32.	Laporte J., Delport D.	*TURBO PROLOG*	Editions Eyrolles, Paris	1988
55.	98.	Lenat D.B., Guha R.V.	*Building Large Knowledge Based Systems*	Addison-Weskey Publi shing Company, Inc.	1997
56.	100.	Lim Peng Cherkassky V.	*Semantic Networks and Associative Databases*	IEEE Expert, University of Minnesota	1992
57.	7.	Luger George F., Stubblefeld William	*Artificial Intelligence and the Design of Expert Systems*	Benjamin/Cummings Publ. Company, Inc., California	1991
58.	68.	Lungu I, Muşat N.	*Baze de Date Relaţionale - SQL*Plus*	Editura ALL, Bucureşti	1993
59.	19.	Lungu I., Bedea C., ş.a.	*Baze de Date. Organizare, Proiectare şi Implementare*	Editura ALL, Bucureşti	1995
60.	44.	Lungu Ion	*Sistemul FOXPRO 2.6.*	Editura ALL, Bucureşti	1996
61.	72.	Manolescu N., ş.a.	*Manualul Inginerului Mecanic. Mecanisme, Organe de Maşini, Dinamica Maşinilor.*	Editura Tehnică, Bucureşti	1976
62.	1.	Marcu F., Manea C.	*Dicţionar de Neologisme*	Editura Academiei, Bucureşti	1978
63.	16	Masterman M.	*Semantic Message Detection for Machine Translation*	International Conference on Machine Translation	1961
64.	53.	Matek W., Muhs D., Wittel H.	*Roloff / Matek Maschinenelemente. Tabellen*	Springer Verlag Berlin	1990
65.	27.	Meszaros Iudith	*TURBO PROLOG 2.0. Ghid de Utilizare.*	Editura Albastră, Cluj-Napoca	1996
66.	12.	Meyer H. Marc, Curley Kathleen Foley	*The Impact of Knowledge and Technology Complexity on Information Systems Development.*	Expert Systems With Applications, vol.8, nr.1, p.111, Elsevier Science Ltd, Pergamon	1995
67.	104.	Mirbel I.	*A Fuzzy Thesaurus for Semantic Integration of Design Schemes*	Arizona University	1997
68.	69.	Moose A., Shafer D., Sawer B.	*VP - Expert*		1987
69.	40.	Oliver Dick	*Fractali*	Editura Teora, Bucureşti	1992
70.	77.	Paizi Gheorghe	*Organe de Maşini şi Mecanisme*	EDP, Bucureşti	1976
71.	55.	Pavelescu D., Gheorghiu N.	*Organe de Maşini. Vol.I.*	EDP, Bucureşti	1985
72.	37.	Peirce Ch.	*Semnificaţie şi Acţiune*	Editura Humanitas, Bucureşti	1990
73.	5.	Popescu Tiberiu	*Dicţionar De Informatică*	Editura Stiinţifică şi Enciclopedică, Bucureşti	1981
74.	17.	Quillian Ross	*Word Concepts: A Theory and Simulation of Some Basic Semantic Capabilities*	Brachman and Levesque	1985
75.	42.	Rădulescu D., Gheorghiu O.	*Optimizarea Flexibilă şi Decizia Asistată de Calculator*	Editura Stiinţifică, Bucureşti	1992
76.	47.	Rădulescu Gh.	*Indrumar de Proiectare în Construcţia de Maşini*	Editura Tehnică, Bucureşti	1986
77.	46.	Rădulescu Octavian	*Sinteze Optimale în Construcţia de Maşini.*	Editura Tehnică, Bucureşti	1984
78.	4.	Roşca Al., Chircev A	*Psihologie Generală*	EDP, Bucureşti	1975
79.	67.	Savii George	*Proiectarea Asistată de Calculator*	Editura Mirton, Timişoara	1996
80.	6.	Sfez Lucien	*Dictionaire Critique de la Communication*	Press Universtaire de France	1993

81.	21.	Shih-Yang Liu, Jen-Gwo Chen	*Development of a Machine Troubleshooting Expert System Via Fuzzy Multiattribute Decision-Making Approach*	Expert Systems with Applications vol.8, nr.1, p.187, Elsevier Science Ltd, Pergamon	1995
82.	41.	Sima V, Varga A	*Practica Optimizării Asistată de Calculator*	Editura Tehnică, Bucureşti	1986
83.	9.	Stoianovici D.,ş.a.	*Logica Generală*	EDP, Bucureşti	1990
84.	80.	Stoica M., ş.a.	*Introducere în Modelarea Procedurală*	Editura "Scrisul Românesc", Craiova	1989
85.	11.	Stylianou C. Anthony, Smith D. Robert, Madey R. Gregory	*An Empirical Model for the Evaluation and Selection Expert System Shells*	Expert Systems with Applications vol.8, nr.1, p.143, Elsevier Science Ltd, Pergamon	1995
86.	26.	Tăndăreanu Nicolae	*Introducere în Programarea Logică. Limbajul PROLOG.*	Editura INTARF, Craiova	1994
87.	60.	Telek Istvan	*TURBO PROLOG*	Borland International Inc., Canada	1986
88.	61.	Tong C., Sriram D.	*Artificial Intelligence in Engineering Design*	Academic Press Inc.	1992
89.	49.	Tudor A., Prodan Gheorghe	*Durabilitatea şi Fiabilitatea Transmisiilor Mecanice*	Editura Tehnică, Bucureşti	1988
90.	10.	Tzafestas G. Spyros, Henk B.Verbruggen	*Artificial Intelligence in Industrial Decision Making, Control and Automation*	Kluwer Academic Publishers, The Netherlands	1995
91.	99.	Vasiu Lucian	*VISUAL BASIC 3.0*	Editura Tehnică	1996
92.	8.	Voiculescu P.	*"Sahul şi eficienţa gândirii" Revista "Automatică, Management, Calculatoare" Vol. 53, Pag.225*	Editura Tehnică, Bucureşti	1986
93.	18	Wilks Yorrick	*Grammar, Meaning and the Machine Analysis of Language*	Routledge & Kegan Paul	1972
94.	66.	Winston H. Patrick	*Inteligenţa Artificială*	Editura Tehnică, Bucureşti	1981
95.	20.	Yager R. Ronald, Zadeh A. Lotfi	*An Introduction to Fuzzy Logic Applications in Intelligent Systems*	Kluwer Academic Publishers, Boston	1993
96.	103.	Zhong G. & Dooner M.	*Use of Visualisation and Qualitative Reasoning in Configuring Mechanical Fasteners*	Arizona University	1997
97.	2.	* * *	*Petit Larousse*		1967
98.	3.	* * *	*Dicţionarul Explicativ al Limbii Române*	Editura Academiei, Bucureşti	1984
99.	22.	* * *	*Fundamente de Matematică ale Programării Logice în Inteligenţa Artificială (I)*	PC - Magazin, Nr.1 Martie	1990
100.	23.	* * *	*Fundamente de Matematică ale Programării Logice în Inteligenţa Artificială (II)*	PC - Magazin, Nr.2 Mai	1990
101.	24.	* * *	*Bazele Logice ale Inteligenţei Artificiale*	PC - Magazin, Nr.1	1991
102.	28.	* * *	*PDC - PROLOG, Reference Guide*	Copenhagen	1990
103.	29.	* * *	*PDC - PROLOG, Toolbox*	Copenhagen	1990
104.	30.	* * *	*PDC - PROLOG, Professional Users Guide*	Copenhagen	1990
105.	96.	* * *	*Verein Deutscher Ingenieure Riemengetriebe*	VDI 2758	1991
106.	105.	* * *	*CONTITECH, SYNCHRODRIVE Zahnriemen*		
107.	106.	* * *	*BLAURI, Schmalkeilriementriebe und Keilriementriebe, Flender*		
108.	107.	* * *	*OPTIBELT, Power Transmission*		

109.	108.	* * *	*HABASIT. Catalog.*
110.	109.	* * *	*WIPPERMANN. Catalog.*
111.	110.	* * *	*REXNORD. Rollenketten, Rotaryketten, Flyerketten.*
112.	111.	* * *	*GOODYEAR. Positive Drive Belts.*
113.	113.	* * *	*CONTI. Keilriemen.Continental*

1000. http://www-ksl.stanford.edu/htw/dme/thermal-kb-tour/frame-ontology.html#ARITY

1001. Industrial Maintenance Mechanic Competencies Outline
http://machinetool.tstc.edu/deliver/book07/immcout.htm - size 25K - 16 Apr 96

1002. Mechanical Software
http://www.pacificad.com/mechanic.htm - size 11K - 26 Feb 97

1003. MECHANICAL ENGINEERING
http://www.npiec.on.ca/~echoscan/mechanic.htm - size 3K - 11 Oct 96

1004. Anwendungsprojekt aus C++: Fuzzy Logic Expert System mit Verbal Expressions
http://force.wu-wien.ac.at/~j9252717/projekt/projekt.html - size 8K - 30 Jan 97

1005. Netweaver Knowledge Network Fuzzy Logic Expert System Development Tool
http://www.kgarden.com/netweave.htm - size 2K - 23 Jun 96

1006. Chen: A Fuzzy Logic-Based Expert System Development.
http://129.7.21.70/isso/a9596/chenrev.html - size 6K - 21 Jan 97

1007. Klaus Kolowratnik: Anwendungsprojekt aus C++ Fuzzy Logic Expert System.
http://force.wu-wien.ac.at/~j9252717/test.html - size 1K - 20 Jun 96

1008. RESEARCH AREAS CAD/CAPP/CAM
http://www.mcc.ac.uk/~xuxun/research_interest - size 385 bytes - 29 Sep 95

1009. IDS: Integrated Diagnostic System. in collaboration with Air Canada and GE Aircraft
http://ai.iit.nrc.ca/IR_public/project.html - size 2K - 29 Apr 96

1010. Fuzzy-Logik-Expertensystem
http://adonix.microelectronic.e-technik.th-darmstadt.de/isia/alex/Pubs/func_b_95.txt-18 Jun 96

1011. Julia Hodges, Susan Bridges, Shiyun Yie. Preliminary Results in the Use of Fuzzy Logic for a Radiological Expert System
http://www.cs.msstate.edu/PUBLICATIONS/TECHNICAL_REPORTS/960626abs.html - size 2K - 27 Feb 97

1012. Logic and Fuzzy Expert Systems
http://www.csa.runnet.ru/AI/ai_faqs.htm - size 3K - 28 Jun 96

1013. Togai InfraLogic's the World's Source for Fuzzy Logic Solutions.
http://www.ortech-engr.com/fuzzy/TilGen.html - size 6K - 26 Feb 97

1014. Togai InfraLogic -- TILShell Editors.The World's Source for Fuzzy Logic Solutions.
http://www.ortech-engr.com/fuzzy/editors.html - size 14K - 3 Mar 97

1015. Artificial Intelligence Subject Index
http://deeitis1.ai.iit.nrc.ca/subjects/Expert.html - size 2K - 20 Sep 96

1016. Togai InfraLogic -- TILShell 3.0 Debugging & Tuning
http://www.ortech-engr.com/fuzzy/debug.html - size 15K - 4 Mar 97

1017. Togai InfraLogic - Fuzzy Computational Accelerator
http://www.ortech-engr.com/fuzzy/fca.html - size 4K - 4 Mar 97

1018. Togai InfraLogic -- TILShell 3.0 Product List and Ordering Information.
http://ortech-engr.com/fuzzy/orderinf.html - size 6K - 4 Mar 97

1019. Adaptive Fuzzy Systems & Control: Essentials of Fuzzy Modeling & Control.
Fuzzy control systems. Fuzzy.
http://www.opampbooks.com/COM_FUZZ/ - size 3K - 5 Mar 97

1020. Togai InfraLogic -- VY86C500 FCA Core Library. The World's Source For Fuzzy Logic Solutions.
http://www.ortech-engr.com/fuzzy/fcacore.html - size 4K - 26 Feb 97

1021. Togai InfraLogic -- TILShell Objects. The World's Source for Fuzzy Logic Solutions.
http://www.ortech-engr.com/fuzzy/objects.html - size 14K - 26 Feb 97

1022. Togai InfraLogic -- TIL Background
http://www.ortech-engr.com/fuzzy/background.html - size 7K - 26 Feb 97

1023. Togai InfraLogic's TILShell Family
http://www.ortech-engr.com/fuzzy/TilShell.html - size 7K - 26 Feb 97

1024. What Is a Fuzzy Expert System? [5]
http://www.wior.uni-karlsruhe.de/Bibliothek/Software_for_OR/Fuzzy_Logic/FAQ.html - size 85K - 28 Aug 95

1025. Togai InfraLogic -- TILShell Menubars
http://www.ortech-engr.com/fuzzy/menubar.html - size 5K - 27 Feb 97

1026. Togai InfraLogic FuzzyCLIPS
http://www.ortech-engr.com/fuzzy/fzyclips.html - size 6K - 27 Feb 97

1027. EquipTypeXpert - A Rule Based Expert System Running Under Windows.
http://www.definitive.co.za/eqxp02.htm - size 2K - 21 Oct 96

1028. AI Guide: Certainty Factors. A Simplified Form of Probability Reasoning Devised by the Team
(Shortliffe et al) who built MYCIN.
http://www.mdx.ac.uk/www/ai/guide/43-cfs.htm - size 6K - 16 Jul 96

1029. Integrating Fuzzy Logic in an Expert System for Product Configuration.
Stephan Schwarze Institute of Industrial Engineering and Management (BWI) Swiss
http://www.bwi.bepr.ethz.ch/team_sch/staff/schwarze/fuzzy.html - size 35K - 20 Jul 95

1030. AI, CogSci and Robotics:Home Page | AI | Cognitive Science | Neural Science | Robotics | Conferences |
Journals&Publishers |
http://www-mice.cs.ucl.ac.uk/misc/ai/usenet.html - size 5K - 31 Jan 97

1031. Fuzzy Logic Overview
http://viking.cic.cau.ac.kr/fuzzy_sys/html/fuzzy6.html - size 8K - 18 Jun 95

1032. FAQ: Expert System Shells 1/1 [Monthly posting] - [1-6] Commercial Expert System
http://www.ioe.ac.uk/hgm/expert3.html - size 45K - 8 Jul 96

1033. Koryo Acupuncture Expert System (KAES II) SOFTWARE C
http://www.kaess.com/ - size 20K - 18 Feb 97

1034. Fuzzy Logic: Literature
http://htsa.htsa.hva.nl/~pylgroms/fuzlit.htm - size 6K - 20 Jan 97

1035. FAQ: Fuzzy Logic and Fuzzy Expert Systems 1/1 [Monthly posting]
http://www.dsclab.ece.ntua.gr/~kblekas/fuzzy_faq.html - size 84K - 19 Mar 96

1036. FuzzyTECH Apps Paper: Fuzzy Logic in Automotive Engineering
http://www.fuzzytech.com/e_a_esw.htm - size 40K - 12 Nov 96

1037. Fuzzy Logic Control of Oxygen Concentration - Yao Sun MD. Isaac Kohane
http://www.chip.org/chip/projects/avent/sun-fl-paper.html - size 21K - 11 Aug 95

1038. Fuzzy Logic and Neurofuzzy (Neuro-fuzzy) Research Groups
http://www-isis.ecs.soton.ac.uk/research/nfinfo/fzrgroup.html - size 14K - 17 Feb 97

1039. Knowledge Representation. Expert System. Logic. Planning. Symbolic Learning
http://cindy.cis.nctu.edu.tw/AI/Main.html - size 2K - 13 May 96

1040. Artificial Intelligence Subject Index
http://decitis1.ai.iit.nrc.ca/subjects/Expert.html - size 2K - 20 Sep 96

1041. The Poplog-Flex Environment. Prolog-Oriented Expert System.
http://www.cogs.susx.ac.uk/users/adrianh/poplog_flex.html - size 2K - 13 May 96

1042. DB&LP: Fumio Mizoguchi
http://joinus.comeng.chungnam.ac.kr/~dolphin/db/indices/a-tree/m/Mizoguchi:Fumio.html
size 3K - 28 Feb 97

1043. Prometheus - AI toolkit
http://www.intellect-net.com/authors/broghton/prometh.htm - size 2K - 10 Jul 96

1044. ITC366 Expert Systems (8)
http://www.csu.edu.au/handbook/subjects/ITC366.html - size 1K - 4 Mar 97

1045. KL-Magma Software Page.The Knowledge Representation Tools.
http://orasql.ilc.pi.cnr.it/KL-Magma.html - size 4K - 10 May 96

1046. Polytechnic University - CS665 Course Description
http://cis.poly.edu/courses/cs665.html - size 875 bytes - 18 Mar 96

1047. The Medtool Project - An Integrated Environment for the Development of Expert Systems
http://www.dc.fi.udc.es/ai/medint.html - size 1K - 21 Dec 95

1048. Artificial Intelligence
http://server.nich.edu/~cmps/courses/cmps418.html - size 4K - 19 Aug 94

1049. Meta Knowledge Representation for Knowledge Base Maintenance and Its Application.
http://www.csl.sony.co.jp/person/nagao/jsai/cdrom/papers/abstracts/3-3-1.html -
size 3K - 26 Feb 96

1050. An Expert System Shell for Aerospace Applications. IEEE Expert. August 1994. Vol. 9.
B E Prasad. T Siva Perraju. G Uma and P Umarani.
http://www.cs.wayne.edu/~tolety/publications.html - size 1K - 17 Oct 96

1051. CN2-based Induction for Updating Knowledge Rules in Medtool. Machine Learning can be applied
into the Expert Systems area with success.
http://finisterrae.dc.fi.udc.es/ai/~silvia/summary.html - size 3K - 11 Jul 96

1052. IEEE Transactions on Knowledge and Data Engineering. Volume 6
http://www.informatik.uni-trier.de/~ley/db/journals/tkde/tkde6.html - size 23K - 22 Nov 96

1053. Publicaciones de 1994. An Expert System for Identifying Steels and Cast Irons. J.L. Perez-de-la-Cruz.
M.J. Marti. R. Conejo. R. Morales Engineering
http://www.lcc.uma.es/TR/94.html - size 6K - 20 May 96

1054. Expert Systems. [OUTLINE - TUTORIAL - TOOLS - RESOURCES - SUBMIT]

http://www.usl.edu/~manaris/ai-education-repository/expert-systems-tools.html - size 16K - 25 Feb 97

1055. Research in the Software and Knowledge Engineering Laboratory
http://www.iit.nrcps.ariadne-t.gr/skel/research.html - size 6K - 13 Feb 96

1056. ADVANCED RESEARCH IN KNOWLEDGE-BASED SYSTEMS: INVENTING
THE NEXT GENERATION.
http://itri.loyola.edu/kb/c1_s5.htm - size 4K - 18 Nov 96

1057. Commercial Expert System Shells.
http://www.ue.eti.pg.gda.pl/~macieyg/faq/faq.ES_shells/odp007.html - size 40K - 18 Dec 95

1058. What is an Expert System? Early Artificial Intelligence Systems: * focused on search. * search trees. * heuristics. ie. tic-tac-toe.
http://www.cs.byu.edu/courses/cs575/lectures/lect1-96.html - size 6K - 18 Jan 96

1059. Examples of Expert System Interface Interactions
http://www.aic.nrl.navy.mil/papers/1991/AIC-91-029.text - size 23K - 8 Jun 92

1060. G. Riley, "Benchmarking Expert System Tools." Proceedings of ROBEX '86.
NASA/Johnson Space Center
http://www.civeng.carleton.ca/Courses/Grad/1994-95/82.562/clipsdoc/usrguide-Index-2.html-size 18K - 4 Apr 95

1061. CMPE 540 Principles of Artificial Intelligence. (Fall '96) Instructor: H. Levent AKIN Phone: + 90 (212) 263 15 00 Ext:1769...
http://dec002.cmpe.boun.edu.tr/~akin/cmpe540.html - size 5K - 5 Nov 96

1062. Constructive "all solutions" in pure Prolog In article. cdsm@doc.ic.ac.uk (Chris Moss) writes: > 1.
http://clement.info.umoncton.ca/pub/PrologDigest/89 - size 471K - 31 Mar 92

1063. Prolog Thousand Database This file is a preliminary version of a database of applications of Prolog and related languages, containing over 500 entries.
http://mufasa.ens.uabc.mx/cursos/computacion/pl/cs152/Other/prolog1000.v1 -
size 533K - 9 Oct 95

1064 Fuzzy Logic
http://www.cs.cmu.edu/Web/Groups/AI/html/faqs/top.html

1065. PTC Endeavor, Inc.--Power Transmission Home Page
http://www.powertransmission.com/prod/copage/ptc.htm - size 4K - 24 Feb 97

1066 Power Transmission Design
http://www.penton.com/corp/mags/ptd.html - size 5K - 19 Dec 96

1067. FLENDER POWER TRANSMISSION (PTY) LTD
3922434. E-Mail: flensa@pixie.co.za. Company Activity: DESIGN..
http://www.cbn.co.za/buslist/metal/30x888.htm - size 2K - 14 Feb 97

1068. Helical Power Transmission
http://www.heli-cal.com/HTML/Products/PT_Cell.htm - size 6K - 25 Jan 97

1069. Bearings & Power Transmission Products
http://www.ibtinc.com/automationpage.html - size 3K - 27 Feb 97

1070. Bearings & Power Transmission Products
http://www.ibtinc.com/conveyorspage.html - size 4K - 7 Mar 97

1071. Fenner Drives--Power Transmission
http://www.powertransmission.com/prod/copage/fenner.htm - size 4K - 24 Feb 97

1072. Knowledge Systems Research Group
http://fims-www.massey.ac.nz/cs/kbs.html - size 11K - 17 Dec 95
http://smis-asterix.massey.ac.nz/Research/kbs.html - size 11K - 20 Nov 96

1073. Mechanical Systems Engineering
http://www.eng.nagasaki-u.ac.jp/mech/MMechE.html - size 12K - 7 Mar 97

1074. Universal Technical Systems Inc.--Power Transmission Home Page
http://www.powertransmission.com/prod/copage/uts.htm - size 4K - 14 May 97

1075. Research Expertise at the NRCCE
http://www.nrcce.wvu.edu/About/Experts.Html - size 30K - 7 Dec 95

1076. ANNES '95 Abstracts Lagrangian Method for Satisfiability Problems of Propositional Calculus.
Masahiro Nagamatu and Torao Yanaru Kyusyu Institute of Technology E-mail:...
http://www.computer.org/conferen/proceed/annes95/abstract.htm - size 86K - 8 Jan 96

1077. HyperQ/Plastics: An Expert System for Plastic Material Selection
http://mml-mac-.stanford.edu/MMLWebDocs/research/papers/1991/beiter.asme.cie.91/
beiter.asme.cie.91.html - size 35K - 25 Mar 96

1078. Department Of Mechanical Engineering -- Research Area
http://www.me.concordia.ca/research/research_areas.html - size 24K - 15 May 95

1079. PUBLICATIONS. Books. Manufacturing Intelligence . Addison-Wesley. Boston. MA. 1988. pp. 352
(with David Bourne). Papers Published in Refereed
http://kingkong.me.berkeley.edu/~inouye/paul/pub.html - size 32K - 28 Mar 96

1080. ASIAN TECHNOLOGY INFORMATION PROGRAM (ATIP) REPORT: ATIP95.65: Human Computer
Interface International. 7/95 Yokohama
http://schooner.cs.arizona.edu:5605/japan/ATIP.reports/atip95.65 - size 52K - 9 Sep 95

1081. David C. Brown -- Publications Artificial Intelligence Research Group Computer
Science Department Worcester Polytechnic Institute Worcester. MA
http://cs.wpi.edu/~dcb/publications.html - size 34K - 21 May 97

1082. Intelligent Systems Research
http://fas.sfu.ca/ensc/research/groups/intelligent.html - size 39K - 21 Feb 97

1083. Accounting Agriculture American Artificail Intelligence Asia Assembly Programming Language
Biology BITNET Business C++ Programming Language C
http://necta.nec.com.tw/~chen/BIG5/comp/text/Internet-Tools/appendix.b -
size 51K - 24 Nov 95

1084. Abstract (2) Major: Mechanical Engineering Title: Development of Expert System for Assisting Process
Planning of Friction
http://www.fedu.uec.ac.jp/indonesia/data/Mechanical_Eng/data/suyato_12_2_96_2 -
size 2K - 12 Feb 96

1085. Analytical and Integrative Expert System Model for Design Project Management. Adedeji B. Badiru
School of Industrial Engineering and Vassilios E.
http://www.ecn.uoknor.edu/~nsfdesmm/nsf/nsfpaper/paper.html - size 62K - 7 Sep 95

1086. IEA/AIE-95 Conference Registration Kit. The Eighth International Conference on Industrial and
Engineering Applications of Artificial Intelligence and Expert Systems.
http://sigart.acm.org/Conferences/ieaaie-95.html - size 53K - 21 Mar 95

1087. IPMM Sessions IPMM'97. AUSTRALASIA-PACIFIC FORUM ON INTELLIGENT PROCESSING
AND MANUFACTURING OF MATERIALS.
http://mining.ubc.ca/ipmm97/ - size 158K - 15 Apr 97

1088. Abstracts Volume 2 - New Review of Applied Expert Systems
http://www.abdn.ac.uk/~acc025/abst2.htm - size 17K - 4 May 96

1089. CURRENT RESEARCH PROJECTS. Virginia Polytechnic Institute and State University. Mechanical Engineering Department. J. H. Břhn (703-231-3276)
http://www.me.vt.edu/ME/research/curres.html - size 17K - 11 Jan 97

1090. CAD/CAM, Korea Institute of Science and Technology. Myon-Woong Park. Principal Research Scientist. myon@kistmail.kist.re.kr. CAD/CAM, KIST.
http://chopin.kist.re.kr/psResearcher/mwpark.html - size 5K - 23 May 96

1091. SICStus Prolog - Prolog Objects
http://cactus.aist-nara.ac.jp/manual/sicstus3/sicstus_32.html - size 69K - 26 Feb 97

1092. IF/Prolog Commercial References in Japan
http://www.biz.isar.de/ifcomputer/FILES/ifprolog_commercial_reference_d.html - size 5K - 15 Mar 96

1093. Prolog.Glossary on Mon Dec 16 12:50:53 PST 1996
http://www.csci.csusb.edu/dick/samples/prolog.glossary.html - size 2K - 16 Dec 96

1094. SICStus Prolog - Debugging
http://www.compsci.bristol.ac.uk/~bowers/sicstus/sicstus_9.html - size 33K - 13 Dec 96

1095. SICStus Prolog User's Manual - Linda-Process Communication
http://www.sics.se/isl/sicstus/sicstus_28.html - size 8K - 5 Nov 96

1096. SICStus Prolog - Index of Built-Ins
http://www.cbil.upenn.edu/~niv/sicstus_manual/sicstus_14.html - size 28K - 9 Jan 95

1097. A Prolog-SQL Interface
http://www.cs.mu.oz.au/~ad/alp/net/sql.html - size 2K - 3 Jan 96

1098. Quintus Prolog Cross-development
http://www.aiil.co.uk/prolog/quintus/32pc/cross.htm - size 1K - 10 Dec 96

1099. SICStus Prolog - The Prolog Language
http://www.cling.gu.se/datorinfo/sicstus/sicstus2_1/sicstus/sicstus_10.html - size 49K - 11 Sep 96

1100. Notes on an Editorial Expert System (in Prolog) at TV Guide
http://www.als.com/nalp/appls/tv_guide.html - size 22K - 8 Mar 96

1101. Applications of Artificial Intelligence
http://boulmer.ncl.ac.uk/modules/1995-96/csc304/Coursework2.html - size 4K - 3 May 96

1102. A Prolog-Based Chinese Expert System. L. S. Hsu. Abstract.
http://cpol.csie.nctu.edu.tw/Abs/88187.html - size 1K - 31 Oct 96

1103. CS 575 - Fifty "Buzz" Words and Concepts for the Final. Advantages and disadvantages of expert system shells. Metrics used with expert systems.
http://www.cs.byu.edu/courses/cs575/buzzword.html - size 2K - 14 Jan 96

1104. Concrete Mix Design Knowledge-Based Expert System
http://www.designlab.ukans.edu/concreteDesgn.html - size 2K - 7 Jun 96

1105. R&M Design Expert System
http://erd.rl.af.mil/ER-News/21/expert.html - size 2K - 4 Aug 95

1106. FUZZY EXPERT SYSTEM for WWTS DESIGN
http://ev001.ev.nctu.edu.tw/~ctyang/ - size 2K - 9 Oct 96

1107. Computational Mathetics
http://www.cbl.leeds.ac.uk/~jas/cm.html - size 599K - 2 Apr 96

1108.
http://clement.info.umoncton.ca/pub/PrologDigest/92/dec92 - size 726K - 13 Jan 93

1109. CPN-AMI MIAMI 2.1.The Expert System MIAMI. version 2.1. Fawzia Derrough (1988-93). Amal El Fallah (1987-92), Samia Zerhouni (1992-), Gilles Corron (1994-95).
http://www-masi.ibp.fr/francais/recherches/sr/agl/cpn-ami1.miami21.html - size 7K - 9.Oct.96

1110. Integer and Combinatorial Programming.
http://bighurt.gsia.cmu.edu/Phd/OR_Topics.html - size 17K - 26.Sep.96

1111. Fuzzy Complex Knowledge Representation and Retrieval in Prolog
http://www.ceng.metu.edu.tr/~mturhan/ICSC95/konferan.html - size 3K - 30.Oct.96

1112. COMPUTER SCIENCE (Artificial Intelligence) UCAS Code: G800 BSc/CSAI.
http://cswww.essex.ac.uk/undergraduate/g800.htm - size 1K - 22.Jan.97

1113. Prolog Code Example. Knowledge Representation in MRS:
(defterm rotary-friction-damper friction-damper)
http://www.eit.com/presentations/nasa.phase1.review/review.slides.12.html - 25.May.97

1114. The Poplog-Flex Environment. Prolog-oriented Expert System Facilities Including: Frame-based Knowledge Representation, Combining Data Slots.
http://www.cogs.susx.ac.uk/users/adrianh/poplog_flex.html - size 2K - 13.May.96

1115. Databases and Artificial Intelligence 3
http://www.cec.hw.ac.uk/~alison/ai3notes/all.html - size 13K - 19.Aug.94

1116. REALITY, REPRESENTATIONS AND CYBERGRAPHY - Martin Sperka
http://www.inm.de/library/sperka_dec_95.html - size 29K - 1.Dec.95

1117. Knowledge Representation
http://www.is.saga-u.ac.jp/syllabus/KnowRep.html - size 3K - 6.Feb.97

1118. Math 480: Introduction to Fractal Geometry
http://homepage.seas.upenn.edu/~rajiyer/math480.html - size 10K - 15.Feb.97

1119. CSC 418 Artificial Intelligence (3) Prolog Programming
http://www.messiah.edu/acdept/depthome/mathsci/syllabi/csc418.htm - size 2K - 31.Oct.95

1120. Logic Programming and Constraint Satisfaction
http://www.cs.rug.nl/OZSL/basiccourse/logicprog.html - size 2K - 21.Feb.96

1121. Basic Concepts of Artificial Intelligence (AI).
http://phoenix.sas.muohio.edu/san/486.html - size 2K - 3.May.96

1122. Electrical and Computer Engineering 348 Introduction to Artificial Intelligence.
http://www.cs.uiuc.edu/CS_INFO_SERVER/DEPT_INFO/CS_PROGRAM_INFO/COURSES/348.html
size 3K - 20.Sep.96

1123. Expert Systems. [outline - tutorial - tools - resources - submit]
http://www.usl.edu/~manaris/ai-education-repository/expert-systems-tools.html - size 16K - 25.Feb.97

1124. Artificial Intelligence and Pattern Recognition.
http://nexttime.csfac.uwlax.edu/cs452.html - size 2K - 22.May.97

1125. Applications of Artificial Intelligence. Artificial Life.
http://sigart.acm.org/ai/subject.html - size 24K - 26.Jun.96

1126. The Fractal Store.
http://www.campusmall.com/fractal.html - size 7K - 2.May.97

1127. FuzzyCLIPS is an enhanced version of CLIPS developed at the National Research Council of Canada to allow the implementation of Fuzzy Expert Systems.
http://ai.iit.nrc.ca/fuzzy/fuzzy.html
http://ai.iit.nrc.ca/cgi-bin/fuzzy_log
http://ai.iit.nrc.ca/pub/fzclips/fixes.604/FUZZYCOM.C
http://www.ru.cs.nl/people/janw/CLIPS/documents.html
http://www.cs.ruu.nl/~janw/Clips6.0/documents.html

1128. KAREN KAFADAR. Education. 1975 B.S. Mathematics, Stanford University. 1975
http://www-math.cudenver.edu/~kk/kk-vita.html - size 25K - 21.May.97

1129. Engineering and Computer Science
http://www.usc.edu/Research/Science/seleng.html - size 38K - 8.Sep.95

1130. Ametric's® World Class Metric Power Transmission Products (BS. DIN. JIS. NF.UNI and ISO Standard) Balls. Belts. Bearing Sleeves. Bearing Units
http://www.ametric.com/products.html - size 15K - 30.Dec.96

1131. Power Transmission Design
http://www.penton.com/corp/mags/ptd.html - size 5K - 18.Apr.97

1132. Power Transmission Home Page--Power Transmission Industry Buyers Guide
http://www.powertransmission.com/ - size 4K - 5.May.97

1133. Power Transmission Professional Societies
http://www.powertransmission.com/links/prof.htm - size 9K - 20.Dec.96

1134. ETH Zürich - Power Transmission Group/Elektrische Energieübertragungss
http://www.eus.ee.ethz.ch/ - size 3K - 29.Apr.97

1135. Mitrpak Power Transmission Products
http://www.powertransmission.com/prod/copage/mitrpak.htm - size 3K - 27.Feb.97

1136. Power Transmission and Control - HANNOVER MESSE '97
http://pluto.messe.de:8000/hm97/prog/antriebfluid_e.html - size 8K - 15.May.97

1137. Overton Gear & Tool Corp
http://www.powertransmission.com/prod/copage/overton.htm - size 3K - 24.Feb.97

1138. Rexnord - Manufacturers of Power Transmission and Conveying Components
http://www.rexnord.com/ - size 4K - 15.May.97

1139. Power Transmission--Controls & Sensors
http://www.powertransmission.com/prod/controls.htm - size 10K - 6.May.97

1140. CS580 Introduction to Artificial Intelligence George Mason University DEPARTMENT OF COMPUTER SCIENCE. SPRING 1997. CS 580.
http://www.cs.gmu.edu/syllabus/syllabi-spring97/cs580-michalski.html - size 5K - 3-Feb-97

1141. INTEGRATING DESIGN AND COST PLANNING : A KNOWLEDGE BASED APPROACH.
Robin Drogemuller and George Najjar Department of Civil and Systems Engineering
http://lionfish.jcu.edu.au/~csrd/Publications/AIB_93/AIB_93.html - size 25K - 9-Apr-97

1142. Book{:1983, author = {Bundy, Alan}, title = {The Computer Modelling of Mathematical Reasoning
http://bib.informatik.uni-dortmund.de/literature/0154.bib - size 64K - 2-Apr-96

1143. Answers to Questions about Artificial Intelligence

http://longwood.cs.ucf.edu/~hull/ai_faq/ai-faq.04 - size 81K - 21-Aug-96

1144. Autor(1): Bundy, Alan Titel: The Computer Modelling of Mathematical Reasoning
http://bib.informatik.uni-dortmund.de/literature/0154.WRb - size 56K - 2-Apr-96

1145. B.A. ARTIFICIAL INTELLIGENCE (CSAI Handbook extract)
http://www.cogs.susx.ac.uk/degrees/BA-AI.html - size 9K - 17-Oct-95

1146. Computer Science at UNBC
http://cartan.unbc.edu/csprog.html - size 14K - 14-Aug-95

1147. Ishii, et al., HyperQ/Process: An Expert System for Manufacturing Process Selection.
http://mml-mac-9.stanford.edu/MMLWebDocs/research/papers/1991/ishii.aieng.91/ ishii.aieng.91.html - size38K - 25-Mar-96

1148. ARTNO 0017690-0. ARTICLE Artificial Intelligence. TEXT. Artificial Intelligence (AI)
http://geb.phys.washington.edu/local_web/p485/dzubay/ai1.html - size 10K - 21-May-97

1149. Akin.O. "How Do Architects Design?" in Artificial Intelligence and Pattern Recognition
http://205.130.63.7/htbook/ht13.html - size 34K - 17-Sep-96

1150. FAQ: Fuzzy Logic and Fuzzy Expert Systems
http://www.cis.ohio-state.edu/hypertext/faq/usenet/fuzzy-logic/part1/faq.html - size 86K - 28-Aug-97

1151. COMPUTER SCIENCE: Principles of Knowledge
http://www.cup.org/Titles/COMPUTER.html - size 27K - 21-Apr-97

1152. Introduction to Artificial Intelligence. Programming Languages: C++, Prolog, LISP.
http://www.ccm.itesm.mx/~marc/tcourses.html - size 16K - 21-May-97

1153. Reference-Artificial Intelligence/Agents Computer Books
http://www.opengroup.com/open/books/refai.html - size 61K - 23-Aug-97

1154. George Mason University DEPARTMENT OF COMPUTER SCIENCE
Introduction to Artificial Intelligence
http://www.cs.gmu.edu/syllabus/syllabi-spring95/cs580 - size 3K - 25-Jan-95

1155. Integrating the Building Procurement Process Using Knowledge Based Technology.
R M DROGEMULLER Senior
http://lionfish.jcu.edu.au/~csrd/Publications/IJCIT_94/IJCIT_94.html - size 29K - 9-Apr-97

1156. Semantics for Disjunctive Logic Programs
http://karna.cs.umd.edu:3264/papers/Raja89:phd/raj89.html - size 19K - 23-May-97

1157. Literatur zu Expertensysteme I (SS 1995) 1. Allgemeines. Buchanan, B. G., Shortliffe, E. H. (eds.)
Rule based Expert.
http://wwwwbs.cs.tu-berlin.de/fachgebiete/wbs/es1-95/lit1.html - size 14K - 22-May-95

1158. Introducción a la Inteligencia Artificial. Ingeniería Técnica de Informática de Sistemas Código de la asignatura: 40209.
http://www.dia.uned.es/~jgb/docencia/intro-ia/ - size 15K - 23-Apr-97

1159. Information Sources Sorted by Subject Area. Available Subject Areas. Agents.
Applications of Artificial Intelligence.
http://sigart.acm.org/ai/subject.html - size 24K - 26-Jun-96

1160. Artificial Intelligence. Contents. Quotes. Introduction. Definition. AI Applications in Business.
Business. Industry.
http://euro.net/innovation/Management_Base/Man_Guide_Rel_1.0B1/AI.html - size 16K - 19-Oct-94

1161. Introduction to Artificial Intelligence Professor: Tom Ellman
http://athos.rutgers.edu/~ellman/520/cs520.html - size 7K - 29-Apr-97

1162. ORA Canada Bibliography of Automated Deduction: C to E
Canada Bibliography of Automated Deduction cites over 3,000 reports and papers.
http://www.ora.on.ca/biblio/biblio-prover
http://www.ora.on.ca/biblio/biblio-prover-c-e.html - size 82K - 8-May-97

1163. MU Intelligent Machines Laboratory. I.ARTIFICIAL INTELLIGENCE
http://zansiii.millersv.edu/lectures/cs451.html - size 5K - 30-May-97

1164. Library of the Department of Information Systems Johannes Kepler University Linz.
http://www.ifs.uni-linz.ac.at/ifs/literature/ifslib.html - size 215K - 15-Apr-97

1165. Institute for Human & Machine Cognition. The University of West Florida
http://www.coginst.uwf.edu/~treichhe/consult/consult.html

1166. Robert C. Emmert. The Haley Enterprise, Inc. Info@Haley.CO
http://www.haley.com
http://home.haley.com/cgi-bin/CGI.exe/pdf/Releasetrialftp.pdf
http://home.haley.com/cgi-bin/CGI.exe/pdf/Q_And_A.pdf
http://home.haley.com/cgi-bin/CGI.exe/pdf/EclipseDataSheet.pdf
http://home.haley.com/cgi-bin/CGI.exe/pdf/EclipsePrice.pdf
http://home.haley.com/cgi-bin/CGI.exe/pdf/Rete++datasheet.pdf
http://home.haley.com/cgi-bin/CGI.exe/pdf/Rete++Price.pdf
http://home.haley.com/cgi-bin/CGI.exe/pdf/Reasoningaboutrete++.pdf
http://home.haley.com/cgi-bin/CGI.exe/pdf/TERdataSheet.pdf
http://home.haley.com/cgi-bin/CGI.exe/pdf/TERprice.pdf
http://home.haley.com/cgi-bin/CGI.exe/pdf/Cbrcomp.pdf

DICTIONAR

A.

Abtrieb partea antrenată/condusă.

Abtriebsmaschine maşină antrenată, maşină de lucru.

algoritm este o secvenţă detaliată de acţiuni realizată pentru îndeplinirea unei sarcini; din punct de vedere tehnic, un algoritm trebuie să găsească un rezultat, după un număr finit de paşi; termenul de algoritm este de asemenea folosit pentru a desemna orice secvenţă de acţiuni (care poate sau nu să fie terminată).

Antrieb acţionare, antrenare, transmisie.

Antriebsmaschine maşină de antrenare.

append a adăuga.

B.

backtracking este un algoritm folosit de limbajele de programare logică, cum este PROLOG, pentru a găsi toate căile posibile de a atinge un scop; este un algoritm care poate fi folosit pentr implementarea nedeterminismului; este efectiv, un mecanism de căutare în adâncime, în spaţiul problemei.

Berchnungsleistung putere de calcul.

Betriebsdauer je Tag durata zilnică de funcţionare.

Betriebsfaktor factor de regim.

buffer tampon; memorie intermediară.

C.

cadru este un obiect soft structurat, care conţine informaţia despre un obiect, din universul reprezentat; informaţia este reţinută într-o formă stereotipă.

clauză este o formulă logică, scrisă în formă normal conjunctivă (conţine numai operatori *şi*) şi care conţine numai cuantificatori universali (cuantificatorii universali se consideră impliciţi şi nu se scriu în formule).

close a închide.

compilare acţiunea efectuată de un program de calculator, numit program compilator, care transformă un text într-un program obiect (în cod maşină). Programul compilator, de obicei, intră în componenţa unui mediu integrat de dezvoltare.

concatenare legarea a două şiruri de caractere, pentru a forma un şir unic.

cuantificator universal este un operator logic, care indică faptul că, o propoziţie este adevărată pentru toate valorile din domeniu, luate de variabilele sale.

cuantificator existenţial este un operator logic, care indică faptul că, o propoziţie este adevărată numai pentru anumite valori din domeniu.

D.

database bază de date; bancă de informaţii/date (computerizate).

delete a şterge.

deny a nega; a refuza; a nu acorda.

device plan; intenţie; invenţie.

domain este o secţiune obligatorie dintr-un program, scris în limbaj PROLOG, în care utilizatorul defineşte tipul obiectelor şi domeniile nestandard.

E.

euristic este un algoritm, care reduce şi limitează căutarea în domeniu, grăbind găsirea soluţiei; nu garantează optimul, nici chiar realizabilul, pentru soluţiile găsite.

exit a ieşi.

expert specialist, profesionist care deţine o experienţă proprie într-un domeniu de activitate practică.

F.

fail a eşua; a da greş; a nu izbuti.

fapt este un tip de clauză, folosit în programarea logică, care nu are sub-goal-uri şi care este întotdeauna adevărat.

file domeniu fişier, fişier.

flow pattern cadru de reglementare a tipului, numărului şi poziţiei datelor transferate de la un modul de program la altul, de la o zonă de program la alta.

G.

goal este scopul unui program scris într-un limbaj logic, care are ca şi argument unul dintre predicatele programului, predicat ce este corelat cu toate clauzele.

ground instance deducţie de bază

H.

head cap, cap de listă.

I.

inferenţă este un proces logic, prin care sunt derivate fapte noi, pornind de la fapte cunoscute, prin aplicarea regulilor specifice.

integer întreg; (număr) întreg.

K.

keyboard claviatură, tastatură.

L.

limbaj declarativ constituie termenul general folosit pentru limbajul relaţional sau pentru limbajul funcţional; acesta este în opoziţie cu limbajul imperativ (procedural); descrie relaţiile dintre variabile în termenii funcţiilor sau în termenii regulilor de inferenţă; limbajul executor (interpretor sau compilator) aplică un algoritm fix acestor relaţii, cu scopul de a produce un rezultat.

limbaj imperativ este limbajul care specifică explicit secvenţele de paşi, ce trebuie parcurşi, pentru a produce un rezultat.

lower mai jos, funcţie de transformare.

M.

maşina de inferenţă modul de program, din componenţa unui sistem espert, care efectuează inferenţa (deducţia).

member membru.

O.

ontologie este un concept filosofic despre existenţă;
în Inteligenţa Artificială este o specificare teoretică explicită, despre cum se reprezintă obiectele şi alte entităţi, care sunt declarate că există în acelaşi domeniu de interes, precum şi sistemul de relaţii care le ţine împreună; pentru sistemele de Inteligenţă Artificială "existent" este ceea ce poate fi reprezentat; se poate descrie ontologia unui program, prin definirea unui set de termeni de reprezentare.

open a deschide, a deschide un fişier.

P.

printer imprimantă.

procedural ce se petrece după o procedură.

R.

read a citi.

rename a redenumi

reţea semantică este un mod de reprezentare al realităţii, sub formă de graf, folosind diverse tipuri de relaţii, între nodurile grafului.

S.

semantica înţeles

screen ecran.

shell coajă; găoace; carapace.

sistem Shell mediu de dezvoltare al Sistemului Expert, care nu conţine baza de cunoştinţe a utilizatorului.

sistem de producţii constă dintr-o colecţie de reguli, o memorie activă cu fapte şi un algoritm, numit *căutare înainte* şi care produce fapte noi.

string şir; şir de caractere.

T.

tail coadă; coadă de listă.

U.

unificare procedeu de deducţie logică.

universul discursului este un set de obiecte, care pot fi reprezentate într-un limbaj declarativ; conţine definiţii care asociază la numele de entităţi (clase, relaţii, funcţii) un text ce explică numele.

upper mai sus, funcţie de transformare.

W.

write a scrie.

CURRICULUM VITAE

Numele şi prenumele : DEHELEAN Nicolae Mircea născut în Arad la 25.09.1952
Starea civilă : căsătorit
Limbi străine : engleza, franceza (conversaţie tehnică, scris, citit, nivel foarte bine)
Profesia, ocupaţia actuală : inginer mecanic, şef de lucrări la Catedra "Organe de Maşini şi Mecanisme", Facultatea de Mecanică, Universitatea "Politehnica" din Timişoara

STUDII DE CALIFICARE

1971 - Bacalaureat, Liceul teoretic "Ion Slavici", Arad
1976 - Diploma de Inginer, Specializarea Tehnologia Construcţiilor de Maşini, Facultatea de Mecanică, Institutul Politehnic "Traian Vuia" Timişoara

EXPERIENTA

1977 - 1981 - inginer proiectant (S.D.V., Maşini Unelte şi Agregate) la Intreprinderea "Electrotimiş" Timişoara
1981 - 1982 - inginer proiectant (Utilaj Ind. Alimentară) la Intreprinderea I.M.A.I.A. Arad
1982 - 1990 - asistent universitar la disciplina Organe de Maşini, Catedra OMM, Facultatea de Mecanică, Institutul Politehnic "Traian Vuia" Timişoara
1990 - 1998 - şef de lucrări la Catedra OMM, Facultatea de Mecanică, Universitatea "Politehnica" din Timişoara

ACTIVITATEA DIDACTICA

Cursuri predate :
- Organe de Maşini şi Tehnologie (Facultatea de Construcţii)
- Aparate de Cinematografie şi Fotografie
- Elemente de Creativitate Tehnică (Inventică)
- Aparate Optoelectronice
- Baze de Date în Conducerea Proceselor
- Inteligenţa Artificială
- Sisteme "CIM"
- Optoelectronică Medicală
- Conducerea Inteligentă a Roboţilor Industriali (Vedere Artificială)
- Măsurători Asistate de Calculator (Computer Aided Quality)
- Proiectare Asistată de Calculator (Computer Aided Design)

ACTIVITATEA STIINTIFICA

Lucrări ştiinţifice publicate (între 1984 - 1998) : 41
Manuale, cursuri publicate : 1 (în curs de apariţie)
Contracte de cercetare ştiinţifică : 9 (din care la 2 titular)
Invenţii brevetate şi publicate : 10
Inovaţii înregistrate : 3

Timişoara, 15.06.1998

ş.l.ing. Nicolae Mircea DEHELEAN