UNIVERSITATEA POLITEHNICA BUCUREŞTI
Facultatea de AUTOMATICĂ ŞI CALCULATOARE

SISTEME INTELIGENTE HIBRIDE ÎN DIAGNOZA MEDICALĂ

Teză de doctorat

Conducător ştiinţific:
Prof. Dr. Ing. Ioan DUMITRACHE

Doctorand:
Asist.mat.inf. Sabina MUNTEANU

2006

Mamei mele

Mulțumiri

Aş dori să mulțumesc în primul rând Lui Dumnezeu, fără de Care nimic nu este posibil, şi familiei mele pentru sprijinul neprețuit pe care mi l-a acordat.

Adresez cele mai sincere mulțumiri domnului profesor Ioan Dumitrache, pentru susținerea, înaltul profesionalism şi răbdarea de care a dat dovadă în îndrumarea acestei lucrări, şi medicilor Lidia Drăgan şi Cristina Morar care mi-au fost de mare ajutor în realizarea aplicației medicale.

Le mulțumesc tuturor prietenilor şi celor care au avut încredere în mine.

Autoarea

Glosar

ALP Abductive Logic Programming: Programare Logică Abductivă;

ANFIS Adaptive Neural Fuzzy Inference System: Sistem de inferență fuzzy neural adaptiv;

(A)TMS (Assumption-based) Truth Maintenace System: Sistem de menținere a consistenței (bazat pe ipoteze);

BCP (Boolean Constraint Propagation) Propagarea Booleană a Constrângerilor;

(D)CSP (Dynamic) Constraint Satisfaction Problem: Problemă de Satisfacere a Constrângerilor;

KBANN Knowledge-Based Artificial Neural Network: Rețele neurale bazate pe cunoştințe;

NFC Neuro-Fuzzy Classifier: Clasificator neuro-fuzzy;

RBF (network) Radial Basis Function: Rețea cu funcții bază radiale;

SLD (NF) Single Linear Derivation (with Negation as Failure): Derivare singulară lineară.

Cuprins

I.Introducere

Diagnoza este unul dintre aspectele cele mai importante ale problemelor complexe ale lumii reale. Începând cu diagnoza tehnică, ce trebuie să asigure productivitatea şi siguranţa oamenilor în mediile în care omul interacţionează cu maşina, şi continuând cu diagnoza medicală pentru identificarea şi tratarea bolilor, avem o multitudine de situaţii ce relevă utilitatea problemei pe care dorim să o abordăm în cadrul acestei lucrări. Siguranţa centralelor nucleare, pilotarea de avioane în medii dificile, sistemele de producţie (şi lista poate continua pe câteva pagini) sunt de neconceput fără un sistem de monitorizare care să depisteze rapid şi precis dacă sistemul sub observaţie începe să se îndepărteze de comportamentul normal.

Ca subdomeniu al Inteligenţei Artificiale, diagnoza se ocupă cu dezvoltarea de tehnici şi algoritmi care să determine corectitudinea comportamentului unui sistem. Dacă funcţionarea este incorectă, algoritmul trebuie să identifice ce subcomponentă anume a sistemului este defectă, şi de ce tip de defect este afectată. Identificarea se bazează pe *observaţii* asupra comportamentului.

Există două abordări principale ale diagnozei: *diagnoza expertă* (sau pe baza unui sistem expert) şi *diagnoza pe bază de model*. Diagnoza expertă se bazează pe experienţa cu sistemul în cauză (furnizată de exemple gata clasificate sau direct de către un expert uman), de la care se construieşte o funcţie ce asociază defecte cu simptome. Abordarea suferă de mai multe dezavantaje: complexitatea şi dificultatea procesului de învăţare şi obţinere a cunoştinţelor, dimensiunea sistemului final, lipsa robusteţii. În diagnoza pe bază de model, se simulează sistemul pornind de la observaţiile curente, folosind o abstractizare (posibil incompletă) a sistemului (i.e. modelul sistemului), şi se compară observaţiile cu rezultatul simulării, pentru ca pe baza comparaţiei să se identifice cauza disfuncţionalităţii. În această situaţie ne confruntăm cu un exemplu clasic de raţionament *abductiv*. În termeni foarte generali, abducţia este un tip de raţionament (invalid logic în cazul general) de generare a unor ipoteze explicatorii plecând de la simptome observate.

Strâns înrudite cu noţiunea de diagnoză sunt conceptele de *diagnozabilitate* (nu orice sistem este suficient de accesibil observaţiilor şi de bine înţeles pentru a putea fi diagnosticat) şi de *explicaţie* (adică, în cazul de faţă, o demonstraţie a legăturii defect-simptome, bazată pe înţelegerea aprofundată a sistemului diagnosticat).

Formularea unei probleme de diagnoză. Un diagnostic este, informal, "cea mai plauzibilă combinaţie de ipoteze care explică toate datele prezente" (definiţia abducţiei din [Byl91]). Această definiţie are nevoie de trei elemente de bază: o noţiune de "explicaţie", un mecanism de generare a ipotezelor de diagnostic, şi un algoritm de discriminare pe bază de testare. O caracteristică a raţionamentului de diagnoză este că porneşte cu un set incomplet de date iniţiale, pe baza cărora elaborează tentativ direcţii de investigare, care sunt focalizate cu ajutorul unui plan de testare (completarea informaţiilor iniţiale până când ajung la un profil recognoscibil dintr-un set de prototipuri). Pe parcursul algoritmului, o parte din ipotezele generate pot fi infirmate de noile evidenţe, şi pot apare ipoteze noi. Definiţia dată mai sus face problema, în cazul

general, intratabilă (NP-dificilă) ([Byl91]), sursa principală a acestei intratabilități fiind interacțiunile semnificative care pot să apară între elementele unor ipoteze compuse (cum este cazul şi în diagnoza medicală). De exemplu, în medicină, prezența simultană a două boli poate să mascheze prezența unor simptome altfel manifeste şi semnificative pentru fiecare boală în parte şi să introducă simptome specifice doar apariției combinate; o altă situație dificil de formalizat este cea a diagnosticului diferențial, în care prezența unor anumite simptome/combinații de simptome într-un context dat poate să anuleze o ipoteză plauzibilă, discriminând între ea şi un diagnostic alternativ, şi invalidând raționamentul ipotetic dezvoltat până în punctul respectiv. Intratabilitatea problemei este depăşită cel mai adesea prin folosirea unor euristici. Acestea fie simplifică spațiul problemei căutând o soluție exactă pentru spațiul simplificat, fie se mulțumesc să ofere o soluție aproximativă pentru problema inițială, după cum se va vedea şi în cadrul lucrării de față.

În diagnoză, ca şi în rezolvarea tuturor problemelor lumii reale, dificultățile iau naştere din surse diferite. O primă limitare este dată de spațiul de căutare a soluțiilor, care este adesea foarte mare, făcând imposibilă căutarea exhaustivă. În al doilea rând, înainte de rezolvarea propriu-zisă trebuie construit modelul problemei (care este întotdeauna, într-o măsură mai mare sau mai mică, o simplificare a realității descrise). La acest punct, se alege între două compromisuri:

1. păstrarea modelului cât mai aproape de realitate şi aplicarea unei tehnici aproximative de rezolvare (cum sunt şi tehnicile inteligente, de care ne vom ocupa în Capitolele III-IV), având drept urmare găsirea unei soluții aproximative, sau

2. simplificarea modelului pentru a se încadra limitărilor necesare aplicării unor metode tradiționale de găsire a soluției exacte (algoritmi de optimizare, raționament în cadrul unei logici formale).

A treia problemă se naşte din cauza elementelor de zgomot ce intervin în funcția de evaluare a soluțiilor, sau din faptul că această funcție se poate modifica în timp.

În fine, o sursă suplimentară de dificultate de care dorim să ne apropiem în această lucrare, o constituie constrângerile. Acestea impun restricții asupra spațiului de căutare determinând restricționarea acestuia în regiuni fezabile (soluții posibile) şi regiuni interzise (ne vom ocupa în detaliu de constrângeri în Capitolele VI,VII).

1.1.Obiectivele şi conținutul lucrării

Capitolul II se ocupă de problematica diagnozei în medicină, cu prezentarea detaliată a dificultăților ce pot apare, şi cu sublinierea direcției în care calculatorul poate fi de un folos real experților umani. Caracteristicile evidențiate în acest capitol au stat la baza argumentării în favoarea sau împotriva anumitor tehnologii în cadrul capitolelor următoare.

Capitolul III îşi propune o clasificare şi o prezentare comparativă a diferitelor metode folosite până în prezent în aplicaţiile de diagnoză, cu accent pe diagnoza medicală (care este şi aplicaţia finală a lucrării). Sunt evidenţiate avantajele şi dezavantajele formalismelor folosite, cât şi eventualele posibilităţi de traducere a unei scheme de reprezentare în alta, împreună cu ordinul de complexitate al fiecărei abordări. Am selectat cu precădere, din multitudinea de sisteme şi abordări posibile, pe cele care au fost utilizate deja, sau am considerat noi că ar putea fi aplicate cu succes în diagnoza medicală. Am detaliat unele aspecte (reprezentare a cunoştinţelor, algoritmi) care să pună în valoare mai bine anumite apropieri sau diferenţe între abordări, şi să justifice opţiunea noastră finală. Organizarea materialului în Secţiunile 3.2-3.4. se face pe baza criteriului tipului de diagnoză: bazată pe asocieri superficiale clase-caracteristici sau pe un model cauzal de profunzime. În jurul acestui criteriu se grupează tehnicile prezentate: reţele neurale, sisteme de inferenţă fuzzy, reţele bayesiene şi posibiliste, diagnoză abductivă sau bazată pe consistenţă folosind logica formală.

În Capitolul IV, secţiunile 4.2.-4.3. abordează problema hibridizării în general, respectiv în diagnoza medicală în special, iar 4.4.1-4.4.2 prezintă un sistem hibrid combinativ de diagnoză medicală (CHECK) folosit ca termen de referinţă pentru sistemul original DiaMed introdus în 4.5. Obiectivele capitolului sunt atât încadrarea sistemului DiaMed în contextul formalismelor de diagnoză actuale, dar şi deschiderea de noi perspective asupra aplicabilităţii anumitor tehnici în diagnoza medicală.

Capitolul V prezintă o abordare originală a fazei de selectare a ipotezelor în cadrul unui proces de diagnoză, abordare ce defineşte fiecare diagnostic cu ajutorul unei funcţii de decizie fuzzy. Tehnica este comparată cu sistemele de inferenţă fuzzy, considerate drept cel mai apropiat termen de referinţă.

Capitolul VI cuprinde suportul teoretic al abordării originale din cadrul nivelului de discriminare a ipotezelor în DiaMed. După o prezentare a sistemelor de argumentare, ca formalism nou şi eficient de reprezentare adecvat pentru abducţie, introducem semantica argumentativă a admisibilităţii, folosită în continuare pentru o definire originală a diagnosticelor multiple şi o viziune deopotrivă eficientă şi naturală asupra păstrării consistenţei pe parcursul raţionamentului de diagnoză. Secţiunea face legatura între calculul semanticii admisibile şi satisfiabilitatea unei formule logice, pentru a cărei abordare există algoritmi specializaţi (ne-am oprit la cei de satisfacere a constrângerilor). Dintre aceştia am ales o versiune de backtracking dinamic pentru DCSP (Dynamic Constraint Satisfaction Problems) [VerSchi], preferând-o pentru eficienţa sa şi adecvarea la specificul diagnozei medicale, al cărei context se modifică dinamic prin testare. Această versiune de algoritm o vom adapta la integrarea în sistemul DiaMed.

Capitolul VII conţine o prezentare mai aprofundată a relaţiilor care există între sistemele de argumentare directă şi programarea logică, între abordările CSP (Constraint Satisfaction Problems) şi sistemele TMS (Truth Maintenance Systems), între argumente şi sistemele TMS, toate privite prin prisma abducţiei (adică tipul de raţionament pe care se bazează aplicaţia lucrării de faţă). Secţiunea este, astfel, o justificare în plus a opţiunii

noastre pentru anumite formalisme, dintre alternativele posibile, şi o pledoarie pentru puterea de reprezentare şi calcul a sistemelor de argumentare directă.

Capitolul VIII prezintă în detaliu algoritmul asociat cu nivelul de discriminare a ipotezelor în sistemul hibrid DiaMed. Algoritmul este o adaptare originală a algoritmului prezentat în Secţiunea V, în care dinamica constrângerilor este determinată de rezultatele testelor medicale. Câteva exemple relevante sunt prezentate în detaliu, cu evidenţierea facilităţilor oferite de program.

Capitolul IX adaptează abordarea DiaMed la diagnoza sistemelor dinamice (fără modificări semnificative), şi o compară cu un algoritm dezvoltat special pentru sisteme dinamice, algoritm care încearcă să depăşească problema spaţiului de reprezentare şi memorare tot printr-o metodă de ajustare dinamică a contextului problemei.

Capitolul X concluzionează expunerea, subliniind, o dată în plus, contribuţiile originale ale tezei şi posibilele extinderi şi optimizări.

II.Specificul problemelor de diagnoză medicală

"Nu putem pretinde unui domeniu mai multă exactitate decât cuprinde natura sa"
(Aristotel)

2.1.Introducere

Diagnoza medicală este procesul de identificare a unei boli pe baza simptomelor sale (gr.: dia = prin, gnosis= cunoaştere, deci cunoaştere mijlocită, intermediată: de exemplu, cunoaşterea bolii prin intermediul simptomelor), plecând de la anumite *criterii de diagnostic*, pentru a ajunge la tratamentul adecvat. Medicul elaborează o mulţime de ipoteze posibile, pe care o analizează cu ajutorul anamnezei (i.e. consultaţie directă) şi a testelor specializate (de imunologie, biochimie, hematologie, coprocultură, imagistică, electrocardiogramă, biopsie etc.). Istoria domeniului începe să devină cunoscută odată cu medicul Hippocrate, în Grecia antică, şi este încă dominată de teorii dezvoltate în anii 1900, de către William Osler. Acesta punea mai puţin accent pe individualitatea bolnavului (de fapt, pe individualizarea bolii la fiecare pacient), privindu-l ca pe un reprezentat al unei clase (de diagnostic). Succesorul lui Osler a fost Archibald Garrod, care a fost primul care a definit conceptul de predispoziţie genetică spre anumite boli, orientând astfel mai mult atenţia spre unicitatea fiecărui pacient.

Lumea contemporană încearcă să integreze computerele procesului de diagnoză. Vom arăta în continuare cât de folositor şi de adecvat poate fi acest proces, prin prisma dificultăţilor ce pot apare în diagnoza medicală.

Trebuie subliniat de la început ca diagnoza medicală nu poate fi realizată efectiv decât de către un medic, şi întotdeauna în practica medicală judecata clinică a expertului uman are precedenţă asupra rezultatului furnizat de un program, indiferent cât de bine documentat şi valid algoritmic. Sistemele automate de diagnoză, în general, estimează probabilităţile bolilor, pe baza logicii lor interne şi a răspunsurilor utilizatorului la anumite teste. Rolul calculatorului în acest domeniu critic, în care riscul şi responsabilitatea sunt foarte mari, poate fi definit de două componente:

- ajută medicul să îşi reamintească rapid stări patologice neobişnuite;
- sugerează ipoteze alternative şi îngustează spaţiul de căutare mai repede şi mai bine decât expertul uman.

Astfel, expertul uman este de fapt *asistat*, nu *înlocuit*. Acesta nici nu va putea fi vreodată înlocuit de maşină (spre liniştea noastră!) întrucât gândirea umană este mai presus de raţionamentul discursiv ce poate fi reprezentat algoritmic. Există un spaţiu al intuiţiilor profunde ce nu se poate formaliza matematic, aşa cum există în fizică o lume subcuantică inaccesibilă cercetării umane (şi deci cu atât mai puţin controlului uman). Putem însă profita de calităţile în care suntem depăşiţi de calculatoare: capacitatea de memorare şi viteza de procesare a informaţiilor, în ideea reducerii erorilor umane din medicină.

Întrucât domeniul abordat este o lume dinamică inaccesibilă de o complexitate încă departe de a fi înțeleasă de către noi, incertitudinea este inevitabilă, ca şi erorile pe care le determină ([RusNor02]). Aşa cum observa autorul în [Res88], fenomenele din patologia umană sunt complexe, neliniare, discontinue şi variabile, astfel că diagnosticul este cea mai dificilă problemă în practica medicală.

Ne propunem în continuare o prezentare succintă a erorilor posibile şi a cauzelor lor, prezentare ce ne va ajuta să înțelegem mai bine sursele de dificultate în diagnoza medicală, şi să înțelegem unde anume apare necesitatea folosirii unui program automat de asistență în vederea îmbunătățirii performanței umane.

2.2.Tipuri şi surse de erori în diagnoza medicală

O problemă ce poate apare în practică este diagnosticarea greşită (cu o frecvență ce variază între 8% şi 40 %. Cauzele erorilor pot proveni din trei direcții principale [conform www.wrongdiagnosis.com].

O primă cauză posibilă rezidă în *erorile pacientului*, şi include erori de comunicare în care unele simptome nu sunt raportate din jenă/frică sau fiind considerate nerelevante, sau sunt raportate simptome false cu un anumit interes.

A doua cauză cuprinde *erorile testelor de laborator*: testele sunt supuse unor limitări din cauza cărora pot obține un fals negativ (simptomul raportat absent în situații în care este prezent) sau un fals pozitiv (simptom raportat când de fapt nu este prezent); în plus, unele teste eşuează la pacienții care suferă de o boală suplimentară, într-un proces nedorit de *mascare a simptomelor* întâlnit când este prezentă o anumită combinație de boli.

În fine, partea unde sistemele expert pot ajuta sunt *erorile medicului*, întrucât aceste erori provin (dincolo de unele limitări umane cum ar fi oboseala etc.) din neluarea în considerare la timp a unor alternative de diagnostic mai rare, mai puțin cunoscute.

Tipurile de diagnostic greşit întâlnite în practică sunt diagnosticul complet greşit şi diagnosticul parțial greşit. În ultima categorie se încadrează mai multe situații. Spre exemplu, *diagnosticarea greşită a subtipului*: se diagnostichează, să presupunem, diabetul, dar încadrarea în unul din cele două subtipuri (tip I sau tip II) este greşită- ar trebui menționat aici că diferențierea subtipului corect este necesară numai atunci când există tratamente diferite pentru subtipuri diferite. O altă situație este cea în care *boala cauzatoare a fost ignorată*: boala diagnosticată a fost cauzată de o alta care a fost neglijată; spre exemplu, există persoane la care diabetul este o stare primară, dar există şi cazuri în care diabetul este o stare secundară, cauza primară fiind hemocromatoza (avem aici o generalizare a relației simptom boală: unele boli pot fi simptome în relație cu alte boli). La fel de uşor se pot *ignora complicațiile*. Deasemenea se pot *ignora cauzele medicamentoase*: unele stări patologice sunt numai efectul secundar al medicației folosite de pacient (cele mai afectate organe la medicamentele puternice sunt ficatul şi rinichii),

sau se *ignoră stările înrudite*: unele boli tind să apară grupat, chiar dacă nu se cauzează între ele; este vorba probabil de o cauză comună declanşatoare pentru toate, nu întotdeauna clară (informaţia medicală are încă multe lacune). Aici de obicei există pericolul ca doar una dintre ele să fie diagnosticată. În fine, unul dintre cele mai dificile cazuri este cel în care se *ignoră o boală nerelaţionată* (cel mai probabil din pricina mascării unor simptome).

Există boli mai dificil de diagnosticat decât altele. Astfel, unele boli sunt *supra-diagnosticate*, în timp ce altele sunt *sub-diagnosticate* (neglijate): spre exemplu, sindromul colonului iritabil este deseori diagnosticat în locul unor boli cum ar fi boala Chron sau boala Celiac, mai rare şi mai uşor neglijate. Un rol important în sub-diagnosticare îl au simptomele nespecifice, vagi, lipsa testelor definitive (care să tranşeze foarte clar în favoarea unui anumit diagnostic), lipsa unor cunoştinţe temeinice asupra bolii şi a mecanismelor sale, sau apariţia simptomelor abia în stadii foarte avansate ale bolii ("ucigaşii tăcuţi", cum ar fi glaucomul, hipotiroidismul, hipertensiunea, osteoporoza, cancerul de plămâni, atacul de cord etc.) Între bolile dificil de diagnosticat se deosebesc câteva grupe cunoscute: *bolile mintale, emoţionale şi de comportament* (aici dificultatea provine din lipsa unor criterii foarte clare şi universal valabile), *bolile digestive*: simptomele sunt vagi şi accesul în diferite puncte ale tractului digestiv este anevoios, *bolile cu simptome vagi* ca: stare de rău, oboseală (de exemplu: fibromialgia, lupusul, boala Lyme, scleroza multiplă etc.) şi *bolile rare* (cu care medicii nu sunt familiarizaţi).

Am menţionat mai sus ca sursă de dificultate a diagnosticului faptul că este totdeauna o problemă dinamică: acesta se construieşte treptat, pe măsura descoperirii şi prelucrării informaţiilor descoperite. Cauzele pentru care informaţiile nu duc automat la diagnostic ar fi [Res88]:

- Există puţine semne patognomonice (semne evidente care să sugereze în special o anumită boală), şi "diagnosticul rezultă din **modul în care se combină diferitele semne şi simptome** mai puţin patognomonice" [Rîm83] (discontinuitatea este o consecinţă a acestei situaţii; diferenţe mici în mulţimea de simptome pot duce la diferenţe semnificative de diagnostic- adică boli neînrudite);

- Majoritatea semnelor şi simptomelor pot apare în foarte multe boli (idem nota precedentă);

- Bolnavul nu expune spontan toate semnele, acestea fiind descoperite treptat de medic (dinamism);

- Una şi aceeaşi boală se prezintă individualizat la fiecare pacient, în funcţie de terenul întâlnit (variabilitate).

Prezentăm în continuare, mai în detaliu, câteva exemple clasificate de situaţii dificile.

Boli mascate. În această situație, manifestările clinice ale unor boli pot să fie atât de atipice încât să se confunde cu tabloul clinic al altor boli. Spre exemplu, în unele cazuri, infarctul miocardic poate evolua sub forma unor dispepsii (greață, balonare, diarie), din cauza dereglărilor circulatorii ce produc congestii asupra unor organe din cavitatea abdominală. De asemenea, ulcerul gastro-duodenal poate evolua sub forma unei pancreatite cronice, iar cancerul bronho-pulmonar sub masca unei pneumonii, sau a unei hipertiroidii, datorită secreției de hormon tireo-stimulant de către celulele canceroase. Unele forme de cancer bronho-pulmonar pot evolua ca sindroame nefrotice, iar cancerul pancreatic poate evolua sub masca unei litiaze biliare sau a unui diabet, şi acestea sunt doar câteva exemple. Soluția în acest caz stă în găsirea corectă a cauzei bolii.

Boli asociate. Este situația în care o boală apare însoțită de complicațiile sale. Multe din "măştile" menționate mai sus, sunt, de fapt, boli autentice, spre exemplu: astmul însoțit de consecința sa -emfizemul, retinopatia şi glomeruloscleroza ce complică diabetul, infecția urinară şi insuficiența renală ce pot apare în litiaza renală, sau pancreatita acută din cadrul litiazei biliare.

Boli concomitente. Acestea sunt boli simultane ce pot evolua independent, fără legături patogenice importante, şi situația este una dintre cele mai confuze.

2.3.Etape ale procesului de diagnoză

După cum s-a mai menționat, nu există reguli clare şi precise după care experții se ghidează, şi care să poată fi traduse uşor algoritmic: raționamentul depinde în cea mai mare măsură de experiența, cunoştințele şi intuiția expertului (de aceea se şi folosesc reguli vagi în sistemele expert).

Practica medicală este, formal, o succesiune de procese informațional-decizionale de tipul:

Informație→ Semn → Decizie

O **decizie** se defineşte ca "procesul de alegere a unei **alternative**, pe baza unor **informații**, prelucrate după anumite **criterii**" [Res88], acestea din urmă fiind date în domeniul medical de posibilitățile de asociere a simptomelor în diferite boli.

Simptomele sunt informații cu valoare simbolică: ele sunt "expresia unor modificări anatomopatologice sau fiziopatologice pe un anumit teren"[Res88]. Astfel, bolile pot fi privite drept mulțimi vagi, fiind formate din elemente ce nu le aparțin în exclusivitate.

Plecând de la un set de simptome inițiale, medicul încearcă, după clarificarea simptomelor, reducerea bolilor la un set restrâns de posibilități (i.e. **diagnosticul clinic**). Aceasta este o operație de clasificare în care medicul trebuie să completeze petele albe sau să ignore unele simptome particulare, -adică recunoaşterea unei forme puternic afectată de zgomot, în limbajul inteligenței artificiale. Rolul diagnosticului clinic este de

a indica medicului ce investigaţii paraclinice trebuie să efectueze, pentru decizia finală. Desigur, pot exista mai multe etape de testare. În abordarea noastră, vom apela la funcţii de decizie fuzzy pentru partea de diagnostic clinic.

Diagnosticul clinic se stabileşte plecând de la **diagnosticul de sindrom**. Sindromul este "un ansamblu coerent şi sistematizat de manifestări clinice, care sintetizează trăsăturile comune ale mai multor entităţi patologice" [Res88]. De la diagnosticul clinic trebuie trecut la **diagnosticul etiologic şi patogenic** (cauza şi mecanismul de inducere a bolii), care explică şi argumentează diagnosticul clinic, sugerând în acelaşi timp şi alternativa de tratament. Etapa următoare este **diagnosticul diferenţial**, în care se compară diagnosticul pozitiv cu alte boli asemănătoare posibile, cu scopul de a evita confuziile. Diagnosticul diferenţial (diferenţierea între alternative) stă dealtfel la baza întregului proces de diagnoză. În final se face şi **diagnosticul complicaţiilor**. Logica ce stă la baza acestui proces este o combinaţie între logica formală nemonotonă, şi logica internă a fenomenelor (ce modelează domeniul).

Ultima etapă în diagnosticare constă în **argumentarea diagnosticului**. După cum am observat deja, diagnosticul etiologic stă la baza explicaţiilor pentru decizia luată. În abordarea prezentă, am apelat la elemente de teoria argumentaţiei pentru a modela această parte.

Concluzia demersului nostru este că, oricâtă matematică, logică şi cibernetică am utiliza, patologia umană nu este atât de exactă încât să poată fi tradusă în formule logico-matematice (după cum observa şi autorul în [Res88]), astfel că nu vom obţine niciodată diagnosticul doar prin astfel de formule. După cum afirma Osler, *"medicina este o artă bazată pe multă ştiinţă*".

2.4.Concluzii şi contribuţii

Capitolul propune o imagine de ansamblu a procesului de diagnoză medicală. Principala sa contribuţie stă în faptul că, relevând o parte din dificultăţile acestei diagnoze, oferă o justificare teoretică a necesităţii şi adecvării abordărilor pe care le vom folosi pentru această problemă în cadrul lucrării de faţă.

III.Starea în domeniul diagnozei asistate de calculator

3.1.Introducere

Capitolul de față prezintă principalele posibile abordări ale unei probleme de diagnoză, tehnică sau medicală, cu exemple de aplicații ce folosesc efectiv formalismele descrise. Am pus în evidență avantajele şi dezavantajele diferitelor modele, felul în care se pot completa, cât şi eventualele echivalențe. Am adăugat pe parcursul expunerii, acolo unde a fost posibil, şi câte un exemplu / o sugestie concretă de utilizare a metodei respective în diagnoza medicală.

Ne-am propus să prezentăm sistemele importante (în special pentru diagnoza medicală) nu printr-o simplă descriere şi exemplificare, ci şi dezvăluind modul de lucru, formalismul din spatele sistemului, întrucât această abordare ne-a înlesnit argumentarea alegerii unui anumit formalism în favoarea altuia.

Organizarea materialului pe subsecțiuni ține cont de un principiu de bază, şi anume- dacă s-au folosit pentru raționament asocieri superficiale clase-caracteristici, sau un model cauzal detaliat al procesului diagnosticat. Tehnicile inteligente se grupează în prezentare în funcție de acest criteriu.

Secțiunea 3.2. îşi propune o detaliere a tehnicilor de diagnoză privită în termenii clasificării de forme prin metode statistice. Acestea au ca numitor comun construirea - explicită sau implicită- a unor funcții discriminante utilizate în ierarhizarea ipotezelor de diagnostic şi sunt în general eficiente întrucât folosesc mai ales cunoştințe de suprafață (asociative, gata compilate). Prin specificul lor, sunt potrivite pentru selectarea ipotezelor de diagnoză şi restrângerea spațiului problemei (partea de diagnostic pozitiv). În introducere prezentăm principiul clasic al clasificării bayesiene, şi apoi felul în care acest principiu este prezent în cadrul antrenării unui perceptron multistrat. Aplicabilitatea rețelelor neurale în diagnoza medicală este discutată cu ajutorul unui exemplu de problemă de discriminare între două boli ce se pot confunda (ciroza şi insuficiența cardiacă). Secțiunea discută şi inadecvarea algoritmilor genetici (ca tehnică inteligentă ce cuprinde în esență o căutare aleatoare) la specificul diagnozei medicale.

Secțiunea dedicată modelelor intermediare 3.3. (considerate intermediare întrucât, deşi teoretic au capacitatea de a reprezenta un model complet, practic folosesc şi metode statistice de focalizare a raționamentului pentru a fi mai eficiente computațional) începe cu o prezentare extinsă a modelului conexionist al rețelelor bayesiene bazat pe regula de clasificare a lui Bayes. Metodele din Secțiunea 3.3 aduc în plus față de cele din 3.1.2 posibilitatea de a integra cunoştințe structurate şi modele parțiale în baza de cunoştințe. După o prezentare a inferenței probabiliste în rețelele bayesiene (ilustrată de un algoritm liniar pentru rețelele poliarbore şi de versiuni de algoritmi euristici sau stocastici pentru rețelele multiplu conectate), este prezentat modelul rețelelor noisy-OR (ce pot exprima relații logice incerte –cu zgomot-), aplicate deja în diagnoză. Formalismele sunt exemplificate de sistemele de diagnoză HUGIN şi PATHFINDER. Rețelele cauzale sunt

introduse ca o specializare semantică a rețelelor bayesiene. În acest context este introdus sistemul clasic de diagnoză medicală CASNET, bazat pe o rețea cauzală, şi care a constituit un punct de plecare al sistemului original DiaMed prezentat în lucrarea de față (Secțiunea VII). Prezentarea detaliată a bazelor teoretice ale rețelelor bayesiene ne permite să înțelegem mai bine dezavantajele inferenței probabiliste din CASNET.

Dacă în inferența probabilistă avem de-a face în principiu cu incertitudine statistică, abordarea incertitudinilor non-statistice face obiectul sistemelor de inferență fuzzy (Secțiunea 3.3.2) şi măsurilor de posibilitate (Secțiunea 3.3.3). Acestea din urmă sunt o generalizare a mulțimilor fuzzy şi un model de probabilități subiective ce depăşeşte unele aspecte contra-intuitive ale probabilităților clasice, şi permit extinderea raționamentului incert cu date precise din rețelele bayesiene, la raționament incert cu date imprecise (în rețelele posibiliste). Din păcate, lipsa unei semantici clare a determinat o anumită abandonare a formalismului de către cercetările recente. Finalul Secțiunii 3.3. se ocupă de modelele simbolice ale sistemelor pe bază de reguli sau arborilor de decizie. Aici se încadrează şi sistemul clasic de diagnoză medicală MYCIN, care este prezentat în detaliu, cu avantajele şi dezavantajele sale.

Secțiunea 3.4 pune accent pe abordările logice specifice diagnozei pe bază de model, urmărind o expunere a metodelor abductive şi bazate pe consistență. Acestea ar fi interesante pentru partea de explicare a concluziilor, şi pentru eventualele situații conflictuale ce pot apare, de exemplu, în cadrul diagnosticului diferențial, când trebuie discriminat între boli ce au fost selectate datorită unor simptome comune, dar care nu pot fi prezente simultan la un pacient dat. Tot acest capitol introduce sumar formalismele diagnozei pe bază de model a sistemelor dinamice, cu accent pe diagnoza sistemelor cu evenimente discrete.

3.2. Diagnoza prin metode statistice

Metodele statistice aproximează inferența logică pe cunoştințe formale, prin asocieri superficiale cauze-efecte, furnizând soluții aproximative. În general sunt eficiente, dar au slabe capacități de explicare a concluziilor.

3.2.1. Probabilități şi clasificare bayesiană

Definiție. Ieşirea unui experiment al cărui rezultat depinde de factori aleatori (adică poate lua o valoare oarecare fără să putem preciza dinainte care anume), se numeşte *variabilă aleatoare*.

Definiție. Fie X o variabilă aleatoare ce reprezintă ieşirea unui experiment dat. Presupunem că experimentul are doar un număr finit de ieşiri posibile, şi fie Ω spațiul probelor (i.e. mulțimea de valori posibile pentru X). O *funcție de distribuție* pentru X este o funcție reală *m* cu domeniul Ω care satisface:

1. $m(\omega) \geq 0, \forall \omega \in \Omega$; (3.1)
2. $\sum_{\omega \in \Omega} m(\omega) = 1$. (3.2)

$\forall E \subseteq \Omega$, *probabilitatea* lui E este numărul P(E): $P(E) = \sum_{\omega \in E} m(\omega)$. (3.3)

Observație! O variabilă aleatoare este o funcție pentru care se cunosc probabilitățile de apariție ale fiecărei valori.

Definiție. Fie $\Omega=\{\omega_1,\ldots\ \omega_k,\ldots\ \omega_r\}$ spațiul probelor. Se defineşte *distribuția de probabilitate condiționată* de evenimentul $E \subseteq \Omega$:

$$m(\omega_k \mid E) = \frac{m(\omega_k)}{P(E)}, \quad (3.4)$$

iar pentru evenimentul general F:

$$P(F \mid E) = \sum_{F \cap E} m(\omega_k \mid E) = \sum_{F \cap E} \frac{m(\omega_k)}{P(E)} = \frac{P(F \cap E)}{P(E)}. \quad (3.5)$$

Clasificare Bayesiană. Regula lui Bayes este una dintre cele mai cunoscute şi mai clasice metode de clasificare. Dezavantajele sale principale sunt: necesitatea cunoaşterii apriorice a probabilităților (acestea pot fi însă estimate din date), şi restrictivitatea ipotezei de independență între clase.

Fie x un vector de caracteristici pe baza cărora se efectuează clasificarea, $x \in \Re^d$, şi $\omega_1,...,\omega_c$ clasele posibile (categoriile). Se consideră în plus $P(\omega_j)$ probabilitățile apriori ale claselor, şi $p(x \mid \omega_j)$ probabilitățile caracteristicilor din vectorul de evidențe condiționate de prezența unei anumite clase (ω_j). Atunci regula de *recondiționalizare în prezența evidențelor a lui Bayes* exprimă probabilitățile claselor în funcție de evidențe astfel:

$$P(\omega_j \mid x) = \frac{p(x \mid \omega_j) P(\omega_j)}{p(x)}, \quad (3.6)$$

unde p(x) reprezintă evidența şi se exprimă ca:

$$p(x) = \sum_{j=1}^{c} p(x \mid \omega_j) P(\omega_j).$$

Probabilitățile apriori ale claselor şi cele condiționale pot fi estimate din mulțimea de date (=*D*) (*învățare Bayesiană*), regula rescriindu-se:

$$P(\omega_j \mid x, D) = \frac{p(x \mid \omega_j, D)P(\omega_j \mid D)}{p(x \mid D)}, \quad (3.7)$$

unde:

$$p(x \mid D) = \sum_{j=1}^{c} p(x \mid \omega_j, D)P(\omega_j \mid D) .$$

Regula de decizie Bayesiană:

Se alege categoria ω_i astfel încât $P(\omega_i \mid x) > P(\omega_j \mid x), \forall j \neq i$.

S-a demonstrat că această regulă minimizează probabilitatea medie a erorii de clasificare [DudHarSto00].

3.2.2.Diagnoza folosind cunoştinţe asociative

3.2.2.1.Introducere

Diagnoza poate fi privită în termenii clasificării de forme (clasificarea este operaţia elementară care stă la baza diagnozei). Clasificarea prin metode metrice presupune construirea unei mulţimi de *funcţii de discriminare* [DudHarSto00]: $g_i(x)$, $i=1,\ldots,c$, iar clasificatorul atribuie vectorul de caracteristici x clasei ω_i dacă $g_i(x) > g_j(x)$ pentru toţi $j \neq i$. De obicei aceste funcţii sunt construite pe bază de probabilităţi condiţionale sau măsuri fuzzy- ţinând cont de incertitudinile ce apar frecvent într-o problemă de clasificare. Funcţiile de discriminare definesc regiunile de decizie (graniţele de decizie) asociate diferitelor categorii.

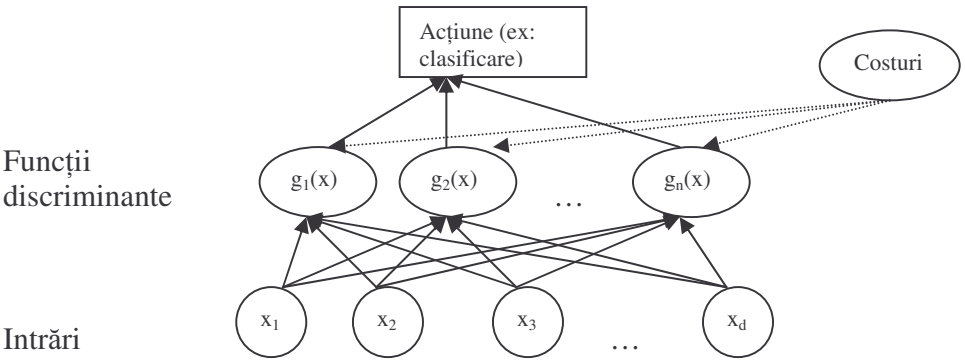


Figura 3.1. Schema generală a unei clasificări

Abordările statistice care urmează pot fi toate privite din acest punct de vedere.

Deşi diagnoza pe bază de model logico-simbolic (vezi Secţiunea 3.4.) poate trata probleme complexe, nelineare, variabile în timp, nu există încă o abordare unificată

satisfăcătoare. S-a ajuns astfel la *tehnicile inteligente*: acestea cuprind modele conexioniste structurate (rețelele bayesiene sau rețelele neurale fuzzy) sau nestructurate (rețelele neurale "clasice" = de tip cutie neagră), modele logico-simbolice pe bază de reguli, cum sunt sistemele expert (acestea, deşi în principiu pot conține reguli ce descriu comportamentul cauzal de profunzime, sunt de obicei folosite pentru a reprezenta euristici ce condensează expertiza umană în cunoştințe superficiale). Sistemele de inferență fuzzy sunt tot sisteme pe bază de reguli, dar au un mecanism de inferență propriu, specializat pentru lucrul cu informații imprecise şi incerte, ceea ce le conferă un statut aparte. Rețelele neurale sunt un exemplu tipic de raționament inductiv. Ele construiesc modele aproximative "ascunse" şi distribuite în ponderile rețelei, pornind de la mulțimi de exemple asociate cu domeniul de interes. Rețelele bayesiene, cauzale şi posibiliste sintetizează într-un model compact inferențe statistice.

Un avantaj în plus al tehnicilor inteligente stă în faptul că acestea şi-au dovedit deja atât limitările, avantajele, cât şi posibilitățile de traducere/comunicare de la un formalism la altul şi pot fi combinate în sisteme hibride în cadrul cărora cooperează pentru a îmbunătăți performanțele sistemului (câteva exemple vor fi prezentate spre sfârşitul secțiunii.)

Aşadar, tehnicile inteligente prezentate în continuare, deşi nu se bucură de claritatea şi expresivitatea diagnozei pe bază de model, pot trata eficient probleme complexe, nelineare, variabile în timp, unul dintre atuurile lor principale fiind faptul că lucrează, de obicei, cu aproximări ale domeniului (cunoştințe de suprafață), aproximări care simplifică mult inferența, dar în acelaşi timp fac necesare mecanismele de tratare a incertitudinii şi impreciziei.

3.2.2.2.Rețelele neurale şi clasificarea

Una dintre deficiențele cele mai importante ale modelelor structurate probabiliste şi posibiliste este dată de necesitatea cunoaşterii mulțimii de probabilități condiționale (obiective sau subiective). Acest dezavantaj este înlăturat în rețelele neurale, modele nestructurate ce realizează regresie neliniară (adică aproximarea unei funcții nelineare pe baza unei mulțimi de exemple- de fapt o mulțime de puncte ce aparțin graficului funcției), multe arhitecturi fiind recunoscute ca aproximatori universali. Dacă mulțimea de exemple este reprezentativă pentru domeniul ales, rețeaua are bune capacități de generalizare în spațiul problemei.

Unul din avantajele principale ale rețelelor neurale este capacitatea de învățare, ele ilustrând foarte bine inferența *inductivă.* Rețelele sintetizează modelul unui domeniu din exemplele de învățare, şi, odată cu modelul, construiesc implicit şi algoritmul asociat problemei considerate (fiind astfel utile în situațiile în care algoritmi specializați nu există). Aceste cunoştințe (model + algoritm) sunt reprezentate distribuit în ponderile numerice asociate conexiunilor sinaptice. Fiind o situație particulară de regresie neliniară, ele sunt, în plus, foarte tolerante la zgomotul din datele de intrare.

Rețelele folosesc, aşadar, cunoştinţe superficiale (de suprafață), pentru un raționament bazat pe asocieri (nu pe un model cauzal de profunzime). Prin urmare, prețul plătit pentru avantajele de mai sus stă în lipsa facilităților de explicare a rezultatului obținut (element critic pentru diagnoza medicală). Acest dezavantaj a fost parțial ameliorat de unele arhitecturi moderne de rețele neurale, dintre care câteva vor fi prezentate mai jos. Un alt dezavantaj al rețelelor neurale, privite prin prisma diagnozei medicale, este faptul că în cazul sistemelor complexe, ce necesită rețele neurale mari, apar dificultăți în antrenare datorate minimelor locale.

Pentru a îmbunătăți performanțele unui clasificator de forme inductiv este foarte importantă extragerea caracteristicilor celor mai relevante pentru definirea formelor sub observație. Pentru selectarea acestor caracteristici se pot folosi tehnici statistice tradiționale [Sta87], cum ar fi *analiza componentelor principale* (sau *dezvoltarea Karhunen-Loeve*: determină direcțiile care maximizează varianța norului de puncte-exemplu, pentru a spori separabilitatea între clase), sau *criteriile entropice* de extragere a caracteristicilor (reduc dimensionalitatea spațiului de caracteristici astfel încât pierderea de informație să fie minimă: se renunță la caracteristicile cele mai puțin informative). Cele două tehnici menționate optimizează separabilitatea, respectiv relevanța. În practică se doreşte o calitate satisfăcătoare pentru ambele criterii. Calitatea unui extractor de forme se poate evalua pe baza criteriilor de separabilitate (matrici de împrăştiere, divergența informațională –numerele Kullback-Leibler, distanța Bhattacharyya [Sta87]).

Perceptronul multistrat

Este una dintre cele mai cunoscute şi utilizate arhitecturi feedforward, şi foloseşte învățare supervizată de tipul corectare a erorii. Fiecare neuron din rețea are asociată o funcție de activare neliniară, diferențiabilă (de exemplu, funcția logistică). În afara stratului de intrare şi a celui de ieşire, rețeaua conține şi unul sau mai multe straturi ascunse de neuroni, care au rolul de a extrage caracteristici din ce în ce mai relevante ale vectorilor de intrare. În cadrul algoritmului de antrenare (backpropagation) se întâlnesc două tipuri de semnale: propagarea înainte a semnalelor calculate pe baza funcțiilor de activare a neuronilor, şi propagarea înapoi a semnalelor de eroare, pe baza cărora ponderile rețelei sunt ajustate folosind regula Delta (gradient descendent în spațiul ponderilor) [Hay98]. Complexitatea algoritmului este dată de o funcție ce depinde linear de numărul de ponderi.

Unul din aspectele interesante ale algoritmului backpropagation este faptul că acesta calculează de fapt probabilitățile condiționale aposteriori şi se poate aplica în final regula de decizie Bayes, rețeaua folosind pentru clasificare funcții discriminant implicit construite, şi constituind astfel un exemplu particular al aplicării teoriei deciziei în recunoaşterea formelor, după cum se vede în continuare.

Dacă avem o problemă de clasificare cu M clase în care uniunea celor M clase reprezintă spațiul total de intrare şi notăm intrarea $x_j \in \Re^m$ (al *j*-lea pattern de antrenare) şi ieşirea $y_j \in \Re^M$, rețeaua calculează o funcție vectorială $F : \Re^m \to \Re^M$:

$$y_j = [y_{1,j},...., y_{M,j}]^T = [F_1(x_j),....,F_M(x_j)]^T = F(x_j) \quad (3.8)$$

În continuare ne vom referi la următoarea problemă:

> *Care este regula optimă de decizie pentru clasificarea în cele M clase, după antrenarea perceptronului multistrat?*

În general, tot ce se ştie despre F, (dependentă de exemplele de antrenare <x_j , dj>) este faptul că este o funcție continuă ce minimizează funcționala riscului empiric [Hay98] :

$$R = \frac{1}{2N}\sum_{j=1}^{N} \| d_j - F(x_j) \|^2 \quad (3.9),$$

unde d_j este răspunsul dorit pentru x_j, iar N este numărul de exemple de antrenare.

Presupunând că antrenăm rețeaua cu valori țintă binare: $d_{kj} = \begin{cases} 1, x_j \in C_k \\ 0, x_j \notin C_k \end{cases}$, clasa C_k se va reprezenta ca un vector binar:

$$C_k = \begin{pmatrix} 0 \\ ... \\ 1 \\ ... \end{pmatrix} \text{ (cu 1 pe poziția k) (3.10)}$$

<u>Legea (slabă a) numerelor mari</u>. Fie $X_1,\ldots,X_n$ o secvență de variabile aleatoare independente identic repartizate, de medie $<X_i>=\mu$ şi varianță σ. Se defineşte variabila aleatoare:

$$X \equiv \frac{X_1 + ... + X_n}{n} . \quad (3.11)$$

Atunci, când $n \to \infty$, media <x>=μ.

$$< X > \equiv < \frac{X_1 + ... + X_n}{n} > = \frac{1}{n}(< X_1 > + ... + < X_n >) = \frac{n\mu}{n} = \mu \quad (3.12)$$

(Adică media valorilor variabilelor aleatoare converge la media μ a fiecărei variabile).

■

Din legea slabă a numerelor mari, dacă w este vectorul de ponderi al rețelei după antrenare (ce minimizează R), iar w^* este valoarea absolută ce minimizează media lui $\frac{1}{2}\| d - F(w,x) \|^2$ avem:

$$w \underset{N\to\infty}{\to} w^* \quad (3.13),$$

-adică un perceptron multistrat antrenat cu backpropagation pe o mulțime de exemple independente identic repartizate, calculează o aproximare a probabilităților aposteriori ale claselor:

$$F(w^*, x) = P(C_k \mid x), \text{ k=1,…,M. (3.14)}$$

În continuare se poate aplica regula de decizie a lui Bayes:

$$\text{Clasifică x drept } C_k \text{ dacă } F_k(x) > F_j(x),\ \forall j \neq x\ . \text{ (3.15)}$$

Observație. Clasificarea adaugă problemelor de modelare cu rețele neurale şi logică fuzzy cerința ca ieşirea să fie *discretă*.

Exemplu de clasificare cu ajutorul unui perceptron multistrat

Un dezavantaj al rețelelor neurale, privite prin prisma diagnozei medicale, este dat de discontinuitatea cunoştințelor medicale. Cazuri apropiate ca tablou al simptomelor pot face parte din clase diferite, dar este foarte probabil că o rețea neurală le va confunda. În plus, sistemele complexe, de dimensiuni mari sunt o problemă pentru o rețea neurală, din cauza minimelor locale. Am ales totuşi o pereche de boli (*ciroza* şi *insuficiența cardiacă*) şi am încercat să le clasificăm cu ajutorul unui perceptron multistrat. S-au putut urmări avantajele învățării din exemple, rapidității calculului paralel la partea de recunoaştere, şi robustеții la forme afectate de zgomot. Între alte dezavantaje, în afară de lipsa facilităților de explicare, s-au adăugat restricțiile prohibitive impuse de necesarul de memorie şi de timp de antrenare, ca şi lipsa datelor suficiente şi semnificative pentru a lucra cu toate cele 30 de boli considerate în aplicația finală.

Rețeaua are structura din Figura 3.2, cu 28 de intrări (cele 28 de simptome ale bolilor- a se vedea Anexa 3), 15 neuroni pe stratul intermediar şi două ieşiri (1 şi 0 pentru ciroză, 0 şi 1 pentru insuficiență). Stratul intermediar şi cel de ieşire folosesc drept funcție de transfer tangenta sigmoidă.

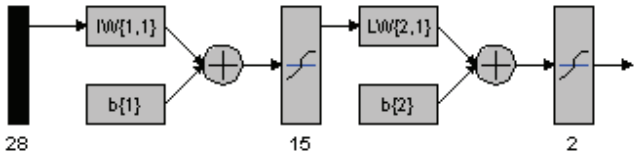


Figura 3.2. Structura rețelei

Funcția de performanță utilizată a fost media erorilor pătratice:

$$F = mse = \frac{1}{N}\sum_{i=1}^{N} e_i^2 = \frac{1}{N}\sum_{i=1}^{N} (t_i - a_i)^2 \quad (3.16)$$

(t=ieşirea dorită, a=ieşirea rețelei).

Am antrenat rețeaua cu algoritmul de optimizare numerică Levenberg-Marquardt, preferat ca fiind o metodă mai rapidă de antrenare, comparativ cu metodele de gradient descendent sau gradient descendent cu termen-moment. Performanțele după aproximativ 4000 de epoci de antrenare (batch) sunt redate de Figura 3.3.

Performanțele de recunoaştere în cadrul cirozei au fost de 100%, iar în cazul insuficienței cardiace de 76% recunoaşteri corecte, pe un set de exemple nefolosit la antrenare. Aceste rezultate sunt destul de aproape de situația reală, în care insuficiența cardiacă este uneori confundată cu ciroza, după cum s-a mai menționat.

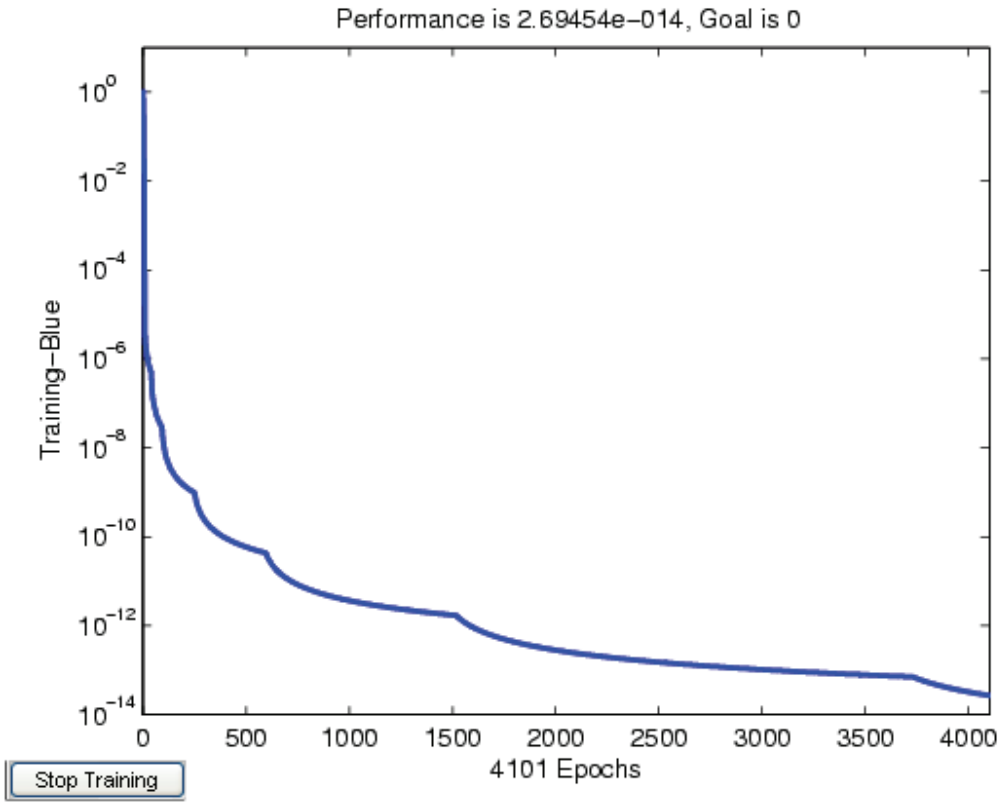


Figura 3.3.Performanțele rețelei după 4000 de epoci

Un avantaj al rețelelor neurale asupra abordării fuzzy este şi faptul că pot învăța ponderile din date, nefiind supuse astfel eventualelor erori ce pot proveni din setarea greşită a acestor ponderi.

Arhitecturi neurale cu auto-organizare

Arhitecturile neurale cu învățare nesupervizată sunt utile în problemele de clustering. Una dintre cele mai utilizate pentru performanțele sale este rețeaua cu auto-organizare Kohonen [Hay98]- folosită deja în multe aplicații de prelucrare a imaginilor (de exemplu, segmentarea imaginilor medicale în asistarea chirurgiei sau ca instrumente auxiliare în diagnoză, prin interpretarea datelor clinice de tip vizual.) Aceste arhitecturi lipsite de facilități de explicare nu sunt însă potrivite pentru procesul de diagnoză medicală în sine, care poate utiliza, eventual, doar rezultatul interpretării simbolice a imaginilor.

3.2.2.3.Algoritmii genetici şi diagnoza medicală

Algoritmii genetici ocupă un loc aparte în cadrul tehnicilor inteligente. Dacă tehnicile tradiționale de rezolvare algoritmică cum ar fi greedy, backtracking etc. (căutare deterministă), şi călirea simulată / căutarea tabu (căutare probabilistă) au în comun faptul că folosesc un singur punct al spațiului de căutare pentru explorarea de la fiecare iterație, algoritmii genetici sunt o tehnică inteligentă ce îmbunătățeşte aceste abordări prin faptul că procesează o populație de soluții simultan, introducând conceptul de *competiție* între soluțiile unei populații.

Noțiunea de "vecinătate" a doi indivizi-soluție din algoritmii genetici conduce la necesitatea definirii unor operatori de recombinare corecți relativ la semantica problemei, problemă dificilă în cele mai multe cazuri, şi sursă principală de limitări a acestei tehnici. Se deosebesc două tipuri principale de probleme, diferențiate pe criteriul adecvării (neadecvării) la folosirea algoritmilor genetici:

I. Probleme de căutare a unei soluții optime într-un spațiu continuu de soluții, spațiul soluțiilor posibile ocupând o pondere semnificativă din spațiul total al problemei (spațiu continuu de căutare).

Aici este adecvată şi utilă utilizarea algoritmilor genetici, după cum s-a demonstrat deja în cadrul a numeroase aplicații (a se vedea, spre exemplu, [MichFog04] şi [Mich96]). În acest caz, constrângerile problemei nu sunt complexe şi prohibitive pentru aplicarea acestor algoritmi, şi sunt utilizate în slujba optimizării prin aplicarea de penalizări şi funcții de reparație pe funcția de evaluare a soluțiilor a algoritmului genetic (folosirea de constrângeri pentru optimizare).

Alegerea a priori a unei metode evoluționare pentru o problemă de optimizare neliniară este încă o problemă deschisă. Caracteristicile care fac o problemă cu constrângeri dificilă pentru algoritmii genetici nu sunt clare, se ştie doar că ele depind de topologia

subspațiului fezabil (cu cât spațiul este mai puțin conex, problema este mai dificilă), şi de caracteristicile funcției de evaluare.

II. Probleme de găsire a unei soluții (nu neapărat optimă). Acestea sunt probleme în care constrângerile reduc semnificativ spațiul soluțiilor (spațiu discret de căutare, de exemplu, problema damelor pe tabla de şah).

De obicei se abordează problema CSP cu metode specifice de rezolvare, iar algoritmii genetici sunt folosiți pentru îmbunătățirea acestor metode (optimizare în slujba rezolvării constrângerilor). Aceşti algoritmi genetici speciali reţin câte constrâgeri sunt încălcate pentru a evalua calitatea soluțiilor [Par94].

Ca urmare a observațiilor precedente, am considerat greoaie şi neadecvată utilizarea algoritmilor genetici într-o problemă de diagnoză medicală, ce are întotdeauna un spațiu discret de căutare (fiind discontinuă), după cum observam şi în Capitolul 2. Dificultățile apar chiar din partea de modelare şi determinare de operatori adecvați, iar rezultatele sunt prea slabe şi costisitoare în raport cu specificul acestor probleme. În plus, complexitatea şi amploarea domeniului medical nu se potriveşte cu viteza redusă de procesare a algoritmilor genetici.

3.3. Modele intermediare de diagnoză

Structurile prezentate în această secțiune reprezintă o etapă intermediară între metodele asociative şi diagnoza pe bază de model. Le-am denumit "intermediare" întrucât, deşi au capacitatea de a reprezenta natural şi precis modelul profund al unui sistem, sunt de obicei folosite în modelarea cunoştințelor empirice, sau a modelelor parțiale, din motive de eficiență.

3.3.1.Modele probabiliste structurate. Rețele bayesiene

Rețelele bayesiene sunt un formalism ce are la bază ca metodă de calcul regula lui Bayes, şi ca reprezentare a cunoştințelor un model al domeniului redus la relațiile de influență directă (aceste influențe nu sunt neapărat cauzale; după cum se va argumenta mai jos, probabilitățile condiționale nu sunt cele mai potrivite în modelarea cauzalității.)

3.3.1.1.Inferență probabilistă în rețelele bayesiene

Dacă în semantica abordărilor logice valorile de adevăr se referă direct la starea lumii, probabilitățile sunt măsuri ale încrederilor/presupunerilor/ ipotezelor noastre relativ la lume. Aşadar, avantajul raționamentului probabilist asupra logicii este că un agent inteligent poate lua decizii chiar atunci când nu există suficiente informații care să demonstreze că acțiunea aleasă va da cu certitudine rezultatele dorite.

Inferența probabilistă reprezintă un mecanism de inferență pentru raționamentul aproximativ, care deşi este exponențial în cel mai rău caz, este eficient în multe situații practice**.** Ideea de bază este că distribuția de probabilitate produs a unui domeniu poate răspunde practic la orice întrebare legată de domeniul respectiv. Deşi abordarea devine intratabilă la un număr mare de variabile, în contextul aplicării regulii lui Bayes independența condițională dintre variabile simplifică mult calculul interogărilor şi reduce numărul de probabilități condiționale ce trebuie specificate.

Definiție[RusNor02]. O rețea bayesiană este o structură de date ce reprezintă dependențele între variabilele domeniului şi dă o descriere concisă a distribuției de probabilitate produs.

Nodurile rețelei reprezintă variabile aleatoare; nodurile rețelei X şi Y sunt legate printr-un arc ce porneşte din X dacă X are influență directă asupra lui Y în domeniul modelat. Fiecare nod are asociată o tabelă de probabilități condiționale ce cuantifică efectul părinților asupra nodului.

Topologia rețelei se poate interpreta ca o bază abstractă de cunoştințe ce reprezintă structura generală a proceselor de dependență din cadrul domeniului, rețeaua fiind definită complet de această topologie şi tabelele de probabilități condiționale.

Semantica rețelelor bayesiene poate fi privită din două puncte de vedere. Astfel, rețelele pot fi:

- o reprezentare a probabilității produs;
- o mulțime de aserțiuni de independență condițională.

Orice intrare din probabilitatea produs (şi deci orice interogare) se poate calcula din rețea după formula [RusNor02]:

$$P(x_1,...,x_n) = \prod_{i=1}^{n} P(x_i \mid Parents(x_i)) \quad (3.17)$$

dar în general:

$$P(x_1,...,x_n) = P(x_n \mid x_{n-1},...,x_1).P(x_{n-1},...,x_1) = P(x_n \mid x_{n-1},...,x_1).P(x_{n-1} \mid x_{n-2},...,x_1)...P(x_2 \mid x_1).P(x_1)$$

$$= \prod_{i=1}^{n} P(x_i \mid x_{i-1},...,x_1) \; ; \quad (3.18)$$

deci pentru a avea (3.17) este necesar:

$$\forall i : P(X_i \mid X_{i-1},...,X_1) = P(X_i \mid Parents(X_i)) \quad (3.19)$$

Această din urmă relație are loc dacă se numerotează cu un algoritm corespunzător nodurile rețelei: *Părinți(Xi)* $\in \{X_{i-1},...,X_1\}$, ideea fiind că orice nod este independent de predecesorii din rețea fiind dați părinții (depinde doar de părinți).

Rețelele bayesiene sunt *sisteme local structurate,* adică o variabilă interacționează cu un număr limitat de alte variabile, conducând astfel la algoritmi cu complexitate liniară. Reducerea de informație ce apare în practică în aceste rețele se datorează faptului că rețeaua surprinde foarte bine structura domeniului (şi domeniile lumii reale au, în general, o structură intrinsecă bine conturată.)

Pentru a ajunge la o structură optimă, în procesul de construire a rețelei, (vezi spre exemplu [RusNor02]) ordinea corectă de adăugare a nodurilor trebuie să țină cont de relațiile de cauzalitate din domeniu: de la cauze de bază, spre variabilele pe care acestea le influențează, până la frunze (care nu au nici o influență cauzală asupra altor variabile - aici "cauzalitatea" trebuie deocamdată înțeleasă în sensul mai restrâns de "influență" de un anumit tip).
O altă simplificare ce ajută mult calculul cu rețele bayesiene vine din faptul că tabelele de probabilități condiționale se încadrează de multe ori în distribuțiile canonice (spre exemplu, distribuția normală).

Relații de independență condițională în rețele bayesiene
Fiind dată o rețea, se poate citi din topologia acesteia dacă o mulțime X de noduri este independentă de o mulțime Y de noduri, fiind dată o mulțime E de noduri-evidență. Noțiunea folosită este cea de *d-separare*, şi anume: dacă X şi Y sunt d-separate de E, atunci X şi Y sunt independente fiind dat E. Această proprietate va folosi în algoritmii de interogare a rețelei.

Definiție. O mulțime de noduri E *d-separă* două mulțimi X şi Y de noduri dacă orice cale nedirecționată de la un nod în X la un nod în Y este *blocată* de E. O cale se numeşte blocată dacă există un nod Z pe ea care satisface una dintre condițiile următoare:

1. Z este în E şi în Z intră un arc de pe cale şi altul iese;
2. Z este în E şi din Z ambele arce ale căii sunt de ieşire;
3. Nici Z şi niciunul din descendenții săi nu aparțin lui E şi ambele arce ale căii intră în Z (vezi şi Figura 3.4).

Să considerăm o rețea bayesiană de tip poliarbore (adică există cel mult o cale nedirecționată între oricare două noduri). Presupunem că avem mulțimea de variabile-evidență E şi suntem interesați în valoarea variabilei aleatoare X, deci trebuie să calculam P(X|E). Notăm cu E_X^+ *suportul cauzal* pentru X (nodurile-evidență de deasupra lui X, conectate la X prin părinții săi), şi cu E_X^- *suportul evidențial* al lui X (variabilele-evidență de sub X, şi conectate la X prin copiii săi). Notăm cu U_i părinții lui X, cu Y_i copiii lui X, E_X toată evidența conectată la X, $E_{Ui\backslash X}$ evidența conectată la nodul U_i mai puțin calea ce trece prin X, $E^+_{Y_i\backslash X}$ evidența conectată la Y_i prin părinți, cu excepția lui X (Figura 3.5).

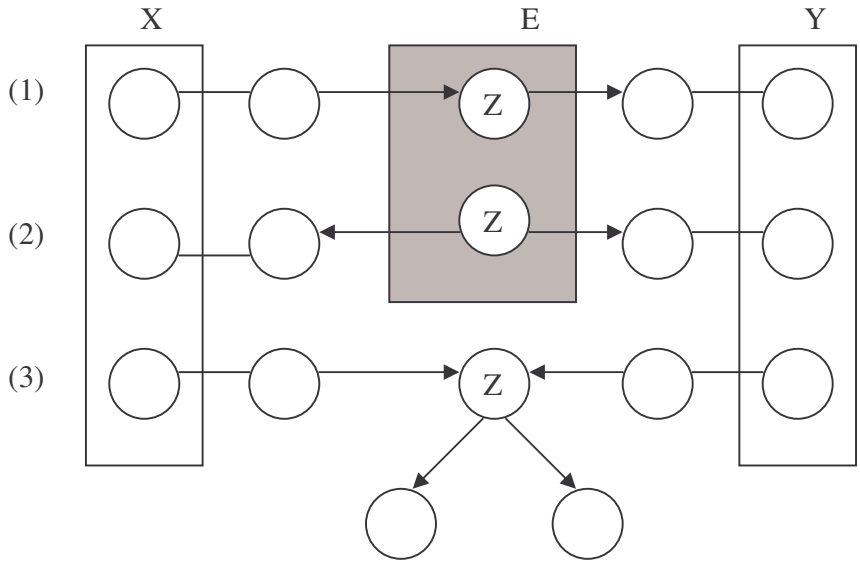


Figura 3.4.Definiția d-separării

Strategia generală de lucru va fi:

1)Exprimă P(X|E) în termenii contribuțiilor E_X^+ (suport evidențial), E_X^- (suport cauzal):

$$P(X|E) = P(X|E_X^+, E_X^-) \quad (3.20)$$

Folosind versiunea regulii Bayes condiționată de evidența de fond E:

$$P(Y \mid X, E) = \frac{P(X \mid Y, E)P(Y \mid E)}{P(X \mid E)}, \quad (3.21)$$

putem rescrie (3.20) ca:

$$P(X|E_X^+, E_X^-) = \frac{P(E_X^- \mid X, E_X^+)P(X \mid E_X^+)}{P(E_X^- \mid E_X^+)} \quad (3.22)$$

Cum X d-separă E_X^+ şi E_X^- în rețea , dacă notăm $\alpha = \frac{1}{P(E_X^- \mid E_X^+)}$, din (2) =>

$$P(X|E) = \alpha P(E_X^- | X) P(X|E_X^+). \quad (3.23)$$

2)Calculează efectul lui E_X^+ asupra părinților lui X şi apoi transmite efectul lui X (model recursiv).

După calcule în care se ține cont de proprietățile de d-separare din rețea şi de regula lui Bayes, se obține:

$$P(X \mid E_X^+) = \sum_u P(X \mid u) \prod_i P(U_i \mid E_{U_i \setminus X}), \quad (3.24)$$

adică se calculează contribuția evidenței pentru fiecare părinte U_i al lui X (=$E_{Ui\setminus X}$).

Ținând cont că porțiunile din rețea corespunzătoare părinților U_i sunt d-separate, deci independente, rezultă că putem folosi produsul pentru a obține probabilitatea unei anumite configurații complete u=(u_i) a părinților lui X. Această configurație se ponderează cu P(X|u) luat din tabelele de probabilități condiționale, şi în final se însumează după toate valorile posibile u ale lui U.

3. E_X^-=? Calculează efectul evidenței asupra copiilor lui X şi apoi transmite efectul asupra lui X (vom avea deci, din nou, apel recursiv).

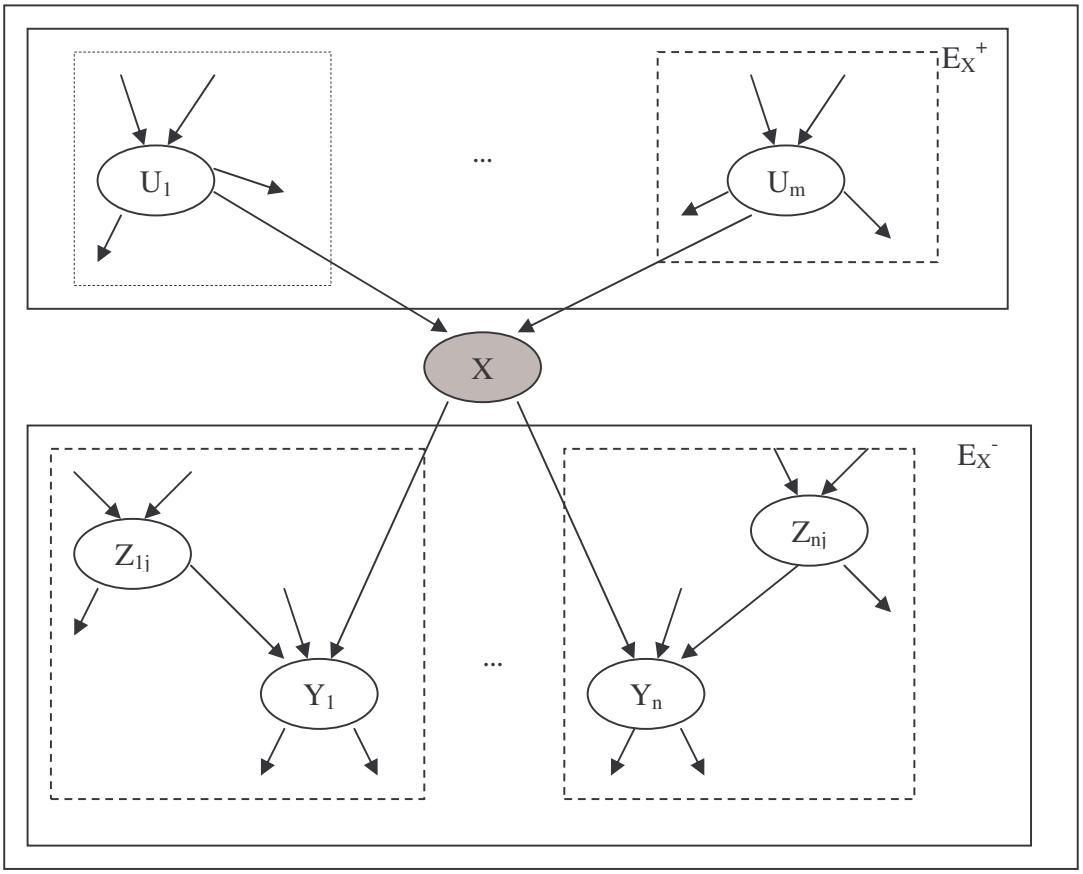


Figura 3.5.Rețea generică singular conectată. Partiționarea este facută în funcție de părinții şi copiii nodului X.

Expresia finală este:

$$P(E_X^- \mid X) = \beta \prod_i \sum_{y_i} P(E_{Y_i}^- \mid y_i) \sum_{z_i} P(y_i \mid X, z_i) \prod_j P(z_{ij} \mid E_{Z_{ij} \setminus Y_i}) \text{ . (3.25)}$$

adică se calculează contribuția evidenței asupra probabilității părinților Z_{ij} ai lui Y_i diferiți de X, produsul acestor contribuții după j dă contribuția evidenței pentru Y_i pentru o anumită configurație z_j a părinților, această contribuție totală se ponderează cu probabilitatea lui y_i condiționat de (z_i , X). Se ia apoi suma după toate configurațiile posibile z_i ale părinților şi această sumă se ponderează cu suportul dat de evidența copiilor lui Yi pentru valoarea y_i . În final se însumează după toate valorile y_i ale lui Y_i şi produsul este posibil datorită d-separării copiilor Y_i ai lui X în rețea.

Recursia se opreşte în nodurile evidență, nodurile rădăcină şi nodurile frunză, fiecare apel recursiv excluzând nodul din care a fost apelat, astfel că nu se trece de două ori prin acelaşi nod. Algoritmul este aşadar liniar în numărul de noduri al rețelei. Corectitudinea sa este restrânsă la clasa rețelelor de tip poliarbore.

Algoritm

function BELIEF-NET-ASK(X) **returns** o distribuție de probabilitate a valorilor lui X;
inputs: X o variabilă aleatoare;
SUPPORT_EXCEPT(X,null);

function SUPPORT-EXCEPT(X,V) **returns** $P(X|E_{x \setminus v})$
if EVIDENCE?(X) **then return** distribuția punctuală observată pentru X
else
calculează ($P_{X \setminus V}^- \mid X$)= EVIDENCE-EXCEPT(X,V)
U← PĂRINȚI [X]
if U vid
then return $\alpha P(E_{X \setminus V}^- \mid X) P(X)$
else
pentru fiecare U_i din U
calculează şi reține $P(U_i \mid E_{U_i \setminus X})$ =SUPPORT-EXCEPT(U_i , X)
return $\alpha P(E_{X \setminus V}^- \mid X) \sum_u P(X \mid U) \prod_i P(U_i \mid E_{U_i \setminus X})$

function EVIDENCE-EXCEPT(X,V) **returns** $P(E_{X \setminus V}^- \mid X)$
Y←COPII[X]-V
if Y vid
then return o distribuție uniformă
else
pentru fiecare Y_i din Y
calculează $P(E_{Y_i}^- \mid y_i) = EVIDENCE - EXCEPT(Y_i, null)$

Z_i ←PĂRINȚI[Y_i]-X
pentru fiecare Z_{ij} din Z_i
calculează $P(Z_{ij} \mid E_{Z_{ij}\setminus Y_i})$ =SUPPORT-EXCEPT(Z_{ij} , Y_i)

return $\beta \prod_i \sum_{y_i} P(E_{Y_i}^- \mid y_i) \sum_{z_i} P(y_i \mid X, z_i) \prod_j P(z_{ij} \mid E_{Z_{ij}\setminus Y_i})$.

■

Deşi, după cum s-a văzut, există algoritmi liniari pentru rețelele de tip poliarbore, pentru rețelele multiplu conectate inferența probabilistică exactă ramâne NP-dificilă. Pentru astfel de rețele generale se încearcă reducerea lor la poliarbori, aplicarea algoritmului de bază pentru fiecare poliarbore şi combinarea rezultatelor, conform unei strategii de tip "divide et impera". Algoritmii generalizați sunt în principal de patru tipuri: metode de clustering, metode pe bază de mulțimi tăietură, algoritmii de simulare stocastică, şi metodele de căutare pentru rețele de tip "noisy-or".

1.Metode de clustering: [HuaDar96] Se transformă rețeaua într-un poliarbore prin combinarea mai multor noduri într-un meganod.

Problema rămâne însă NP-dificilă întrucât creşte dimensiunea tabelei de probabilități condiționate (care se calculează pe produsul cartezian al domeniului variabilelor). Cheia acestei metode, care este totuşi cea mai eficientă, stă în alegerea meganodurilor potrivite.

2.Condiționalizare pe baza unei mulțimi-tăietură (cutset conditioning)[RusNor02]
Algoritmii de acest tip transformă rețeaua în mai multe rețele tip poliarbore, fiecare rețea simplă având una sau mai multe variabile deja instanțiate. P(X|E) este în final o medie ponderată a valorilor calculate de fiecare poliarbore.
(Mulțime tăietură = mulțime de variabile ce pot fi instanțiate pentru a rezulta poliarbori).

3.Metode de simulare stocastică[PraDag96, Russell & al. 2002]
Cele mai cunoscute sunt scalarea logică ("logic sampling") şi ponderarea verosimilității ("likelihood weighting").

a)Scalarea logică
Se aleg aleator valori pentru variabilele rădăcină şi se generează valori pentru variabilele dependente ținând cont de tabelele de probabilități condiționale; apoi se numără frecvența de apariție a evenimentului de interes:

$$P(X \mid E) = \frac{\mid X = t; E = t \mid}{\mid X = f; E = t \mid}. \quad (3.26)$$

Problema acestor metode este că, dacă suntem interesați de valori rare ale lui E, acestea vor apare rar la simulare. O rezolvare este dată de abordarea următoare.

b)Ponderarea verosimilității
Variabilele evidență sunt asumate adevărate, dar fiecare simulare este ponderată cu probabilitatea valorii de adevăr "A" pentru evidența condiționată de valorile simulate

pentru părinții acestei evidențe.Un exemplu de rețea bayesiană care foloseşte acest algoritm cu aplicații în diagnoză în medicina internă este proiectul **CPSC** (Malcolm Pradhan).

Din păcate, aproximările stocastice ale inferenței probabiliste sunt deasemenea NP-dificile [HuaHen96]. Pentru un anumit tip de rețele bayesiene (noisy-OR) însă, lucrurile se simplifică foarte mult, după cum vom vedea în continuare.

3.3.1.2.Rețelele noisy-OR

Pentru nodurile deterministe ale rețelelor bayesiene clasice, probabilitățile valorilor sunt specificate exact de valorile părinților, fără incertitudine. În schimb, în rețelele de tip "noisy OR" sunt modelate relații logice cu zgomot (ce adaugă incertitudine la abordarea logică).

Ex. Febră=Răceală sau Gripă sau Malarie (în logică: A(devărat) = A sau A sau A)

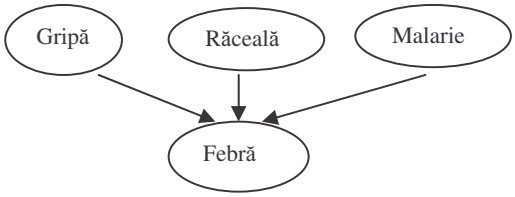


Figura 3.6.Exemplu de mini-rețea noisy-OR

Presupunem că:

- toate cauzele sunt listate ;
- fiecare cauză are o şansă independentă de a produce efectul;
- orice previne Gripa de a produce Febră e independent de ceea ce previne Malaria de a produce Febră.

Parametrii zgomot sunt priviți ca posibili inhibitori ai dependențelor modelate în probabilitățile condiționale; astfel, dacă P(Febră|Răceală)=0.4=> P(inhibitori)=1-0.4=0.6.

Regula de calcul: probabilitatea ca ieşirea să fie falsă este produsul parametrilor zgomot pentru părinții cu valoarea "adevărat". Prin urmare, nu mai avem tabele de probabilități condiționale ci maximum k parametri (unde k=numărul de părinți). Este o modelare diferită a ideii de "undercutting defeaters" [Poll92, Poll02]: o implicație poate avea excepții de utilizare.

4.Metode de inferență pe bază de căutare Ideea acestei metode este de a căuta instanțieri parțiale sau complete ale spațiului produs aposteriori, folosite apoi pentru estimarea valorilor variabilelor de interes. Succesul metodelor de căutare depinde de doi factori [HuaHen96]:

- în distribuția produs, masa majoritară a probabilității este concentrată într-o mică fracție a variabilelor ipoteze (după cum este şi cazul în diagnoză) ;
- există reguli eficiente de reducere masivă a spațiului de căutare.

Proprietatea 2 depinde de clasa de rețele bayesiene. Astfel, Max Henrion a dezvoltat un algoritm simplu (- TopN) pentru inferență în rețele noisy-OR cu două nivele [HuaHen96]. De exemplu, pentru rețeaua din Figura 3.7, algoritmul găseşte cele mai probabile N instanțieri ale nivelului boli.

O generalizare a algoritmului TopN pentru rețele noisy-OR multinivel este dată de către algoritmul TopEpsilon prezentat în continuare [HuaHen96].

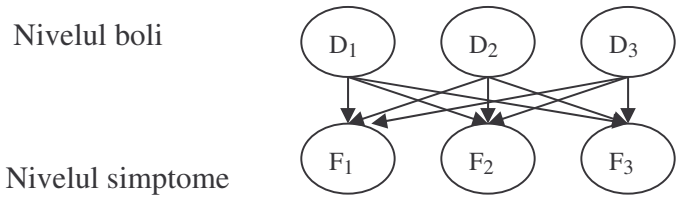


Figura 3.7.Rețea de diagnoză cu două nivele

Algoritmul TopEpsilon

TopEpsilon(ε_{target} , rețea bayesiană)
EticheteazăNoduri (rețea bayesiană)
nivel_max := nivelul nodurilor-evidență celor mai adânci //mai aproape de frunze

pentru fiecare nod al rețelei bayesiene
scor(nod):=0;

masa_acumulată:=0;
stiva:=vidă;
starea_curentă= instanțierea parțială a rețelei corespunzătoare nodurilor evidență de la nivel_max;

PUSH starea_curentă (în stivă);

WHILE stiva nevidă DO
BEGIN
starea_curentă=POP(stiva);
IF completă(starea_curentă) AND probabilitate(starea_curenta)>= ε_{target}
THEN
masa_acumulată=masa_acumulată+probabilitate(starea_curentă);
ELSE
$\varepsilon_{new} = \varepsilon_{target}$ / probabilitate (starea_curentă);
lista_extensii=EpsilonML(starea_curentă, ε_{new});
PUSH(lista_extensii, stiva);

```
                    END ELSE
                END WHILE

            FOR  fiecare nod din reţea DO
                    estimare_aposteriori(nod)=scor(nod) / masa_acumulată
```

În algoritmul de mai sus, EpsilonML reprezintă o procedură care găseşte instanţieri de verosimilitate sporită pentru părinţii direcţi (şi fără legături colaterale între ei) ai unei mulţimi-evidenţă. Probabilitate(starea curentă) determină verosimilitatea instanţierii parţiale construite până la momentul respectiv. Algoritmul enumeră instanţierile complete ale reţelei cu o probabilitate produs $\geq \varepsilon_{target}$.

Câteva exemple de sisteme care se încadrează în formalismul de mai sus sunt **HUGIN [And89], PATHFINDER [Heck91].**

3.3.1.3.Bayesianism şi cauzalitate

Cele mai riguros apropiate de diagnoza pe bază de model cauzal profund sunt reţelele cauzale. Acestea sunt reţele bayesiene în care părinţii unui nod reprezintă cauzele sale directe (din modelul natural al domeniului). În construirea unei reţele bayesiene, adăugarea nodurilor pornind de la cauzele "de bază" este o recomandare care îmbunătăţeşte mult eficienţa modelului, dar acesta nu este singurul avantaj: după cum demonstra Poole, reţelele cauzale pot fi modelate ca un program logic cu probabilităţi asociate diferitelor ipoteze, fiind astfel echivalente cu viziunea logică asupra abducţiei (vezi şi Secţiunea 3.4.1).

După cum observa autorul în [Pea01b], statistica este domeniul care se ocupă de descoperirea asocierilor ce pot apărea între variabile, sub forma unor coeficienţi de corelaţie, folosind tehnici de regresie şi estimare, şi orice concept statistic poate fi exprimat în termenii unei distribuţii de probabilitate (subiectivă sau bazată pe frecvenţe). În plus, statistica nu surprinde decât relaţiile statice din cadrul lumii modelate. Condiţiile dinamice care ţin de modificările posibile ale variabilelor trebuie modelate într-un formalism nou, specific cauzalităţii. O idee care decurge de aici este că probabilităţile condiţionale nu sunt totdeauna potrivite pentru modelarea cauzalităţii. În sprijinul acestei afirmaţii, un argument în plus este dat şi de următorul exemplu.

Conform teoriei probabilităţii, dacă H reprezintă o ipoteză şi E o mulţime de evidenţe, şi formalizăm cauzalitatea cu ajutorul probabilităţii condiţionale, atunci:

$$P(H|E) = 1 - P(H'|E) \quad (3.27)$$

implică o relaţie cauză-efect între E şi H' când există o relaţie cauză-efect între E şi H .
Un exemplu simplu din [GiaRil94] ne arată că această asociere nu este corectă:

"Dacă P(a absolvi | nota "A" la cursul X) = 0.7" (3.28)

aceasta nu înseamnă neapărat că:

$$P(\text{a nu absolvi} \mid \text{nota "A" la cursul X}) = 0.3, \quad (3.29)$$

întrucât există mulți alți factori ce pot încă împiedica absolvirea.

Astfel, Pearl a propus *modelul ecuațiilor structurale* pentru a trata sistemele cauzale [Pea01a], şi noțiunea de *intervenție* este noțiunea cheie pentru a distinge între o relație cauzală şi orice alt fel de dependență: într-o relație cauză –efect modificări ale părinților rezultă în modificări ale copiilor şi ale nici unui alt nod. Rețelele cauzale au fost folosite în diagnoza medicală în sisteme precum **CASNET** (pentru diagnosticul glaucomului [Lon01]), **Heart Disease Program** [LongNai92] .

3.3.1.4.Sistemul de diagnoză medicală CASNET

Detaliem în continuare sistemul CASNET [Weis74], întrucât acesta a constituit o sursă de inspirație şi pentru abordarea originală a acestei lucrări (în special în partea de reprezentare a cunoştințelor).

Reprezentarea cunoştințelor. Dacă domeniul medical abordat este suficient de bine înțeles încât permite reprezentarea detaliată a mecanismelor fiziologice ce stau în spatele declanşării simptomelor, este de bun simț să renunțăm la o reprezentare bazată doar pe asocieri boli-simptome (cum folosesc de exemplu sistemele de diagnoză medicală - clasice deja- PIP sau INTERNIST). Rețeaua cauzală a sistemului CASNET cuprinde stări de disfuncționalitate (diferite de bolile propriu zise) şi teste (informații externe ce definesc probabilitatea de existență a stărilor ascunse). Un statut particular îl au stările inițiale (primare din punct de vedere etiologic) şi cele finale (care nu au consecințe disfuncționale). Relațiile cauzale între noduri sunt legături de diferite ponderi nod cauză-nod efect, legături interpretate drept frecvența cu care primul nod îl cauzează pe al doilea. Nu sunt permise ciclurile în rețea. Rețeaua de stări de disfuncționalitate este un model simplificat al bolii, cu rolul de a ghida procesul de diagnoză. Nodurile pot avea statutul de confirmate sau infirmate prin teste.

Mecanismul de inferență. Inferența este probabilistă, bazându-se, în principiu, pe un proces de propagare a încrederilor/ neîncrederilor în adevărul nodurilor pe baza testelor de care sunt legate, creând căi acceptabile (adică fără noduri negate) în rețea. Aceste căi constituie potențiale explicații pentru diagnosticul final, şi focalizează atenția pe regiunile de interes ale rețelei. Planul de testare este definit prin intermediul structurii rețelei- se testează acele noduri "necunoscute" aflate pe căile aflate în construcție.

Practic, pentru fiecare nod se calculează două măsuri probabilistice: *ponderea* (estimarea verosimilității sale pe baza tăriei legăturilor cauzale cu nodurile confirmate/ infirmate cele mai apropiate-înrudite-) şi *starea* (estimarea verosimilității pe baza testelor direct relevante pentru nod). Astfel, de câte ori se obține rezultatul unui test nou, starea fiecărui

nod legat de testul respectiv este reactualizată: dacă rezultatul testului are încredere mai mică decât valoarea de încredere a nodului, nu modificăm nimic, dacă rezultatul este mai mare, valoarea nodului devine egală cu încrederea testului, dacă sunt egale dar de semne opuse, se raportează utilizatorului prezența unei contradicții.

Dezavantaje:

- Un prim dezavantaj al CASNET este moștenit de la rețelele bayesiene: după cum am subliniat mai sus, pentru rețelele multiplu conectate (cu mai mult de două căi între două noduri) inferența probabilistă este NP-dificilă. O problemă legată de acest aspect vine din faptul că se lucrează teoretic cu toată rețeaua simultan, o consecință nedorită fiind şi că multe noduri cu relevanță foarte mică sunt implicate în calcul. Aceasta afectează nu doar eficiența, dar are rezultate nedorite asupra rezultatului final al calculului (ponderi mai mari decât trebuie pentru anumite diagnostice, diagnostice care apar deşi ar trebui să lipsească etc).
- O altă problemă importantă în CASNET este felul în care se tratează agregarea rezultatelor testelor legate de un singur nod. Dacă trebuie redat faptul că două teste susțin un anumit nod doar când apar conjugate, atunci un test nou este definit pentru a reprezenta cele două teste împreună. Autorul argumentează că este suficient, în această situație, să considerăm tăria maximă a celor două teste ca rezultat al agregării, ceea ce, după părerea noastră, este discutabil în cazul general. În plus, abordarea este inaplicabilă acelor situații (destul de frecvente în diagnoză) când trebuie modelat faptul că o ipoteză e susținută doar de prezența unui "număr consistent" (criteriu vag) de teste din cele legate de ea.
- Maniera de tratare a contradicțiilor constituie, deasemenea, un dezavantaj. Modul de calcul al scorului unui nod prin adăugarea şi scăderea de cantități în funcție de confirmarea/ infirmarea unor teste poate duce la rezultate ambigue, greu de interpretat şi chiar greşite. Cea mai bună dovadă în acest sens este faptul că, dacă prin scăderi şi adunări repetate se junge la scor 0, sistemul raportează contradicția utilizatorului, el neştiind cum să o trateze. În plus, stabilirea pragului T ridică multe semne de întrebare: este clar că acest test ar trebui să fie dependent de nodul asociat, de numărul de teste confirmatoare/ infirmatoare etc., dar detalii clare despre cum ar trebui definite aceste dependențe nu avem. De aceea considerăm că o abordare categorică a problemei contradicțiilor constituie o soluție mai bună decât metoda probabilistă.

O îmbunătățire notabilă a sistemului CASNET a fost făcută în sistemul CHECK [TorCon89], dar şi în sistemul original DiaMed. Avantajele vor fi detaliate în secțiunile dedicate celor două sisteme (Secțiunea 4.4 şi Capitolul 8).

3.3.2. Sisteme de inferență fuzzy

Mulțimile fuzzy au apărut ca o necesitate de a modela incertitudinile non-statistice, în modelarea termenilor şi expresiilor lingvistice şi a raționamentului aproximativ. După cum a fost subliniat în Capitolul II, medicina este un domeniu în care mulțimile fuzzy îşi găsesc în mod natural aplicativitatea, datorită multiplelor surse de incertitudine,

imprecizie şi subiectivitate. O abordare recentă ce foloseşte agregarea fuzzy pentru diagnoza pacienților cu tremurături se găseşte în [TeoCheKan01].

Sistemele de inferență fuzzy reprezintă un formalism potrivit pentru folosirea unui model furnizat de un expert uman, construind o schemă simplificată a unui model complex. Specificul lor principal stă în formalismul de reprezentare a incertitudinii. Conceptele din cunoştințele asociate cu o anumită problemă (ontologia domeniului) sunt reprezentate ca *variabile lingvistice* (de exemplu: vârstă, înălțime, temperatură). Aceste variabile pot lua *valori lingvistice* definite ca mulțimi fuzzy (de exemplu *mic*, *mediu*, *mare*), definite pe un domeniu (discret sau continuu) asociat variabilei căreia îi aparțin.

Regulile care stau la baza unui sistem de inferență fuzzy pot partiționa spațiul de intrare astfel ca în fiecare subspațiu fuzzy comportarea sistemului este liniară, această proprietate fiind utilă în aproximarea funcțiilor complexe, nelineare.

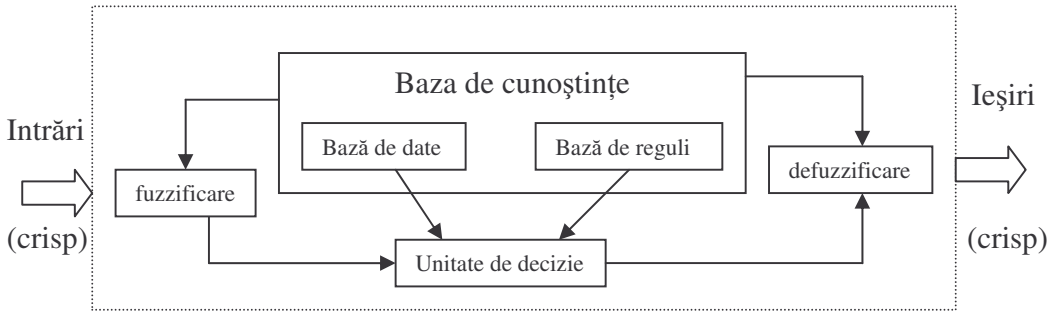


Figura 3.8. Schema unui sistem de inferență fuzzy

Raționamentul fuzzy (inferența folosind reguli fuzzy) constă din următorii paşi [Jan92]:

1. FUZIFICARE. Se compară variabilele de intrare cu funcțiile de apartenență la mulțimile fuzzy din antecedentul regulilor şi se obțin măsuri de compatibilitate ale intrării cu fiecare etichetă lingvistică.

2. Se combină printr-un operator de tip t-normă măsurile de compatibilitate pentru antecedentul fiecărei reguli şi se obține puterea de activare (ponderea) fiecărei reguli.

3. Se generează consecventul fiecărei reguli (fuzzy sau crisp), în funcție de ponderea fiecărei reguli.

4. DEFUZIFICARE. Se agregă consecvenții regulilor pentru a produce o ieşire crisp.

Există 3 tipuri importante de inferență fuzzy, în funcție de tipul regulilor folosite.

Tipul I.(FRF: Fire Rules First) Ieşirea sistemului de inferență este media ponderată a ieşirilor crisp ale regulilor, ponderile fiind date de puterea de activare a fiecărei reguli.

Tipul II.(CRF: Combine Rules First) Se aplică "max" (sau un alt operator de agregare de tip t-conormă) ieşirilor regulilor şi se obţine un rezultat fuzzy (o mulţime fuzzy), din care apoi se obţine valoarea finală (crisp) prin diferite metode: centrul de greutate, centrul ariei, media maximelor ş.a.

Tipul III. Se folosesc reguli de tip Takagi-Sugeno. Ieşirea fiecărei reguli este o combinaţie lineară de variabilele de intrare plus o constantă, ieşirea finală este media ponderată a ieşirilor regulilor.

Vom prezenta în continuare locul sistemelor de inferenţă fuzzy în cadrul raţionamentului aproximativ. Regula ce stă la baza raţionamentului aproximativ este modus ponens generalizat. De la aceasta s-a dedus regula de inferenţă compoziţională pentru raţionamentul aproximativ [Zad73]:

Premiză: Dacă x este A atunci Y este B

Fapt: x este A'

_____

Concluzie: y este B'

Concluzia B' se determină ca o compunere între faptul descris de premiză şi operatorul de implicaţie fuzzy: $B'=A' \circ (A \rightarrow B)$. Deci:

$$B'(v) = \sup_{u \in U} \min\{A'(u), (A \rightarrow B)(u,v)\}, \forall v \in V. \quad (3.30)$$

(S-a folosit compunerea sup-min). Între operatorii pentru implicaţia fuzzy menţionăm implicaţia Mamdani[1] (foarte frecvent utilizată în controlul fuzzy).

Regulile fuzzy de tipul "dacă-atunci" sunt expresii de tipurile prezentate mai jos la punctele a), b) ce captează regulile empirice folosite de experţi umani în condiţii de imprecizie şi incertitudine; fiecare regulă poate fi privită ca o descriere locală (a unei părţi limitate) a sistemului.

a)Reguli de tip Mamdani

"Dacă A atunci B", unde A şi B reprezintă mulţimi fuzzy.

Exemplu: "Dacă presiunea este ridicată, atunci volumul este mic."

R_1: *DACĂ* x_1 *este* A_{11} *ŞI…ŞI* x_m *este* A_{m1} *ATUNCI* *y este* B_1
R_2: *DACĂ* x_1 *este* A_{12} *ŞI…ŞI* x_m *este* A_{m2} *ATUNCI* *y este* B_2
…
R_n: *DACĂ* x_1 *este* A_{1n} *ŞI…ŞI* x_m *este* A_{mn} *ATUNCI* *y este* B_n
Fapt: x_1 *este* x_{10}, x_2 *este* x_{20},… x_m *este* x_{m0},

_____

[1] $x \rightarrow y = \min\{x,y\}$.

Concluzie: Y este B

b)Reguli de tip Takagi-Sugeno
"Dacă A atunci y=f(A)".
Exemplu: "Dacă viteza este mare, atunci forța =k* (viteza)^2 ".

$$R: IF\ x_1\ is\ A_1\ AND \ldots AND\ x_k\ is\ A_k\ THEN\ y=p_0+p_1x_1+\ldots+p_kx_k, \quad (3.31)$$

unde *y* este ieşirea sistemului, iar $A_1, \ldots, A_k$ sunt mulțimi fuzzy cu funcții de apartenență lineare, reprezentând subspațiul fuzzy în care regula *R* este aplicabilă. Specificul acestor reguli stă în faptul că partea de consecvent este descrisă de o ecuație non-fuzzy, sub forma unei funcții dependentă de variabila de intrare (din antecedent).

Iată un exemplu de schemă de raționament fuzzy pentru cazul sistemelor MISO (Multiple Input Single Output) [Ful95], pentru sistemul reprezentat la punctul a). Concluzia se calculează ca $Agg(\text{Fapt}^\circ R_1,\ldots \text{Fapt}^\circ R_n)$, obținându-se:

$$B=Agg(x_{10}\times\ldots\times x_{m0}\ ^\circ R_1,\ldots, x_{10}\times\ldots\times x_{m0}^\circ R_n). \quad (3.32)$$

Ieşirea fuzzy a acestei baze de cunoştințe se calculează astfel:

- Nivelul de procesare al regulii R_i este $A_{1i}(x_{10}) \times\ldots\times A_{mi}(x_{m0})$;
- Ieşirea fiecărei reguli R_i este mulțimea fuzzy $B_i'(w)=A_{1i}(x_{10}) \times\ldots\times A_{mi}(x_{m0})\rightarrow B_i(w)$, $\forall w \in W$, (W fiind mulțimea de discurs a variabilei Y);
- Ieşirea globală B a sistemului este: $B(w)=Agg\{B_1',\ldots, B_n'\}, \forall w \in W$.

(S-a folosit metoda CRF (**C**ombine **R**ules **F**irst)).

Operatorii care stau la baza agregării pot fi atât t-norme cât şi t-conorme, după cum agregarea are o semnificație conjunctivă sau disjunctivă. Opțiuni diverse există şi pentru produsul cartezian sau implicație (spre exemplu, t-norma produs).

Se observă că în modelul propus mai sus, funcția de apartenență ce defineşte mulțimea fuzzy-consecvent este o valoare lingvistică pentru una şi aceeaşi variabilă (*y*). Dacă dorim o valoare crisp apelăm la defuzzificare. Dacă folosim modelul fuzzy simplificat, în care consecventul fiecărei reguli este de forma $y\ este\ y_i$, este evident că nu mai avem nevoie de defuzificare, ieşirea y_0 calculându-se, de exemplu, după formula:

$$y_0 = \frac{\alpha_1 y_1 + \ldots + \alpha_n y_n}{\alpha_1 + \ldots + \alpha_n}, \quad (3.33)$$

unde $\alpha_i=(A_{1i} \times\ldots\times A_{mi})(x_{10},\ldots, x_{m0})$, i=1,…m.

Din punct de vedere matematic, sistemele fuzzy sunt funcții ce pot aproxima un proces dat. Ele sunt aproximatori universali, proprietate ce relevă adecvarea utilizării lor atât în problemele de decizie cât şi în cele de control automat [TeoCheGal89].

3.3.3.Măsuri de posibilitate

După cum observau autorii în [RusNor02], funcționalitatea relativ la valoarea de adevăr modelată prin intermediul probabilităților nu este recomandată în raționamentul incert. Iată un scurt exemplu prin care se ilustrează această afirmație. Dacă avem 100 de premize, agregate conjunctiv, şi fiecare dintre ele are probabilitatea 0.99, concluzia acestora va avea o probabilitate de abia 0.37-adică ar putea fi, eventual, neglijată în urma aplicării unui prag. În plus, am exemplificat în Secțiunea 3.3.1.3 dezavantajele relației P(H|E) =1-P(H'|E).

S-a ajuns, astfel, la probabilitățile subiective, dintre care un prim model a fost al *factorilor de încredere* – folosit în Mycin. Mulțimile vagi şi teoria posibilității au ajutat la depăşirea unor dezavantaje ale metodelor probabiliste, şi pot fi privite drept un model dezvoltat / generalizat al factorilor de încredere.

Măsurile de posibilitate şi mulțimile vagi au fost concepute special pentru modelarea incertitudinii şi impreciziei. Aşa cum observau Dubois şi Prade în [DubPra85], incertitudinea se referă la valoarea de adevăr, în timp ce imprecizia este o noțiune legată de conținut. O valoare a unei date este cu atât mai puțin certă, cu cât este mai precisă. Pe baza distribuțiilor de posibilitate s-a încercat dezvoltarea rețelelor cauzale posibiliste, care se doresc o alternativă a rețelelor bayesiene (raționament incert cu date precise), pentru a trata probabilist date imprecise. [**GebKru96**]

Definiție([DubPra85]). Fie Ω evenimentul întotdeauna cert şi $\emptyset$ evenimentul întotdeauna imposibil. Fie $A \subseteq \Omega$ un eveniment. Se numeşte *măsură de încredere* o funcție g de evenimente (g(A) furnizat eventual de un expert uman) care satisface:

- $g(\emptyset)=0, g(\Omega)=1$; (3.34)
- Dacă $A \subseteq B$, atunci $g(A) \leq g(B)$ (monotonie în sensul incluziunii). (3.35)

Consecințe ale axiomei de monotonie. $\forall A, B \subseteq \Omega$ avem:

- $g(A \cup B) \geq \max(g(A), g(B))$; (3.36)
- $g(A \cap B) \leq \min(g(A), g(B))$. (3.37)

Definiție([DubPra85]). *Măsurile de posibilitate* sunt un caz particular al măsurilor de încredere, pentru care:

$$\Pi(A \cup B) = \max(\Pi(A), \Pi(B)). \quad (3.38)$$

3.3.3.1.Proprietăți ale măsurilor de posibilitate

- $\Pi_E(A) = \begin{cases} 1, dacă \cap E \neq \varnothing \\ 0, altfel \end{cases}$; (3.39)

- $\max(\Pi(A), \Pi(\bar{A})) = 1$; (3.40)

- dacă avem definită o distribuție de posibilitate pe **Ω**-finită $\pi:\Omega \rightarrow [0,1]$ atunci:

$$\Pi(A) = \sup\{\pi(\omega) \mid \omega \in A\}. \quad (3.41)$$

Pe baza proprietăților menționate s-a definit versiunea posibilistă a relației 3.27:

$$\max(\Pi(h \mid e), \Pi(\neg h \mid e)) = 1, \quad (3.42)$$

care înlătură neajunsurile menționate mai sus (spre exemplu, este evident că dacă absența lui *h* e imposibilă în prezența lui *e*, *h* este posibil în prezența lui *e*).

Se observă că distribuția oricărei măsuri de posibilitate ce ia valori în intervalul [0,1] poate fi văzută ca funcția de apartenență a unei mulțimi fuzzy F:

$$\forall \Pi, \exists F \in [0,1]^{\Omega} \forall \omega \in \Omega: \quad \Pi(\{\omega\}) = \Pi(\omega) = \mu_F(\omega) \quad (3.43)$$

Reciproc, orice mulțime fuzzy normalizată poate fi tradusă natural ca o măsură de posibilitate.

3.3.3.2.Raționament incert şi imprecis folosind măsuri de posibilitate

În mecanizarea raționamentului există două puncte de vedere în reprezentarea implicațiilor:

- **logic** (implicația materială): $p \rightarrow q \equiv \neg p \vee q$;

- **funcțional:** o regulă reprezintă specificarea parțială a unei funcții $f: \{p, \neg p\} \rightarrow \{q, \neg q\}$.

În cadrul punctului de vedere logic regulile de derivare sunt *modus ponens* şi *modus tollens* (contrapozitiva); în cazul funcțional, regula este interpretată în termeni de condiționare şi valoarea de adevăr a regulii p→q nu este definită decât dacă p este adevărată:

$$v(p \wedge q) = \min(v(q \mid p), v(p)) \quad (3.44)$$

Diferența de bază între cele două abordări este următoarea: în logică, dacă p este fals, nu pot să exprim că "p nu implică q" fără a ajunge la o contradicție (p nu implică q $\equiv p \wedge \neg q$, care nu poate fi adevarată indiferent de valoarea lui q, cât timp p este falsă), în timp ce în cazul funcțional v(q|p) nu depinde de valoarea lui p (chiar dacă p este fals, v(q|p)=0 nu presupune o contradicție.)

V(p→q)	**V(p)**	**V(q)**
1	**1**	**1**
1	**0**	**{0,1}**
0	**1**	**0**
0	**0**	**∅ (caz imposibil)**

Tabel 3.1. Modus ponens pentru cazul logic

V(q|p)	**V(p)**	**V(q)**
1	**1**	**1**
1	**0**	**{0,1}**
0	**1**	**0**
0	**0**	**{0,1}**

Tabel. 3.2. Modus ponens generalizat pentru abordarea de tip condiționare

Deci generalizarea modului ponens se exprimă prin:

$$v(q) \geq \min (v(q|p), v(p)) \quad (3.45)$$

Explicația pentru diferența menționată mai sus stă în faptul că în abordarea funcțională dispare cazul imposibil, deci v(p) şi v(q|p) pot fi definite independent (în timp ce v(p) şi v(p→q) sunt legate prin inegalitatea v(p)≤v(p→q)).

Inferență cu premize vagi

Plecând de la propozițiile p:"X este A" şi q:"Y este B" (unde A şi B sunt mulțimi vagi definite pe suportul S, respectiv T), existența unei legături cauzale între p şi q (dacă p atunci q) se poate reprezenta ca o distribuție de posibilitate condițională $\pi_{Y|X}$. Din relațiile

$$v(p \wedge q) = \min(v(q \mid p), v(p)) \quad (3.46)$$

şi

$$v(q) \geq v(p \wedge q) \quad (3.47)$$

se deduc:

$$v(q) \geq \min(v(q \mid p), v(p)) \quad (3.48)$$

şi

$$\Pi(p \wedge x) = \Pi(x \mid p) * \Pi(p)(x = q, \neg q) \quad (3.49)$$

Soluția maximală pentru 3.49 este măsura de posibilitate condițională cea mai puțin specifică:

$$\Pi(x \mid p) = \sup\{r \mid \Pi(p \wedge x) = r * \Pi(p)\}; x = q, \neg q. \quad (3.50)$$

Ținând cont că în anumite condiții unei distribuții de posibilitate i se poate asocia în mod unic o mulțime vagă şi reciproc, rezultă, în continuare, că putem exprima "dacă X e A atunci Y e B " prin:

$$\forall t \in T : \mu_B(t) \geq \sup_{s \in S} \pi_{Y|X}(t, s) * \mu_A(s) \quad (3.51)$$

(μ_A şi μ_B sunt distribuțiile de posibilitate asociate mulțimilor A, respectiv B, iar $\pi_{Y|X}$ este necunoscută. Dacă "*" este o normă triunghiulară continuă, cea mai mare soluție pentru 3.51 este dată de ([DubPra85]):

$$\pi_{Y|X}(t, s) = \mu_A(s) * \rightarrow \mu_B(t). \quad (3.52)$$

Exemple:

$$a * b = \min(a, b) \Rightarrow a * \rightarrow b = \begin{cases} 1, a \leq b \\ b, altfel \end{cases}; \text{(Gödel)} \quad (3.53)$$

$$a * b = a.b \Rightarrow a * \rightarrow b = \begin{cases} 1, a = 0 \\ \min\left(1, \frac{b}{a}\right), altfel \end{cases}; \text{(Goguen)} \quad (3.54)$$

$$a * b = \max(0, a + b - 1) \Rightarrow a * \rightarrow b = \min(1, 1 - ab); \text{(Lukasiewicz)} \quad (3.55)$$

Concluzionând, fiind date două mulțimi vagi A şi B , se poate defini o distribuție de posibilitate $\pi_{Y|X}$ după modelul descris, care măsoară cât de posibil este ca între conceptele descrise de B şi A să existe o relație de condiționare, ținând cont că, teoretic, trebuie să se respecte condiția 3.46.

Alte metode de aplicare ale posibilităților în modelarea incertitudinii raționamentului. Regulile generale cu excepții sunt una din sursele principale ale incertitudinii în cadrul unui raționament: concluziile lor pot fi infirmate de excepțiile ulterior descoperite. Incertitudinea nu este asociată atât cu o piesă anume de informație ci, mai degrabă, cu ordinea (aleatoare) defectuoasă în care pot proveni evidențele de la mediul extern. Raționamentul folosind cuantificatori vagi reprezintă o posibilă abordare a regulilor cu excepții. Aceştia sunt cuantificatori intermediari între $\exists$ şi $\forall$, cum ar fi: "*în general", "câțiva", "cea mai mare parte"* etc. Spre exemplu, Zadeh propune ca gradul de adevăr al unei propoziții p : "QA sunt B" (unde Q este un cuantificator vag) să se calculeze cu expresia:

$$v(p) = \mu_Q\left(\frac{|A \cap B|}{|B|}\right) = \mu_Q\left(\frac{\sum_{\omega\in\Omega}\min(\mu_A(\omega),\mu_B(\omega))}{\sum_{\omega\in\Omega}\mu_A(\omega)}\right) \quad (3.56)$$

O altă sursă de incertitudine este lipsa de fiabilitate a surselor. Gradele de incertitudine relativ la o singură propoziție (despre care primim informații din surse distincte, şi care pot fi contradictorii) se pot combina folosind teoria posibilității sau regula lui Dempster ([DubPra85]). Aceste tipuri de combinare se exprimă prin conjucția informațiilor urmată de o renormalizare a rezultatului. Fiind însă discontinue în vecinătatea valorilor ce exprimă conflictul total al surselor, sunt necesare tipuri de combinare neconjunctivă în cazul unui conflict sever.

Inferența posibilistă descrisă mai sus a fost aplicată mai multor sisteme de diagnoză, între care menționăm **PROTIS([Sou83]:** reguli incerte în diagnoza şi tratarea diabetului), **POSSINFER [GebKru96], DIAMS [CayDubPra94].**

Din păcate însă, şi abordarea posibilistă are dezavantajele sale. Spre exemplu, relația 3.35 de monotonie în sensul incluziunii păstrează dezavantajul menționat în introducere, privind măsura unei conjuncții de propoziții. În plus, rețelele posibiliste suferă încă de lipsa unei semantici clare, fapt ce a determinat ca aplicațiile să fie destul de restrânse ca număr şi domenii.

3.3.4.Modele simbolice

Cele mai cunoscute şi utilizate structuri preponderent simbolice sunt arborii de decizie şi sistemele expert. Aceste formalisme au fost printre primele utilizate în raționamentul simbolic şi constituie o etapă intermediară între clasificarea metrică şi cea non-metrică.

Cele două structuri menționate folosesc de obicei cunoştințe de suprafață, adică euristici ce rezumă cunoştințele unui expert uman. Spre deosebire de sistemele ce folosesc cunoştințe profunde, nu există lanțuri cauzale, legături tip-cauză efect de la o regulă la alta [GiaRil94], astfel că inferența este mai rapidă dar şi explicațiile mai primitive. Sunt

de preferat atunci când elementele lanțului cauzal profund nu sunt accesibile pentru testare/observare. Componentele principale ale acestor sisteme sunt baza de cunoştințe şi mecanismul de inferență.

Arbori / latice de decizie
Sunt modele a căror structură ierarhică le face foarte potrivite pentru clasificare, şi reprezintă o metodă intuitivă de clasificare de forme descrise prin caracteristici simbolice printr-o secvență de întrebări, unde alegerea întrebării la fiecare pas depinde de răspunsul la întrebarea precedentă. Un algoritm foarte cunoscut de construire a arborilor de decizie din date este reprezentat de metoda CART (Classification And Regression Trees) [DudHarSto00, Hay98].

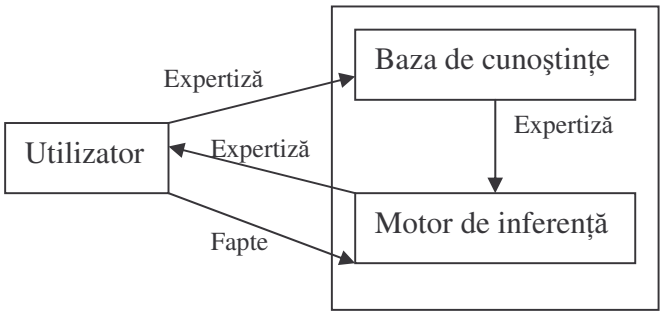


Figura 3.9.Structura generală a unui sistem expert

Sisteme pe bază de reguli.
Structurile de decizie pot fi traduse automat în reguli, aşa că acestea pot fi văzute ca o generalizare a arborilor de decizie, în plus putând lucra şi cu variabile.

MYCIN[GiaRil94]*Modelul factorilor de certitudine (încredere)* ce stă la baza acestui sistem expert de diagnosticare a infecțiilor hematologice a fost propus de Shortliffe şi inspirat de teoria confirmării a lui Carnap. Motivația sa a plecat de la faptul că modelul bayesian nu este foarte potrivit pentru domeniul medical unde numărul testelor accesibile este limitat din diverse motive, şi, în plus, rezultatele se obțin secvențial, în timp, unele cu întârziere. În plus, sunt prea multe probabilități condiționale de cunoscut apriori, şi apare şi problema menționată în discuția cauzalității într-o rețea bayesiană (3.27- 3.29).

În încercarea de a depăşi modelul bayesian, Shortliffe propune o măsură care combină într-un singur număr (=factor de încredere) încrederea şi neîncrederea într-o ipoteză condiționată de prezența unei anumite evidențe:

$$CF(H,E)=MB(H,E)-MD(H,E), \quad (3.57)$$

unde MB=măsură a încrederii şi MD-măsură a neîncrederii [GiaRil94].

Factorii de încredere astfel obținuți sunt folosiți la ierarhizarea ipotezelor de diagnostic.

Combinarea evidenţelor din antecedentul unei reguli se face ca şi în PROSPECTOR (vezi mai jos), utilizând logica fuzzy. Încrederea finală într-o ipoteză se face prin combinarea după un model propriu a regulilor al căror consecvent se referă la ipoteza respectivă. Una dintre problemele mari ale acestui sistem de combinare a evidenţelor este că interacţiuni nedorite apar între regulile bazei de cunoştinţe dacă aceasta nu este atent construită.

S-a demonstrat că teoria factorilor de încredere nu este decât o aproximare a raţionamentului probabilist, şi succesul MYCIN se datorează de fapt simplităţii teoriei domeniului (lanţuri inferenţiale scurte + ipoteze simple), teoretic existând probleme cu modelul propus.

PROSPECTOR [GiaRil94] Este un sistem expert "clasic" ce foloseşte raţionament probabilist (cu aplicaţii în depistarea zăcămintelor în geologie). Legăturile evidenţe-ipoteze sunt memorate sub forma unor *reţele de inferenţă*, asemănătoare reţelelor semantice. Nu este totuşi un sistem pur probabilist întrucât foloseşte logica fuzzy şi factori de încredere asemănători celor din MYCIN pentru combinarea evidenţelor.

În încercarea de a compara sistemele expert şi modelele probabiliste, observăm că sistemele expert sunt mai puţin precise faţă de modelele pur probabiliste (bayesiene), dar mai transparente, prin faptul că folosesc pentru clasificare, în principal, informaţie simbolică (reguli, liste de atribute).

Iată câteva diferenţe între sistemele logice pe bază de reguli şi sistemele probabiliste conform [RusNor02]:

- Sistemele logice pe bază de reguli au proprietatea de *localitate,* (rezultatul aplicării unei reguli nu depinde de alte reguli) spre deosebire de sistemele probabiliste unde trebuie luată în considerare toată evidenţa disponibilă. (Reţelele bayesiene, spre exemplu, sunt totuşi *local structurate*, în sensul că un nod interacţionează numai cu un grup restrâns de noduri în calcularea probabilităţilor condiţionale -arcele din graful ce modelează reţeaua reprezintă un mijloc de ghidare a raţionamentului; acelaşi tip de localitate poate fi întâlnit şi în argumentele din sistemele de argumentaţie-);

- Sistemele logice sunt în general *funcţionale relativ la valoarea de adevăr* (valoarea de adevăr a unei propoziţii complexe poate fi calculată pe baza valorilor componentelor), în timp ce probabilităţile nu se pot combina funcţional decât în ipoteza de independenţă (adesea nerealistă).

3.4. Diagnoza pe bază de model

Introducem în continuare câteva dintre aspectele de bază ale diagnozei abordată cu metode logico-simbolice, pe bază de model cauzal de profunzime. Soluţiile sunt exacte, transparente (bune facilităţi de explicare), dar aici apare în prim plan problema eficienţei.

3.4.1.Abducţia şi tehnicile bazate pe consistenţă

Abducţia reprezintă una din schemele generale de raţionament ce stau la baza procesului de diagnoză. În trihotomia lui Peirce există trei tipuri de inferenţă: "deducţie, inducţie, abducţie". Pe scurt, fiind date premiza majoră **a** ⊃ **b**, premiza minoră **a** şi concluzia **b**, Peirce afirmă că:

- *deducţia* pleacă de la **a** ⊃ **b** şi **a** pentru a deduce concluzia **b**,
- *inducţia* pleacă de la **a** şi **b** pentru a produce regula **a** ⊃ **b**, iar
- *abducţia* pleacă de la **a** ⊃ **b** şi **b** pentru a construi o explicaţie plauzibilă pentru **a.**

Termenul de "abducţie" a fost introdus de Peirce în anii 1800 şi definit drept "procesul de formare a unei ipoteze explanatorii plecând de la o mulţime de observaţii".

Structura generală a unui raţionament abductiv

D=observaţii adevărate, H explică D, nu există H' mai bun decât H care explică D
H este presupus adevărat

Aplicaţii. Printre aplicaţiile cele mai importante ale abducţiei se numără raţionamentul în prezenţa informaţiilor incomplete, generarea explicaţiilor cauzale în diagnoza defectelor, simularea reţelelor bayesiene (David Poole: Independent Choice Logic).

Problemele care trebuie urmărite în cadrul unui raţionament abductiv sunt:

- Care sunt datele care necesită explicaţii?
- Ce înseamnă că "H explică D"?(a) relaţie de cauzalitate;b)consecinţă logică)
- Ce înseamnă set de ipoteze "mai bun"?

Definiţia abducţiei în termeni logici

Fie T teoria domeniului, D - formula ce trebuie explicată. O explicaţie pentru D în T este o formulă E care îndeplineşte condiţiile:

- $T \cup E$ este consistent;
- $T \cup E \vdash D$;
- E este formată numai din predicate asumabile şi e cea mai simplă care satisface 1 şi 2.

De nonmonotonicitatea abducţiei ne putem da seama urmărind exemplul următor ([Dup94]).

Fie

$$p_1 \rightarrow q, ..., p_n \rightarrow q,$$

unde p_i,q -propoziţii din teoria T a domeniului, şi implicaţia materială este folosită pentru a modela relaţii cauză-efect (pentru un comentariu legat de cât este de potrivită această utilizare a se vedea Yeov Shoham-1987). Raţionamentul abductiv spune, în acest exemplu, că pentru a explica q, trebuie să asumăm (să considerăm adevărat) cel puţin un p_i -ceea ce nu este corect decât dacă am presupus că avem cunoştinţe complete în T despre q. Abducţia este aşadar o formă de raţionament de tip invalidabil, întrucât depinde de cunoştinţele (posibil incomplete) disponibile la un moment dat. Pentru a face explicită ipoteza de completitudine a cunoştinţelor, se recurge la o transformare sintactică a lui T (completarea Clark), care iniţial a fost folosită pentru a defini o semantică declarativă pentru "naf" ("negation as failure" =negaţia ca imposibilitate de a demonstra) în cadrul programării logice. Diferenţa faţă de utilizarea iniţială este că atomii abductibili nu se completează, şi nici cunoştinţele factuale (observaţiile). Ideea este de a transforma implicaţiile în echivalenţe, după modelul următor: a→b, c→b se înlocuiesc în teoria completată prin $b \leftrightarrow a \vee c$. Un rezultat important spune că formula ce caracterizează explicaţiile rezultă deductiv din teoria completată şi observaţii: $T_C, \psi \models F$, unde T_C este teoria completată, ψ reprezintă observaţiile, şi F reprezintă explicaţiile([Dup94]).

***

Definiţie[DixJ94]. Un *program logic general disjunctiv* constă dintr-un număr finit de reguli cu un număr arbitrar de clauze pozitive în consecvent:

$$A_1 \vee ... \vee A_n \leftarrow B_1, ..., B_m, \neg C_1, ..., \neg C_l \text{, unde n} \geq 1. \quad (3.58)$$

Dacă l=0 programul este *pozitiv disjunctiv*, dacă n=1 programul este *normal*, dacă n=1 şi l=0, programul este *pozitiv* (sau *definit*).

Un program logic este *ierarhic* dacă nu există dependenţe ciclice între atomii clauzelor (A depinde de B dacă A apare în consecvent şi B în antecedent). Un program logic este *stratificat* dacă dependenţele circulare între atomi nu conţin un număr impar de arce negative (adică dependenţe de un atom negat).

Rezoluţie SLD (Single Linear Derivation). Rezoluţia SLD reprezintă semantica procedurală a programelor logice definite. O interogare (query) Q este o conjuncţie de literali, posibil cu variabile libere. În SLD, negaţia lui Q este rezolvată faţă de clauze din programul P (aplicând eventual substituţii, unde este necesar şi posibil) folosind regula de inferenţă:

$$\frac{C \vee A \ , \ D \vee \neg A}{C \vee D} \quad (3.59)$$

Dacă pe parcurs se obține clauza vidă (fără literali), -adică o contradicție-, am demonstrat Q prin reducere la absurd.

SLDNF = SLD+ Negația ca eşuare finită (negation as finite failure) : $\neg L$ reuşeşte dacă L eşuează într-un număr finit de paşi.

Dacă se consideră că T este program logic propozițional ierarhic (deci şi stratificat) se cunosc algoritmi eficienți pentru rezolvarea problemei. Fie Ψ mulțimea de observații disponibile.

Spre deosebire de definiția generală a abducției în termeni logici, aici T poate conține şi literali negativi. Negația explicită este utilă (în afara motivelor enunțate la programarea logică) în modelarea observațiilor negative (ce pot discrimina între ipoteze competitive), şi în cadrul teoriei explanatorii (pentru a modela interacțiunile între cauze-de exemplu, fenomenele de anulare reciprocă).

Definiția la meta-nivel a abducției([Dup94]). Fie P=<T, Ψ> o problemă de abducție. O *m-explicație* pentru P este o mulțime de atomi abductibili E astfel încât: $T \cup E \vdash_{SLDNF} \Psi$.

Definiția la nivel obiect a abducției([Dup94]). Fie P=<T, Ψ> ca mai sus. Se numeşte *formulă explanatorie* pentru P o formulă F formată din atomi abductibili, cea mai specifică, astfel încât: $T_C,\psi \models F$.

Teoremă(legătura abducție - deducție)([Dup94]). Fie P=<T, Ψ> o problemă de abducție, F o formulă explanatorie pentru P, E o mulțime de atomi abductibili, şi M o interpretare astfel încât: $M \models \alpha$ ddacă $\alpha \in E$. Atunci E este o m-explicație ddacă $M \models F$.

***

Raționamentul abductiv este intratabil în cazul general. S-au folosit însă diverse tehnici euristice şi probabilistice pentru focalizarea sa şi îmbunătățirea eficienței. Spre exemplu, în [Dup94] autorul propune o arhitectură specifică raționamentului abductiv (AID), care calculează explicații minimale, folosind însă numai modele monotone pentru teoria domeniului. Este prezentat un caz tipic de raționament pe bază de model. Acest tip de raționament este folosit în sistemele de IA care utilizează un model declarativ al domeniului, şi abordarea este specifică sistemelor expert de generația a doua, care se bazează pe cunoştințe de "profunzime", spre deosebire de cunoştințele "superficiale"(de suprafață) ale sistemelor expert de generația întâi. Avantajul cunoştințelor de suprafață este că permit "salturi" în procesul de raționament, care devine astfel mai eficient. Baza de cunoştințe a sistemelor expert de generația întâi cuprinde expertiza unui expert al domeniului, cunoştințe de control şi cunoştințe specifice rezolvării problemei. În ceea ce urmează până la finalul lucrării vom considera că termenul de "model" (din sintagma "diagnoză pe bază de model") se referă la structurile cauzale de profunzime ale domeniului.

Avantajele sistemelor bazate pe model (i.e.cunoştinţe profunde, ce surprind structurile componente ale sistemului şi legăturile cauzale între acestea) sunt abilitatea de a rezolva probleme necunoscute expertului uman şi facilitatea de a explica concluziile procesului de raţionament, -care devine însă foarte complex pe un astfel de model declarativ. Una dintre euristicile care ajută în asemenea situaţii este "compilarea cunoştinţelor" (knowledge compilation): acestea sunt prelucrate pentru a extrage condiţii uşor observabile (="operaţionale") cu ajutorul cărora se evită construirea de explicaţii inconsistente/redundante, şi se discriminează între explicaţii (a se vedea mai jos).

Există două tipuri de formalizări logice ale diagnozei pe bază de model. Formalizarea abductivă este preferată atunci când dispunem de un model "de defect" (model al comportamentului anormal) al sistemului, şi acest model este complet-în acest context, prin diagnostic se înţelege o mulţime minimală de ipoteze de anormalitate care "acoperă" (implică) observaţiile (conform observaţiei de mai sus, în condiţii de completitudine, abducţia se reduce la deducţie)([Kon92]). Dacă însă dispunem de un model al comportamentului corect, se foloseşte diagnoza pe bază de consistenţă: în cadrul acesteia, diagnostic înseamnă o mulţime minimală de componente ale sistemului asumate defecte, astfel încât comportamentul corect al celorlalte componente este consistent cu observaţiile. Noţiunea de explicaţie diferă de la o abordare la alta: dacă în cadrul diagnozei bazate pe consistenţă o explicaţie pentru o manifestare m este tot ceea ce nu susţine $\neg$m, în abducţie explicaţia trebuie să susţină direct m. (În abducţie este necesar i$\rightarrow$o, în consistenţă este necesar ca i$\wedge$o să nu fie contradictorie.) Avantajul abordării pe bază de consistenţă stă în validitatea logică, indiferent de completitudinea modelului. Dezavantajele, legate de imposibilitatea de a explica rezultatul şi de a identifica exact defectul pot fi depăşite prin integrarea abducţiei în acest cadru nou, după cum reiese din definiţiile următoare.

În [ConTor92] autorii propun o viziune unificată a celor două direcţii, o problemă de diagnostic fiind privită ca o problemă de abducţie cu constrângeri de consistenţă, astfel:

Definiţie * ([ConTor92]). O problemă de diagnostic (DP: **D**iagnostic **P**roblem) este alcătuită din:

$$DP=<<BM, Comp>, CXT, OBS>, \quad (3.60)$$

unde BM (**B**ehavioral **M**odes) este o mulţime de clauze Horn care descriu structura şi comportamentul sistemului, Comp este mulţimea de componente a sistemului, CXT este mulţimea informaţiilor contextuale (nu trebuie explicate, sunt folosite doar la modelarea sistemului).

Definiţie([ConTor92]). Dacă se partiţionează OBS în Ψ^+ (observaţiile pozitive) şi Ψ^- (observaţiile negative), şi considerăm problema de abducţie:

$$AP=<<BM,COMP>,CXT\ \Psi^+,\ \Psi^->>, \quad (3.61)$$

(adică se redefineşte o problemă de diagnostic ca o problemă de abducţie cu constrângeri de consistenţă), se defineşte o *explicaţie* pentru AP o asignare W de stare (normal/anormal) fiecărei componente din COMP care satisface condiţiile:

1. W acoperă Ψ^+: $\forall m \in \Psi^+ : BM \cup CXT \cup W \models m;$ (3.62)

2. W consistentă cu Ψ^-: $\forall \neg m \in \Psi^- : BM \cup CXT \cup W \not\models m;$ (3.63)
   ($BM \cup CXT \cup W \cup \Psi^-$ este consistentă).

În Figura 3.10., (1) corespunde unei probleme abductive "a la Poole"(pur abductivă) [Po88], iar (2) corespunde unei probleme de diagnoză "a la de Kleer"(pur consistentă) [deKleerMackRei92].

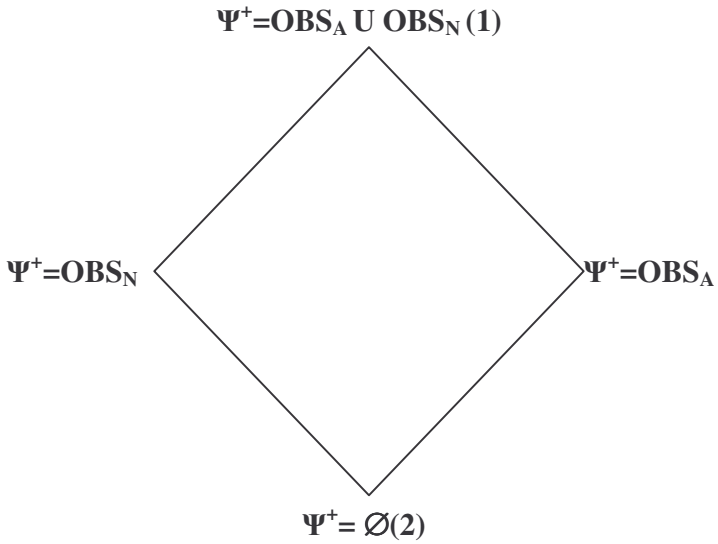


Figura 3.10.Laticea problemelor de diagnoză

Definiţia * este preluată şi în [ConTor92], iar $T=<BM,COMP>$ este un model de comportament al sistemului, format dintr-o mulţime de clauze Horn de tipul:

$$S_1(X_1) \wedge ... \wedge S_n(X_n) \wedge C_1(Y_1) \wedge ... \wedge C_m(Y_m) \wedge f(X_1,...,X_n,Y_1,...,Y_m,Z) \rightarrow S(Z)$$
(n≥1, m≥0) (3.64)

cu restricţiile:

- fiecare S_i este un simbol fie pentru o stare internă, fie pentru o stare iniţială;
- C_j este un simbol de context;
- S poate fi fie o stare internă, fie o manifestare;

- *f* este o relaţie funcţională ce defineşte legătura între valorile atributelor predicatelor din corpul clauzei şi cele ale concluziei.

Clauzele din modelul comportamental pot fi citite ca relaţii cauză-efect, fiind foarte asemănătoare cu ecuaţiile structurale ale lui Pearl ([Pea01a,b]), şi putând fi, dealtfel, asamblate într-o reţea cauzală. Modelul de mai sus este transformat într-un model complet prin adăugarea unui α (=cauză/element necunoscut) în membrul stâng pentru relaţiile incomplete/cauze absente, dar această modificare atrage după sine considerarea implicaţiei "→" ca o relaţie de tip MAY (**posibil** să implice) (aici se vede cum incompletitudinea cunoştinţelor se transformă într-o sursă de imprecizie a raţionamentului).

Aspectele procedurale ale sistemului AID sunt interesante întrucât descriu o tehnică originală de compilare a cunoştinţelor (euristică al cărei rol a fost menţionat mai sus). Algoritmul de bază generează explicaţii plecând de la setul curent de observaţii, explicaţiile prezic anumiţi parametri încă neobservaţi, se testează un parametru ales pentru discriminare între ipotezele explicative, se reactualizează mulţimea de observaţii curente şi se reia ciclul până când nu se mai poate rafina mulţimea de ipoteze. În acest algoritm, se împletesc căutarea înapoi pentru a determina cauzele atomilor din Ψ^+ cu înlănţuirea înainte pentru a testa consistenţa ipotezelor candidat (acestea nu trebuie să prezică valori din Ψ^-). Modul optim în care ar trebui combinate aceste două tipuri de inferenţă este încă o problemă deschisă. Autorul defineşte un tip particular de elemente observabile - *condiţiile necesare şi suficiente* pentru un anumit nod al reţelei cauzale, utilizate cu două scopuri: de a reduce spaţiul de căutare, şi de a dirija achiziţia de noi date (testarea de noi manifestări pe parcursul inferenţei).

Dacă **S(a)** reprezintă faptul că s-a observat valoarea **a** pentru nodul S al reţelei, $\mathbf{NC_{S(a)}}$ este considerată o *condiţie necesară* pentru **S(a**) dacă este adevărată în orice explicaţie a lui **S(a)**. Cu alte cuvinte, dacă o explicaţie susţine **S(a)**, susţine şi toate condiţiile sale necesare $\mathbf{NC_{S(a)}}$. Aceste condiţii sunt utile în evitarea explicaţiilor inconsistente cu datele observate: o stare **S(a)** poate fi eliminată dacă una din condiţiile sale necesare este inconsistentă cu datele.

Condiţiile suficiente sunt utile pentru evitarea explicaţiilor redundante: dacă o condiţie $\mathbf{SC_{S(a)}}$ suficientă pentru **S(a)** este susţinută de o anumită explicaţie, atunci şi **S(a)** este susţinută de explicaţia respectivă -aşadar dacă una din cauzele unei stări este condiţie suficientă pentru aceasta, celelate cauze nu mai trebuie testate.

Proprietăţi.

- $\mathbf{NC_{S(a)}}$ e condiţie necesară pentru **S(a)** ddacă $T_C \models \mathbf{S(a) \rightarrow NC_{S(a)}}$;
- $\mathbf{SC_{S(a)}}$ e condiţie suficientă pentru **S(a)** ddacă $T_C \models \mathbf{SC_{S(a)} \rightarrow S(a)}$.

Optimizările pe care aceste elemente observabile le aduc algoritmului sunt:

- se pot elimina dintre observațiile ce trebuie explicate efectele colaterale ale acestora=condițiile lor necesare (care au aceleaşi explicații);

- în momentul întâlnirii unui punct de ramificare în timpul căutării înapoi, cauzele ale căror condiții necesare sunt inconsistente cu datele se pot elimina; (atomii necunoscuți care apar în condițiile necesare sunt subiect de testare);

- dacă într-un punct de ramificare o cauză este condiție suficientă -celelalte cauze se elimină din spațiul de căutare (pentru mai multe detalii şi algoritmul complet -a se vedea [Dup94]).

Strategia optimă de discriminare este de a alege pentru testare o dată care apare cu valori diferite în condiții diferite. Multe din planurile de testare sunt, de fapt, extinderi ale unui principiu de bază -criteriul entropic de recunoaştere a formelor([Gui77]), prezentat pe scurt în continuare.

Fie e_1 ,..., e_n entitățile de diagnostic, caracterizate de o mulțime de caracteristici (teste) $f_1,...,f_m$ ce pot lua valorile $\{a_1^1,...,a_{n_1}^1\}$,..., şi, respectiv $\{a_1^m,...,a_{n_m}^m\}$ şi entitățile se exprimă în termenii acestor valori astfel:

$$e_1 = \{a_{k_{11}}^1,...,a_{k_{m1}}^m\},...,e_n = \{a_{k_{1n}}^1,...,a_{k_{mn}}^m\}.$$

Se alege pentru testare acel f_i de entropie maximă:

$$H_{f_i} = -\sum_{j=1}^{n_i} p_j . \log p_j \ , \quad (3.65)$$

unde p_j este probabilitatea (i.e.frecvența relativă) a valorii a_j^i în definiția setului de entități. (Informația maximă se obține înlăturând cel mai mare grad de incertitudine -entropia fiind o măsura a incertitudinii şi implicit, a informației). Se ştie că entropia este maximizată de distribuția de posibilitate uniformă; aşadar principiul folosit este intuitiv: este mai discriminantă caracteristica ce are cât mai multe valori distincte în entități distincte- adică repartizate mai uniform. Dacă în urma testării lui f_i se obține valoarea concretă a_k^i, vor fi eliminate e_j cu $a_{kj}^i \neq a_k^i$. Se reiau calculele până rămâne o singură entitate e.

Metoda este foarte eficientă şi minimizează înălțimea arborelui de decizie, dar are şi serioase limitări, şi anume următoarele presupuneri:

- exact un e este prezent din mulțimea de manifestări;

- caracteristicile considerate definesc exhaustiv mulțimea de entități (alte aplicații în diagnoză ale arborilor de decizie vor fi amintite mai jos).

O euristică legată de punctul 3 ("Ce înseamnă explicație *mai bună*?") folosită în [Dup94] constă în structurarea cunoştinţelor pe nivele diferite de abstractizare .

Definiție. Ierarhie explanatorie.([Dup94]) Se numeşte ierarhie explanatorie o structură $<T_E, T_A>$, unde elementele lui T_E sunt de forma:

$$P_1 \wedge ... \wedge P_n \rightarrow Q \quad (3.66)$$

(axiome explanatorii), iar elementele lui T_A: $P \rightarrow Q$ sunt axiomele de abstractizare (P este mai specific decât Q). Se presupune în plus că nu există dependențe ciclice în model.

Această ierarhie introduce o relație de preferință între explicații, bazată pe dimensiunea de abstractizare, care generalizează preferința pentru explicațiile minimale. Pe baza acestei ierarhii, autorul defineşte algoritmi care exploatează criteriul euristic al "explicațiilor leneşe": ideea este de a concentra inițial raționamentul pentru explicarea unei observații la nivelul mai înalt (=mai abstract, mai general) şi trecerea la concepte mai detaliate (nivele inferioare) numai dacă explicațiile candidat găsite la acest nivel sunt inconsistente. Ideea seamănă cu raționamentul implicit, în care la apariția inconsistențelor se folosesc cazurile excepție.

***

Pentru abducție s-au folosit şi tehnici de demonstrare directă ([Kon92, McIlr98]), de exemplu *tehnicile rezolutive de găsire a consecințelor unei teorii*. Această abordare nu respectă însă semantica unei probleme de diagnoză. Ideea folosită este de a rescrie relația $\Sigma \wedge E \vdash O$ (Σ fiind teoria domeniului, E explicația, O observațiile), prin aplicarea contrapozitivei, drept $\Sigma \wedge \neg O \vdash \neg E$. Versiunea duală (indirectă) a abducției, ce se obține astfel este următoarea.

Definiție.(Abducția indirectă) E este o explicație pentru O ddacă:

- $E \subseteq H$;
- $\Sigma \cup \neg O \not\models \neg E$;
- Din Σ nu se deduce $\neg E$

unde $\neg E$ reprezintă conjuncția literalilor negați.

O idee folositoare ce decurge din versiunea duală este că o explicație pentru O se poate genera căutând teoremele lui $\Sigma \cup \neg O$, care nu sunt teoreme doar ale lui Σ .

Revenind însă la problema noastră de diagnoză, lipsa anumitor observații nu înseamnă obligatoriu lipsa unui anume diagnostic (acesta poate fi prezent prin alte manifestări ale sale). Un dezavantaj în plus îl constituie faptul că rezoluția este completă pentru respingere (i.e. completă pentru găsirea demonstrațiilor), dar nu este completă pentru găsirea consecințelor (nu e deductiv completă). De fapt, teoretic există metode (versiuni

ale rezoluției) complete, dar care sunt extrem de ineficiente din cauza unui spațiu de căutare mult prea mare, în timp ce metodele eficiente sunt incomplete [Kon92].

3.4.2.Metode de diagnoză specifice sistemelor dinamice

3.4.2.1.Introducere

În contextul sistemelor dinamice devine evidentă necesitatea integrării timpului în modelul de diagnoză. Timpul este modelat explicit în cadrul sistemelor continue şi implicit în sistemele cu evenimente discrete.

Metodele de diagnoză a sistemelor depind atât de tipul sistemului cât şi de tipurile posibile de defecte ce îl pot afecta. Introducem în continuare câteva definiții elementare ale defectelor aşa cum au fost impuse de către comunitatea IFAC.

Definiție. Se numeşte *defect* (fault) o modificare neaşteptată a unei funcții a sistemului. Aceasta diferă de *căderea* (failure) sistemului, în sensul că cea din urmă presupune încetarea completă a funcționării.

Clasificare. Defectele pot fi de trei tipuri:

- incipiente: defecte de proporții mici cu o evoluție continuă;
- abrupte: efectele defectelor abrupte sunt mai serioase; ele pot aduce sistemul aproape de limita de comportament acceptabil;
- intermitente: efectul lor asupra sistemului rămâne ascuns pe anumite perioade de timp; depistarea cât mai timpurie previne apariția unor efecte serioase şi, eventual, căderea sistemului.

Paşii importanți ai procesului de diagnoză sunt:

- detectarea;
- izolarea ;
- identificarea .

(FDI=Fault Detection and Isolation) *O schemă robustă FDI* trebuie să asigure atât sensitivitate satisfăcătoare la defecte cât şi invarianța la zgomot şi incertitudinile de modelare [PattChe97] (aceste două cerințe aflându-se în conflict).

"*Diagnoza pe bază de model* se defineşte ca determinarea defectelor unui sistem prin compararea măsurătorilor disponibile la un anumit moment cu informațiile a priori date de modelul analitic/matematic al sistemului şi generarea şi analiza de cantități reziduale. Un reziduu este un indicator de defect ce reflectă condiția degradată a sistemului." [Patt94]

Din definiție, se observă că un element central în diagnoza pe bază de model este generarea de reziduuri [Nyb99]. Cele două stadii principale ale diagnozei defectelor folosind reziduuri sunt reprezentate în Figura 3.11 (conform [Fran96]). Problema

principală a diagnozei pe bază de model stă în faptul că erorile de modelare şi perturbaţiile afectează sistemul. Incertitudinile de modelare afectează abilitatea sistemului de a recunoaşte defecte incipiente, ducând la alarme false sau defecte nedetectate. Trebuie, astfel, realizat un compromis între senzitivitatea la erori şi robusteţea la zgomot.

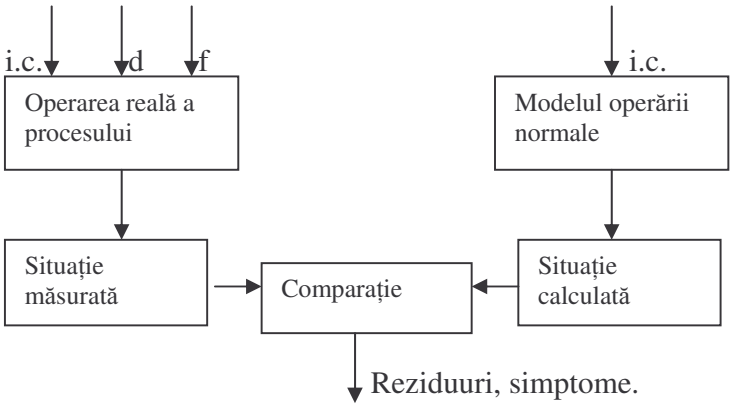


Figura 3.11.Principiul diagnozei defectelor pe bază de model(i.c. = input cunoscut)

Aşadar, diagnoza sistemelor dinamice constă în două etape principale distincte: detectarea defectelor şi diagnoza defectelor (Figura 3.12).

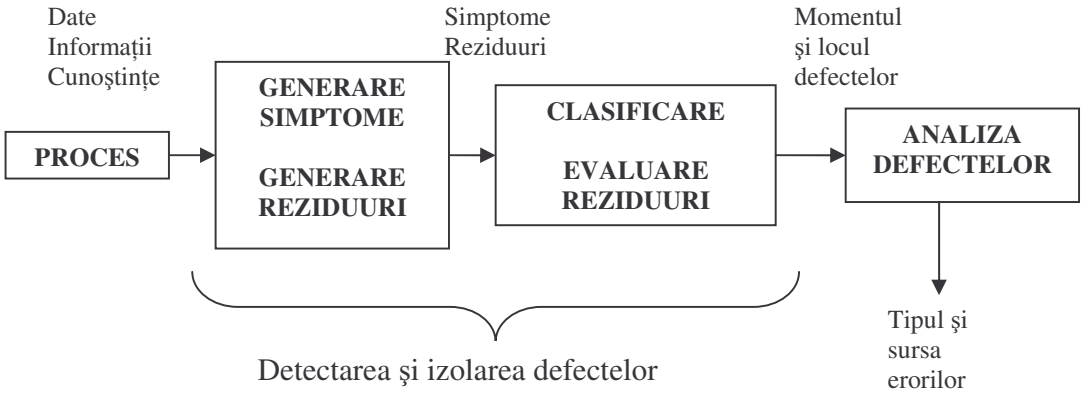


Figura 3.12.[Fran96].Reprezentarea schematică a procedurii de diagnoză a defectelor

Detectarea defectelor constă în generarea euristică sau analitică a simptomelor (bazate pe reziduuri).

Diagnoza defectelor determină tipul, locaţia, momentul apariţiei şi amplitudinea defectului, prin observarea simptomelor euristice sau analitice [Ise97]. În cadrul acestei etape, dacă nu sunt disponibile relaţii cauzale profunde între defecte şi simptome, se folosesc metode de clasificare ce mapează vectorul de simptome într-un vector de defecte (de exemplu, clasificare statistica sau geometrică, reţele neurale, clusterizare fuzzy). Dacă dimpotrivă relaţiile respective sunt accesibile, (spre exemplu, sub forma unor relaţii

cauzale), se pot aplica strategii de raţionament pentru diagnoză (înlănţuire înainte/înapoi, raţionament aproximativ probabilistic pentru reţele bayesiene şi probabilităţi condiţionale sau raţionament posibilist cu ajutorul logicii fuzzy [Ise97]). Capitolul IX prezintă o abordare originală a acestei etape (de diagnoză a defectelor sistemelor dinamice), prin adaptarea sistemului hibrid DiaMed.

3.4.2.2.Diagnoza sistemelor pe bază de stare

În diagnoza pe bază de stare se raţionează asupra stărilor sistemului privite izolat, fără a ţine cont de tranziţiile între ele. De obicei, defectele sunt asignări de stări componentelor, consistente cu observaţiile.

I.Diagnoza sistemelor folosind modelul analitic
Diagnoza sistemelor continue folosind modelul analitic este potrivită pentru depistarea defectelor incipiente. Există două direcţii principale de diagnoză de acest tip: cea care foloseşte redundanţa analitică –utilizată în diagnoza pe bază de model şi redundanţa hardware (care este costisitoare).

Metode de generare a reziduurilor
a)Sisteme observator
Ideea principală a generării de reziduuri pe bază de observatori stă în reconstruirea ieşirilor sistemului şi utilizarea erorii de estimare ca reziduu. Spre deosebire de observatorii de stare folosiţi pentru feedback-ul stării în cazul măsurătorilor incomplete, un observator pentru diagnoză este un observator al ieşirilor şi poate fi construit direct în domeniul frecvenţelor, fără a folosi modelarea în spaţiul stărilor [Fran96].

b)Estimare parametrică
Atât pentru sistemele liniare cât şi pentru cele neliniare, se pot utiliza în FDI pentru a genera reziduuri şi tehnici de estimare a parametrilor, presupunând că defectele se reflectă la nivelul parametrilor fizici ai sistemului. Parametrii estimaţi folosind intrările şi ieşirile actuale ale sistemului se compară cu parametrii obţinuţi în condiţiile de funcţionare normală, iar prezenţa unui defect este semnalată de existenţa unor diferenţe semnificative între cele două mulţimi.

II.Diagnoza sistemelor cu stări discrete pilotate de evenimente
Sistemele cu evenimente discrete sunt o clasă aparte de sisteme neliniare, ce necesită un formalism matematic propriu diferit de cel al ecuaţiilor diferenţiale sau al ecuaţiilor cu diferenţe (folosit de I).

Definiţie[Lar99]. Un sistem cu evenimente discrete (SED) este un sistem dinamic care evoluează în concordanţă cu apariţia bruscă, la intervale de timp posibil neregulate şi necunoscute, a unor evenimente fizice.

Aşadar:

- Spaţiul stărilor este o mulţime discretă; (X)
- Mecanismul de tranziţie al stărilor este pilotat de evenimente cu apariţie asincronă.(E)

Exemplele de aplicaţii din lumea reală variază de la servirea clienţilor într-un sistem de aşteptare până la procesarea joburilor într-un sistem de calcul. Modelele de SED netemporizate se pot aborda folosind formalismele reţelelor Petri sau al automatelor [Cass93] (ambele metode reprezintă explicit funcţia de tranziţie a unui SED, manevrând evenimente în concordanţă cu anumite reguli), sau modelul relaţional [Ger95] -util sistemelor cu defecte abrupte dar necatastrofice (care nu opresc sistemul din funcţionare). În ultimul caz, modelul sistemului este dat de o relaţie ce defineşte ieşirea şi noua stare a sistemului în funcţie de intrare şi de starea anterioară, această reprezentare fiind foarte aproape de diagnoza pe bază de model din inteligenţa artificială, întrucât se bazează pe aceleaşi noţiuni de explicaţie şi de diagnostic minimal (diferă însă modul de calcul al diagnosticului). Însă unul din principalele dezavantaje ale acestui model este faptul că în analiza proprietăţii de diagnozabilitate [Lar99] se consideră, nerealist, că defectarea unei componente nu afectează evoluţia stării unei alte componente. În plus, în ciuda încercărilor de îmbunătăţire a spaţiului de memorie necesar şi a timpului de calcul, nu poate fi considerată, deocamdată, o metodă eficientă.

3.4.2.3.Diagnoza pe bază de simulare

Raţionamentul ce utilizează relaţiile de cauzalitate din sistem şi simularea pot oferi rezultate de diagnostic mai precise faţă de diagnoza pe bază de stare. Există în plus avantajul că se poate lucra cu un model incomplet sau cu o versiune simplificată a unui model foarte complex, în lipsa unui model analitic exact (cum este necesar în metodele convenţionale).

Metode de simulare calitativă

Ideea de bază a acestei metode stă în cuantizarea valorilor variabilelor, adică partiţionarea unui spaţiu într-un număr finit de mulţimi disjuncte (de exemplu, intervale, dacă spaţiul este o dreaptă), în locul valorilor precise folosindu-se submulţimile obţinute [Kui86].

Simularea calitativă este un caz particular al clasei de probleme de satisfacere a constrângerilor temporale extinse (temporally extended Constraint Satisfaction Problem) [ClaKui98], şi o direcţie majoră de cercetare în reprezentarea modelelor de profunzime (“deep models”). Pornind de la o stare iniţială a sistemului (definită prin valorile variabilelor, cuprinse într-o mulţime finită de valori graniţă =“landmarks”) şi folosind o funcţie de tranziţie dată de constrângerile asupra comportamentului sistemului (constrângerile de continuitate ale unor funcţii devenind constrângeri temporale ale CSP: valoarea la momentul/intervalul următor trebuie să fie “apropiată” de valoarea actuală) se construieşte un graf în care traiectoriile –secvenţe posibile de stări- reprezintă de fapt soluţii ale CSP. Cel mai cunoscut algoritm de simulare calitativă este QSIM [Kui86].

Sistemul de raționament cauzal al lui Kuipers conține o mulțime de ecuații ale constrângerilor, ce descriu relațiile structurale relevante din sistem. Simularea calitativă se desfăşoară prin generarea stărilor succesor posibile şi filtrarea lor pe baza mulțimii de constrângeri ce defineşte modelul. Descrierea comportamentului sistemului pe bază de simulare calitativă este apoi folosită pentru explicarea observațiilor (constituind astfel o soluție naturală a unei probleme de diagnoză). Abstractizarea modelului se obține prin exprimarea cantităților cu ajutorul unei mulțimi liniare de valori marcate ("landmarks").

Definim în continuare *diagnoza pe bază de simulare* a observațiilor OBS relativ la starea calitativă S în contextul modelării calitative. În loc să impunem cerința ca S să fie consistentă cu observațiile şi cu defectul F (ca în diagnoza pe bază de stare), se consideră necesar ca S să fie accesibilă dintr-o stare inițială S_0 consistentă cu comportamentul normal şi în care deviațiile sunt 0, iar stările intermediare între S_0 şi S se obțin în conformitate cu operatorul "succesor" al modelării calitative, ținând cont de faptul că între S_0 şi S_1 a fost "injectat" defectul F. (Putem folosi constrângeri cauzale pentru a modela comportamentul discontinuu, ca urmare a injectării unui defect [PanDup00]). Se înlătură prin această abordare unele dintre efectele contraintuitive ale diagnozei pe bază de stare (de exemplu, starea normală a componentelor poate fi consistentă cu observațiile, deşi acestea conțin manifestări de defect.)

Diagnoza sistemelor cu evenimente discrete folosind limbaje formale
Este o abordare pe bază de simulare potrivită atât pentru SED cât şi pentru sistemele continue modelate prin ecuații diferențiale. Unul dintre avantajele sale majore constă în faptul că nu necesită modelarea în profunzime a sistemului de diagnosticat, fiind astfel utilă în diagnosticarea sistemelor complexe, de proporții mari. Metoda construieşte un model discret al sistemului, şi un model al componentei de diagnosticare, care include cunoştințe atât despre comportarea normală cât şi despre cea de defect. Atât modelul sistemului cât şi elementul de diagnosticare sunt reprezentate ca automate finite deterministe. Evenimentele diagnoser-ului se clasifică în observabile sau neobservabile, iar defectele fac parte din evenimentele neobservabile ale sistemului, şi problema constă, de fapt, în detectarea lor pe baza unor "urme" parțiale generate de automat (al cărui limbaj generat exprimă comportarea sistemului) [LunSchro01, SamSen96].

3.5.Concluzii şi contribuții

Capitolul propune o privire de ansamblu asupra unei mari părți a metodelor de diagnoză cunoscute. O contribuție importantă a sa constă în încercarea de clasificare şi sistematizare pe care o propune, pentru a îngloba într-un cadru coerent abordări distincte ale aceleiaşi probleme. Sunt confruntate, în principal, diagnoza pe bază de model şi diagnoza prin metode statistice, fiecare dependentă de accesul la un anumit tip de informație despre sistemul studiat. Prezentarea este ilustrată de exemple clasice de implementare din literatură, şi oferă o imagine clară despre formalismul de bază al fiecărei metode, cu avantajele şi dezavantajele sale. Scopul principal a fost acela de a pune bazele unei justificări a utilității combinării celor două paradigme într-un model hibrid, justificare ce se va finaliza în capitolul următor.

IV.Sisteme inteligente hibride de diagnoză medicală

4.1.Introducere

Capitolul de față aduce în prim plan arhitecturi inteligente hibride aplicate sau potențial aplicabile diagnozei medicale. În unele subsecțiuni detalierea s-a justificat nu doar prin rolul său ilustrativ, dar şi prin faptul că am dorit să relevăm potențialul (încă nevalorificat corespunzător) al tehnicii respective pentru diagnoza medicală.

După ce în Secțiunea 4.2. prezentăm definițiile şi caracteristicile tipurilor de hibridizare în general, în Secțiunea 4.3. ne ocupăm mai în detaliu de tehnici inteligente hibride în diagnoza medicală. Hibridizarea neuro-simbolică este ilustrată de arhitectura KBANN (Knowledge-Based Adaptive Neural Networks: rețele neurale bazate pe cunoştințe) utilizată frecvent în diagnoza medicală, şi de rețelele neurale comitet pe care le-am considerat o generalizare promițătoare a KBANN (această perspectivă fiind o observație originală). Hibridizarea neuro-fuzzy este deasemenea prezentă prin sistemele ANFIS (Adaptive Neural Fuzzy Inference Systems), NFC(Neuro-Fuzzy Clasifier), Perceptronul Fuzzy Multistrat, sistemele neuro-fuzzy fuzionale ce folosesc neuroni fuzzy logici. Acestea din urmă oferă o perspectivă asupra posibilităților de integrare a logicii formale în structura rețelelor neurale. Toate abordările enumerate sunt adecvate diagnozei privită ca problemă de clasificare.

Secțiunea 4.4.1. se ocupă de prezentarea sistemului de diagnoză CHECK, propus de Torasso şi Console, pe care l-am văzut ca o etapă intermediară între CASNET şi sistemul original DiaMed. În 4.4.2. prezentăm CHECK şi CASNET cu evidențierea avantajelor primului, dar şi a problemelor ce pot fi încă îmbunătățite, iar în 4.5. introducem sistemul original DiaMed, prezentând succint avantajele sale în contextul respectiv.

4.2.Tehnici inteligente hibride

Sintetizăm în cadrul tabelului 4.1. [PalNea01] principalele avantaje şi dezavantaje ale tehnicilor inteligente tratate în capitolul anterior.

Hibridizarea urmăreşte compensarea dezavantajelor şi îmbunătățirea performanțelor prin cooperarea tehnicilor inteligente în cadrul rezolvării aceleiaşi probleme. Există patru tipuri de sisteme hibride [PalNea01]:

1. **Sisteme fuzionale**: modul de reprezentare şi prelucrare a informațiilor caracteristic unei tehnici se contopeşte în modul de reprezentare şi prelucrare al altei tehnici (de exemplu, sistemele cu rețele neurale pentru reprezentarea şi procesarea informațiilor simbolice caracteristice sistemelor simbolice bazate pe cunoştințe sau sistemelor fuzzy).

TEHNICA INTELIGENTĂ	AVANTAJE	DEZAVANTAJE
Sisteme simbolice bazate pe cunoştinţe	• Sunt transparente (furnizează explicaţii clare ale deciziilor luate); • Separă cunoştinţele de sistemul ce le exploatează; • Înglobează cu uşurinţă cunoştinţe structurate, de calitate şi precise, prin reprezentarea declarativă.	• Dificultăţi la achiziţionarea de cunoştinţe şi la adaptare; • Număr mare de reguli; • Raţionament lent; • Dificultăţi în operarea cu informaţii imprecise şi incomplete; • Performanţele nu se îmbunătăţesc semnificativ la completarea cunoştinţelor; • Nu generalizează.
Reţele neurale	• Prelucrare paralelă; • Răspuns rapid; • Învăţare, adaptare rapidă; • Capacitate de generalizare şi procesare a informaţiilor incomplete.	• Incapacitatea de a explica rezultatul; • Riscul blocării în minimele locale.
Sisteme fuzzy	• Modelează imprecizia şi incertitudinea; • Aproximează funcţii nelineare; • Implementare ieftină şi comportament robust; • Nu necesită un model foarte precis; • Viteză de inferenţă ridicată;	• Probleme cu învăţarea şi adaptarea; • Probleme în definirea termenilor de apartenenţă, a numărului de reguli, a parametrilor consecinţelor regulilor;
Algoritmi genetici	• Nu necesită un model matematic (sunt "orbi": fac puţine presupuneri asupra domeniului problemei); • Nu au probleme cu minimele locale (în principal datorită faptului că explorează spaţiul de căutare în mai multe puncte simultan, prin intermediul mai multor populaţii de soluţii potenţiale); • Explorează inteligent spaţiul de căutare; • Potriviţi pentru spaţiile de căutare mari; • Au arii largi de aplicabilitate deoarece operează asupra unor codificări ale variabilelor de decizie; • Adaptare lentă.	• Lenţi; • Dificultăţi în construirea operatorilor potriviţi.

Tabel 3.1. Comparaţie a tehnicilor inteligente

2. **Sisteme de transformare**: transformă o formă de reprezentare a cunoştinţelor în altă formă (spre exemplu, sistemele de extragere de reguli simbolice din reţelele neurale, reguli ce sunt apoi folosite de un sistem simbolic).

3. **Sisteme combinative**: păstrează identitatea fiecărei metodologii inteligente şi presupun utilizarea modulară a două sau mai multe tehnici inteligente pentru a rezolva o problemă.

4. **Sisteme asociative**: combină mai multe din tipurile de hibridizare de mai sus.

Hibridizare neuro-simbolică.
Integrarea neuro-simbolică îşi găseşte justificarea în primul rând la nivel cognitiv, ideea fiind de a identifica în inteligența umană operațiile de nivel înalt (legate de partea de raționament şi care sunt cel mai bine modelate de o abordare simbolică), şi operațiile de nivel inferior (legate de percepție şi mai natural de abordat într-un cadru neuronal). Întrucât această hibridizare reconciliază ştiința experților cu estimările statistice din date, este foarte potrivită aplicațiilor complexe ale lumii reale.

În Secțiunea 4.3.1, am ales pentru exemplificarea acestei paradigme sistemul hibrid KBANN (Knowledge-Based Artificial Neural Networks) utilizat în biomedicină [San99], şi rețelele neurale comitet, care ni s-au părut o generalizare adecvată şi promițătoare a acestuia, prin prisma aplicațiilor de diagnoză medicală şi nu numai.

Hibridizare neuro-fuzzy.
Rețelele neurale şi logica fuzzy se completează adesea în chip fericit în cadrul sistemelor inteligente: dacă rețelele neurale pot prelucra cantități mari de date primite de la senzori (în probleme de conducere sau recunoaştere a formelor), logica fuzzy prelucrează aceste date de nivel inferior într-un cadru structurat de nivel înalt. Există în principal două direcții importante în acest tip de hibridizare [DumiBuiu00]:

- Fuzificarea arhitecturilor convenționale de rețele neuronale (vezi Secțiunea 4.3.2);
- Folosirea rețelelor neurale ca unelte în cadrul modelelor fuzzy (de exemplu, în furnizarea unor aproximații de calitate ale funcțiilor de apartenență).

Hibridizare geno-fuzzy.
Combinarea celor două tehnici se face în baza anumitor asemănări şi deosebiri. Între primele, menționăm faptul că nici una din tehnici nu necesită un model precis, utilizând o reprezentare distribuită a cunoştințelor (în populațiile de cromozomi sau, respectiv, în mulțimile şi regulile fuzzy), sunt tolerante la defecte (ca urmare a reprezentării distribuite) şi pot modela incertitudinea. Însă în ceea ce priveşte manevrarea cunoştințelor distribuite, dacă algoritmii genetici efectuează o procesare paralelă şi *probabilistă*, logica fuzzy are un mecanism de inferență *posibilist*. În plus, dacă algoritmii genetici sunt prin natura lor adaptivi, sistemele de inferență fuzzy clasice nu au această proprietate, ele fiind în schimb mai avantajoase din punctul de vedere al vitezei de inferență [DumiBuiu00].

Între exemplele acestui tip de hibridizare menționăm modelul propus de Karr [Karr91], în care un algoritm genetic este folosit pentru învățarea funcțiilor de apartenență şi numărului de reguli unui sistem fuzzy de tip Takagi-Sugeno. Funcțiile de apartenență sunt corespunzător parametrizate şi codificate în cromozomii algoritmului, iar evaluarea populațiilor de soluții se face dependent de aplicația avută în vedere.

Hibridizare neuro-genetică.
Această hibridizare pleacă de le ideea că este benefic ca evoluția şi învățarea să poată lucra împreună sinergetic. Există două tipuri de abordări distincte: combinațiile suport (în care tehnicile sunt folosite secvențial) şi combinațiile colaborative (unde tehnicile sunt rulate simultan). În general, rețelele neurale sunt asistate de către algoritmul genetic în selectarea caracteristicilor (intrărilor), a topologiei sau a regulii de învățare folosite.

Am justificat în secțiunea dedicată algoritmilor genetici în diagnoza medicală de ce nu este oportună utilizarea lor în aplicațiile acestui domeniu. Ne vom ocupa în continuare de exemple de hibridizare neuro-simbolică şi neuro-fuzzy, care au fost aplicate deja sau care am considerat că ar fi potrivite diagnozei medicale.

Tehnicile inteligente şi logica formală
Un tip aparte de hibridizare îl constituie utilizarea unei metode inteligente (simbolică, fuzzy, neurală, genetică) în sprijinul unui proces de raționament bazat pe logică (formală sau informală). Un exemplu notabil în acest sens îl constituie sistemul CHECK (sprijin simbolic cu reguli pentru logica formală), care a fost şi punctul de plecare al genului de hibridizare combinativă folosit de sistemul original DiaMed (sprijin fuzzy pentru logica informală) (Secțiunea 4.4).

4.3.Tehnici inteligente hibride în diagnoza medicală

Am selectat pentru secțiunile următoare câteva sisteme hibride care ar fi potrivite pentru diagnoza medicală, în virtutea argumentelor prezentate până în acest punct. Exemplele de arhitecturi ce urmează au fost sau pot fi aplicate cu succes în acest domeniu.

4.3.1.Hibridizare neuro-simbolică

4.3.1.1.Rețele neurale bazate pe cunoştințe.

Rețelele neurale bazate pe cunoştințe (KBANN: Knowledge-Based Artificial Neural Networks) sunt o metodologie hibridă neuro-simbolică fuzională ce poate procesa mulțimi de date complexe, de dimensiuni reduse, şi cu o repartizare neuniformă, în scopul clasificării. Modulul simbolic cuprinde teoria domeniului sub forma unei mulțimi de reguli structurată ierarhic, iar modulul conexionist asociază fiecărui concept din teoria domeniului câte un nod al rețelei neurale, prin traducerea structurii cunoştințelor din reguli în topologia rețelei.

Algoritmul de învățare KBANN

Fiind date:

- teorie a domeniului aproximativ corectă, cu o mulțime de reguli şi caracteristici;

- mulțime de exemple;

Realizează:
- Mapează structura cunoştinţelor în structura unei rețele neurale (Figura 4.2);
- Antrenează rețelele bazate pe cunoştinţe folosind mulțimea de exemple.

Rezultat:
După antrenare, rețeaua poate clasifica exemple din afara setului de antrenare.

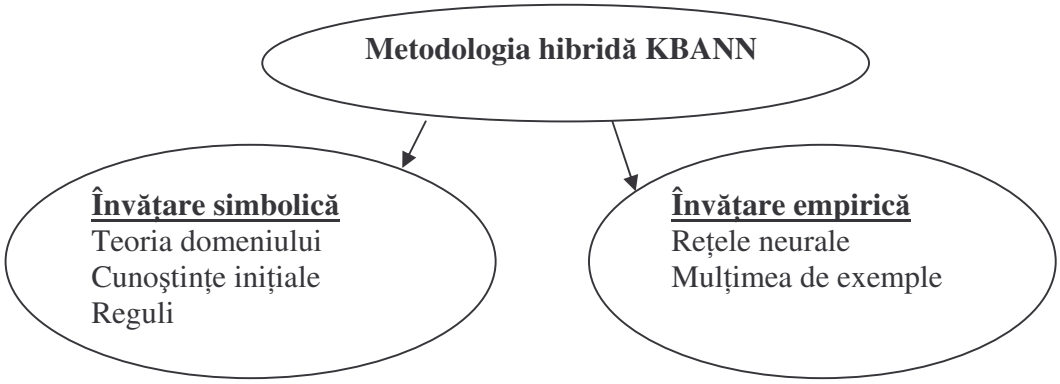


Figura 4.1. Cele două componente ale metodologiei KBANN

Sistemul a fost aplicat în clasificarea de țesuturi, cu rezultate satisfăcătoare în depistarea cancerului mamar.

4.3.1.2.Rețele neurale comitet

Rețelele comitet sunt sisteme hibride fuzionale ce cuprind rețele neurale organizate după modelul unui arbore de decizie, şi conțin rețele neurale cu învățare supervizată organizate după principiul "divide et impera". Învățarea este distribuită între experți (de obicei rețele neurale de tip perceptron) care partiționează spațiul de intrare. Maşina comitet este o combinație de experți, structurată după criterii ce vor deveni clare în continuare. Ca şi perceptronul multistrat, rețelele comitet sunt aproximatori universali. Cuprind două mari subclase de arhitecturi: modelele statice şi cele dinamice. Acestea din urmă vor fi succint prezentate în cele ce urmează.

Modelele dinamice presupun că "integrarea cunoştinţelor asimilate de experți se face sub acțiunea semnalului de intrare" şi cuprind *mixtura de experți* şi *mixtura de experți ierarhică*, exemple ce ilustrează foarte bine şi proprietatea de modularitate.

Modularitate[Hay98]. O rețea neurală este *modulară* atunci când calculul efectuat de rețea poate fi descompus în două sau mai multe module / subsisteme ce operează pe

intrări diferite fără să comunice între ele. Ieşirile modulelor sunt mediate de o unitate integratoare fără feedback care:
1) decide cum trebuie combinate ieşirile modulelor pentru a forma ieşirea finală a sistemului, şi
2) decide pentru fiecare exemplu de antrenare de către ce modul trebuie învățat.

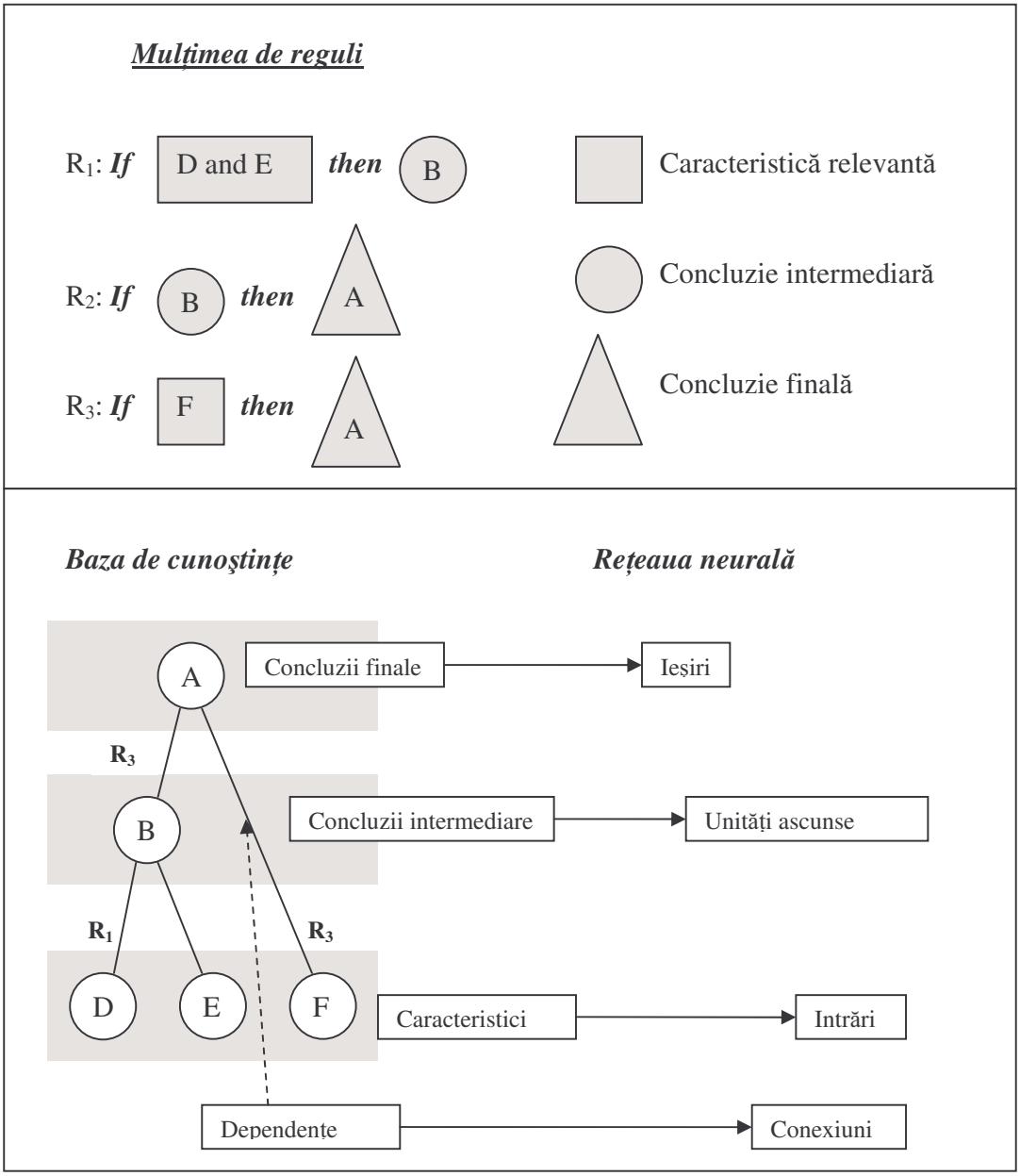


Figura 4.2. Procedura de transformare din reguli în rețea neurală în KBANN

Modularizarea rețelelor neurale este o metodă de inserare a cunoştinţelor explicite într-o rețea neurală, prin descompunerea unei probleme complexe în subprobleme mai simple. Avantajele modularizării se reflectă în creşterea vitezei de învățare, a abilităților de generalizare şi reprezentare/interpretare, şi proprietății de a satisface condițiile impuse de limitările hardware [NeaPal01]. Un avantaj în plus al rețelelor modulare este faptul că acestea pot fi o bază naturală de reprezentare a sistemelor de inferență fuzzy, constituind astfel un pas înainte către structurile hibride. Spre exemplu, un sistem de reguli fuzzy de tip Sugeno poate fi descris ca un anumit tip de rețea modulară: fiecărei reguli îi corespunde o rețea expert, iar rețeaua poartă defineşte contribuția fiecărei reguli la ieşirea finală.

Învățarea în rețelele modulare dinamice de tip comitet combină paradigma învățării supervizate (experții îşi combină ieşirile individuale pentru a modela răspunsul dorit) cu autoorganizarea (experții se autoorganizează în partiționarea optimă a spațiului de intrare).

Modelul mixturii asociative gaussiene

Se pleacă de la o problemă de regresie, pe baza datelor de antrenare $\{(x_i, d_i)\}_{i=1}^{N}$, unde regresorul (cauza) x produce efectul dat de variabila aleatoare D ale cărei realizări se notează d. Ideea principală este de a partiționa spațiul de intrare astfel încât problema redusă pe subspații să fie liniară (avem deci, de fapt, o problemă de alegere a modelului). După cum observa autorul în [NeaPal01], rețelele expert procesează funcții distincte ce descriu comportarea sistemului pentru regiuni diferite din spațiul intrărilor.

Elementele integratoare au rolul de a defini regiunile de decizie între aceste subspații. Concret, pentru generarea lui d se apelează la următorul model probabilist [Hay98]:

1. vectorul de intrare x este ales aleator conform unei distribuții apriori;
2. o anumită regulă (k) este selectată conform distribuției condiționale $P(k|x,a^{(0)})$, fiind dați x şi parametrul $a^{(0)}$;
3. pentru regula k , k=1,…,K răspunsul d al modelului este liniar în x cu o eroare aditivă ε_k luată ca o variabilă normală standard: $E[\varepsilon_k]=0$ şi $var[\varepsilon_k]=1$ pentru toți k.

Răspunsul D al modelului se scrie:

$$P(D = d \mid x, \theta^{(0)}) = \sum_{k=1}^{K} P(D = d \mid x, w_k^{(0)}) P(k \mid x, a^{(0)}) \quad (4.1)$$

unde $\theta^{(0)}$ este vectorul de parametri al modelului generativ şi este format din $a^{(0)}$ şi $\{w_k^{(0)}\}_{k=1}^{K}$ (indicele superior 0 face distincția între parametrii modelului generativ şi cei ai experților, introduşi în continuare.)

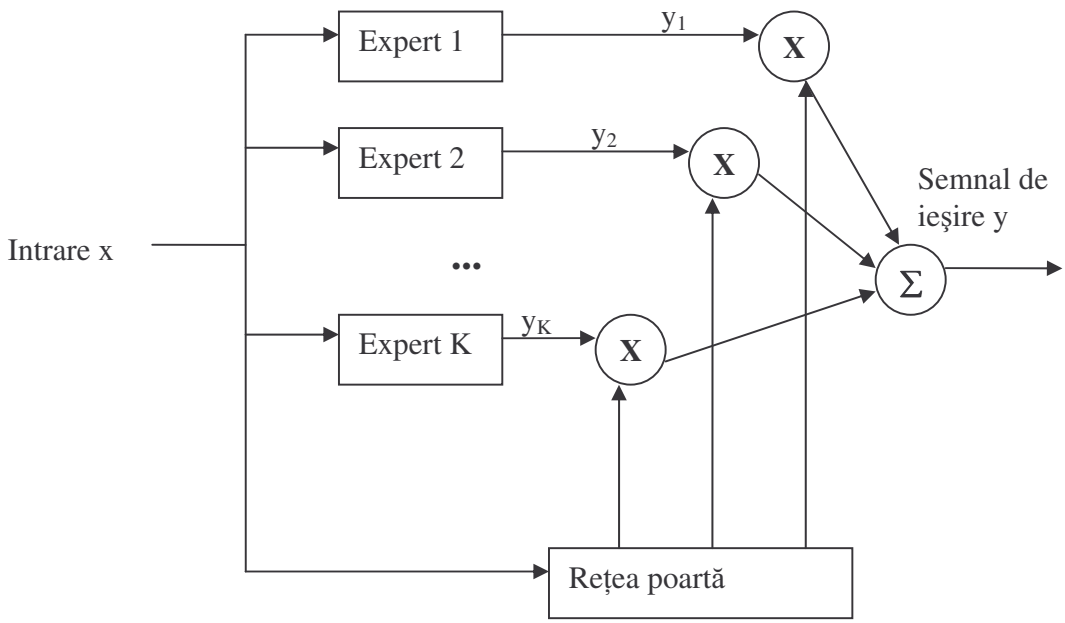


Figura. 4.3.Diagrama bloc a mixturii de experți

Fiecare expert este un filtru liniar şi produce o ieşire de forma $y_k = w_k^T x$, k=1,…,K, unde w_k este vectorul de ponderi al expertului k. Rețeaua poartă este formată din K neuroni neliniari (câte unul pentru fiecare expert) cu funcția de activare *softmax*: (filtru + neliniaritate de tip softmax).

$$g_k = \frac{\exp(u_k)}{\sum_{j=1}^{K} \exp(u_j)}, \text{ k=1,…,K, (4.2)}$$

unde u_k este produsul scalar al lui x cu vectorul de ponderi al rețelei poartă a_k : $u_k = a_k^T x$, k=1,…,K. Rețelele expert se pot modela şi ca rețele multistrat/ recurente.

Numerele g_k pot fi privite ca probabilități ce ponderează răspunsul fiecărui expert în cadrul răspunsului total $y = \sum_{k=1}^{K} g_k y_k$ al mixturii de experți, ele reprezintă [Hay98] "probabilitatea ca ieşirea expertului k să se potrivească pe ieşirea D=d, fiind dat numai vectorul de intrare x".

Dacă se consideră y_k drept o estimare a mediei lui D condiționată de x şi k se poate scrie densitatea de probabilitate a lui D dacă este selectat expertul k:

$$f_D(d \mid x, k, \theta) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}(d - y_k)^2), \text{ k=1,…,K (4.3.)}$$

unde θ este vectorul de parametri ai rețelei poartă şi ai experților.

Densitatea de probabilitate a lui D condiționată doar de x devine:

$$f_D(d \mid x,\theta) = \sum_{k=1}^{K} g_k f_D(d \mid x,k,\theta) = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^{K} g_k \exp(-\frac{1}{2}(d - y_k)^2) \quad (4.4)$$

(modelul mixturii asociative gaussiene).

Mixtura ierarhică de experți

Este o extensie a mixturii de experți prezentate anterior (Figura 4.3), bazată pe principiul divide et impera şi are o structură asemănătoare unui arbore: mai multe rețele poartă distribuite ierarhic împart spațiul de intrare în subspații, unele dintre ele imbricate (Figura 4.4). Această structură reprezintă, de fapt, un arbore de decizie soft, cu avantajul că este posibilă recuperarea din decizii proaste, întrucât informația este păstrată până se ajunge la decizia/ estimarea de parametri finală. Mixtura ierarhică de experți este un model intermediar între perceptronul multistrat -model complex, care este precis dar acționează ca o "cutie neagră", şi arborii de decizie – model simplu mai puțin precis dar transparent.

Pentru arhitectura inițială a rețelei se poate folosi algoritmul CART pentru construirea arborilor de decizie din date. Fiecare punct de separare corespunde unui anumit subset al datelor de intrare. CART crează un arbore de decizie pe baza unui set de date –de exemplu, folosind acele separări ce duc la o eroare (i.e. distanța de la ieşire la media ieşirilor din submulțimea corespunzătoare) medie minimă. Punctele de separare din arbore corespund rețelelor poartă, nodurile terminale- experților.

Rețelele poartă sunt cele care mimează o separare de tipul CART (exprimată ca $a^T x+b=0$), suprafața de regresie soft fiind:

$$g = \frac{1}{1+\exp(-a^T x + b))} . \quad (4.5)$$

Densitatea de probabilitate a lui D este în acest model:

$$f_D(d \mid x,\theta) = \frac{1}{\sqrt{2\pi}} \sum_{k=1}^{K} g_k \sum_{j=1}^{K} g_{j|k} \exp(-\frac{1}{2}(d - y_{jk})^2) , \quad (4.6)$$

unde g_k, $g_{j|k}$ pot fi privite ca probabilități apriori, iar ieşirile rețelei poartă, - drept probabilități aposteriori.

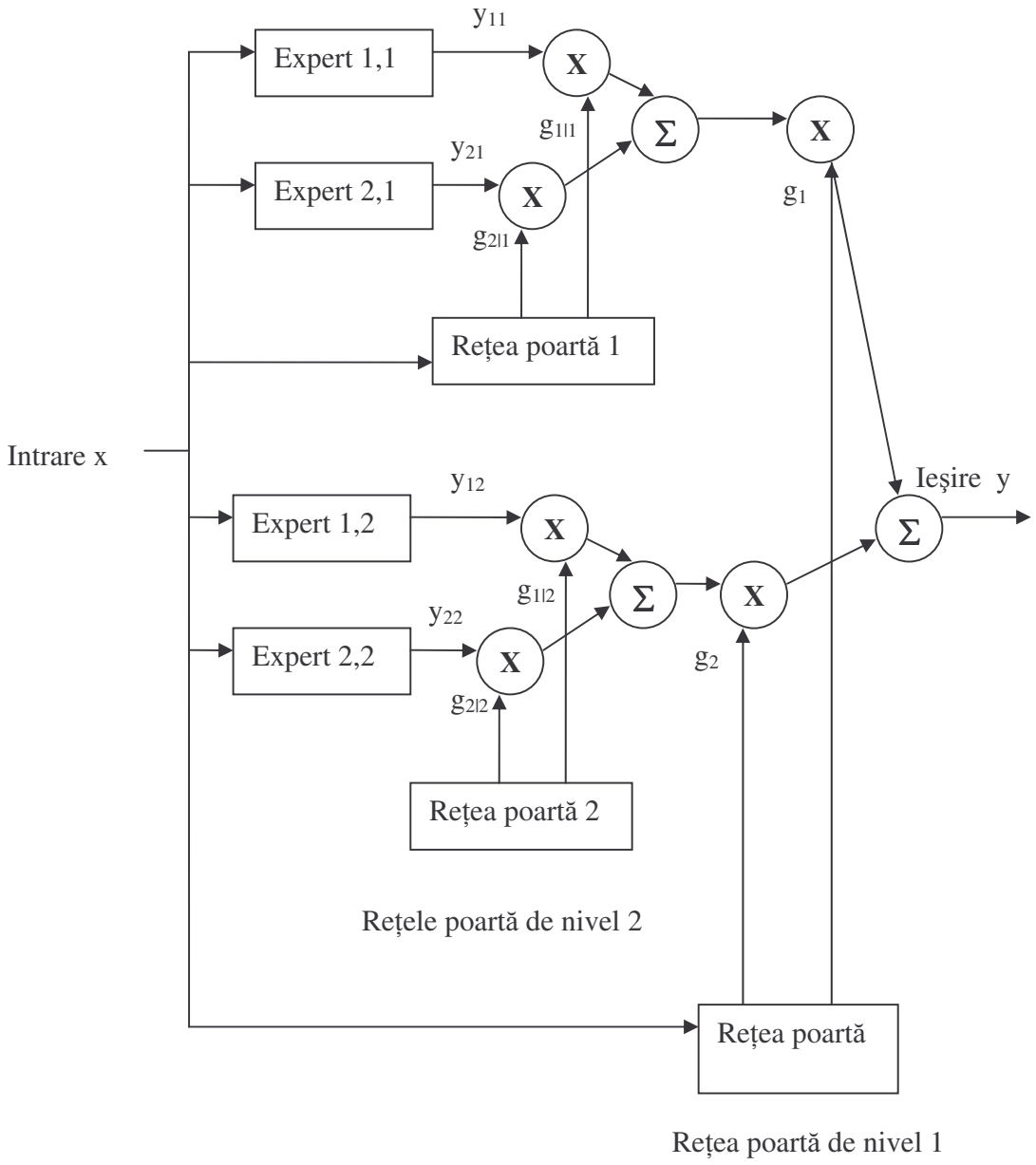


Figura. 4.4. Mixtura ierarhică de experți cu două nivele de ierarhie

Parametrii θ ai modelului pot fi estimați cu metoda verosimilității maxime, pornind de la setul de exemple de antrenare (vezi, de exemplu, algoritmul EM (Expectation Maximization) [Hay98], în cadrul căruia se ia în considerare şi posibila incompletitudine a datelor).

4.3.2.Hibridizare neuro-fuzzy

Flexibilitatea şi capacităţile de adaptare/ învăţare/generalizare din exemple ale reţelelor neurale şi avantajele reprezentării structurate şi uşor interpretabile a cunoştinţelor incerte din sistemele fuzzy de inferenţă pot fi combinate în cadrul unui formalism hibrid: calculul neuro-fuzzy, care şi-a găsit până acum aplicaţii atât în problemele medicale de clasificare cât şi în diagnoza tehnică.

Sistemele neuro-fuzzy se pot folosi ca generatoare de reguli şi vin astfel de multe ori în completarea sistemelor ce folosesc cunoştinţe furnizate de experţi umani: acolo unde cunoştinţele expertului uman sunt incomplete sau cunoştinţele unor experţi diferiţi sunt contradictorii, modelul se completează folosind exemple clasificate pentru crearea regulilor. Prezentăm în continuare câteva exemple reprezentative de arhitecturi hibride şi aplicaţiile lor, care să ilustreze aceste afirmaţii, şi care în plus ar fi aplicabile în diagnoza medicală.

4.3.2.1.ANFIS (Adaptive Neural Fuzzy Inference Systems)

Introducem în această secţiune o arhitectură hibridă fuzională, de sisteme de inferenţă fuzzy extinse cu abilităţile de învăţare ale reţelelor neurale pentru învăţarea parametrilor ce definesc mulţimile fuzzy din antecedentul/ consecventul regulii, bazate pe reţele adaptive[Jan92].

Reţelele adaptive (Figura 4.5) reprezintă un superset al reţelelor neurale feedforward cu învăţare supervizată. Componentele lor de bază sunt nodurile şi legăturile între noduri. O parte din noduri sunt adaptive, în sensul că ieşirea lor depinde de parametrii ce sunt asociaţi cu fiecare din aceste noduri. Mecanismul de învăţare (gradient descendent + chain rule) determină cum trebuie modificaţi aceşti parametri pentru a minimiza o măsură dată a erorii, şi a fost introdus de Werbos în 1970, şi este în principiu acelaşi cu cel folosit de Backpropagation. Vom descrie pe scurt acest mecanism în continuare.

Fiecare nod realizează o anumită funcţie *O* dependentă de intrări şi de parametrii asociaţi nodului. Se calculează derivata erorii în raport cu fiecare parametru α [Jan92]:

$$\frac{\partial E_p}{\partial \alpha} = \sum_{O^* \in S} \frac{\partial E_p}{\partial O^*} \frac{\partial O^*}{\partial \alpha}, \quad (4.7)$$

unde p reprezintă al p-lea pattern de antrenare, S este mulţimea de noduri ale căror ieşiri depind de α, iar

$$E_p = \sum_{m=1}^{\#(L)} (T_{m,p} - O_{m,p}^L)^2 ,$$

unde L este numărul de straturi al rețelei, #(L) este numărul de noduri din stratul de ieşire L, $T_{m,p}$ este componenta m a vectorului țintă numărul p, iar $O_{m,p}^{L}$ este componenta m a ieşirii actuale a rețelei pentru intrarea cu numărul p.

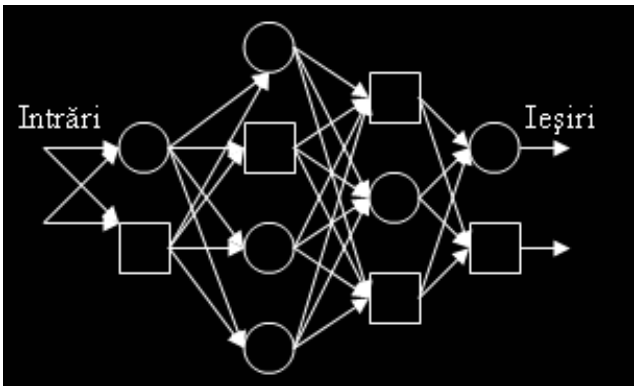


Figura 4.5. Structura unei rețele adaptive
(Nodurile adaptive sunt reprezentate ca pătrate, iar nodurile fixe ca ovale).

Dacă avem în total P pattern-uri de antrenare şi eroarea totală este

$$E = \sum_{p=1}^{P} E_p \,, \quad (4.8)$$

atunci:

$$\frac{\partial E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha} \quad (4.9)$$

şi modificarea parametrului α se face după regula:

$$\Delta\alpha = -\eta \frac{\partial E}{\partial \alpha} \,, \quad (4.10)$$

unde η reprezintă rata de învățare şi este dat de:

$$\eta = \frac{k}{\sqrt{\sum_{\alpha} \left(\frac{\partial E}{\partial \alpha} \right)^2}} \,.$$

Întrucât regula de învățare cu gradient descendent este lentă şi există riscul blocării în minime locale ale erorii, autorii propun o îmbunătățire a algoritmului, înlocuindu-l cu o tehnică hibridă care combină metoda gradientului cu estimare tip cele mai mici pătrate pentru identificarea parametrilor rețelei.

Orice sistem de inferență fuzzy poate fi modelat printr-un sistem ANFIS, dovedindu-se echivalent funcțional, dar bucurându-se în plus de abilitățile de învățare specifice rețelei.

Prezentăm în Figura 4.6 un exemplu de cum se traduce un sistem de inferență fuzzy de tip I într-o rețea adaptivă [Jan92].

În Figura 4.6, nodurile primului strat au asociată câte o funcție de tip $O_i^1 = \mu_{A_i}(x)$, unde x este intrarea în nodul i, şi O_i^1 specifică gradul în care x satisface cuantificatorul definit de mulțimea fuzzy A_i (mic, mare etc.). Parametrii variabili asociați nodurilor din primul strat sunt dați de parametrii ce definesc mulțimile A_i şi se numesc *parametrii premizei*. Nodurile celui de-al doilea strat aplică un operator de tip t-normă mulțimilor A_i , B_i din antecedentul regulii *i* pentru a calcula puterea de activare a acestei reguli. Al treilea strat normalizează ponderile regulilor, iar stratul patru calculează consecințele aplicării regulilor (în funcție de parametrii consecventului).

$$O_i^4 = \overline{w_i} f_i = \overline{w_i}(p_i x + q_i y + r_i), \quad (4.11)$$

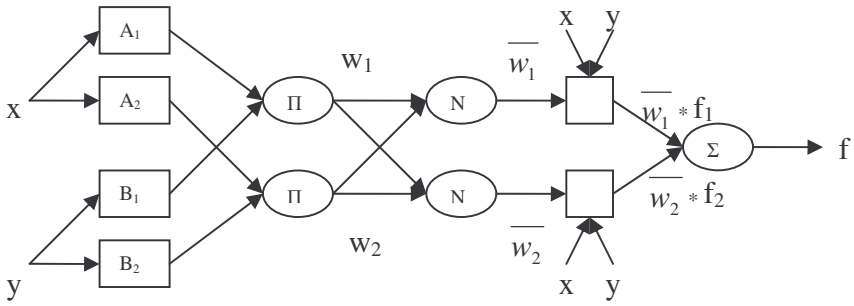


Figura 4.6. Rețea adaptivă pentru un sistem de inferență fuzzy de tipul I

unde $\overline{w_i}$ este ponderea normalizată calculată de stratul anterior. Nodul din ultimul strat calculează rezultatul (crisp) final:

$$O_i^5 = \sum \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \quad (4.12)$$

Autorii arată în aceeaşi lucrare că sistemele de inferență fuzzy sunt echivalente, din punctul de vedere al funcției calculate, cu rețelele neurale de tip RBF(rețele cu funcții bază radiale), ce pot fi privite ca un caz special al ANFIS, dacă asociem fiecare regulă fuzzy DACĂ-ATUNCI cu un câmp receptor al rețelei RBF. (Această asociere este posibilă datorită proprietății de procesare locală a informației în rețelele RBF).

Fiind aproximatori universali, rețelele ANFIS şi-au găsit aplicații multiple în modelarea de funcții nelineare din diverse domenii.

4.3.2.2. Clasificatorul neuro-fuzzy NFC

Este o arhitectură fuzională, concepută special pentru clasificare de forme [NFC=neuro-fuzzy classifier]. Este mai eficient decât clasificatorii neurali întrucât, pe de o parte, permite codificarea transparentă de cunoştinţe apriori în parametrii reţelei şi determină astfel învăţarea în spaţiul parametrilor să înceapă dintr-un punct iniţial bun (aproape de optim). Pe de altă parte, parametrii obţinuţi după învăţare se pot transforma uşor în cunoştinţe structurate de forma regulilor IF-THEN. Arhitectura este prezentată în Figura 4.7, unde parametrul p_{ij} reprezintă măsura a cât aparţine regiunea fuzzy i clasei C_j.

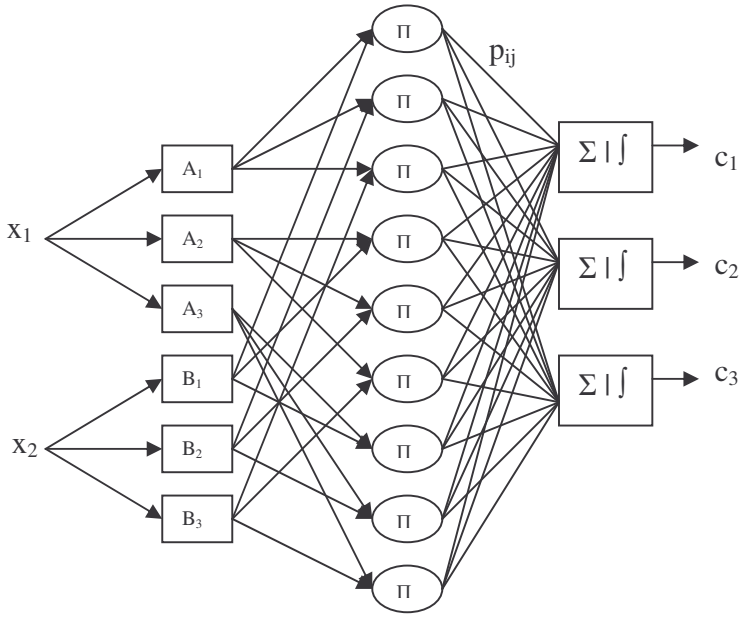


Figura 4.7.Structura NFC

Cum într-o problemă de clasificare ieşirea este *discretă,* avem nevoie de o măsură a erorii corespunzătoare. Autorii propun ca măsură a erorii de tip maximum o funcţie continuă ce reflectă corect numărul de clasificări greşite.

Astfel, pentru un exemplu cu trei clase, dacă notăm $d=(do_1, do_2, do_3)$ vectorul ieşirilor dorite, şi cu $co=(co_1, co_2, co_3)$ vectorul ieşirilor actuale, şi presupunând că ieşirea dorită ar fi clasa numărul 1 eroarea se scrie:

$$E=sig[m(co_2-co_1)]+sig[m(co_3-co_1)], \quad (4.13)$$

unde m este o constantă pozitivă mare care face ca funcţia diferenţiabilă sigmoidală să aproximeze funcţia signum.

Această funcţie de eroare are proprietăţile dorite în condiţiile date (clasa corectă e c_1), şi anume:

dacă $co_1 > co_2$ şi $co_1 > co_3$ atunci E este mică
dacă $co_1 > co_2$ si $co_1 < co_3$ atunci E este medie

dacă $co_1 < co_2$ si $co_1 > co_3$ atunci E este medie
dacă $co_1 < co_2$ si $co_1 < co_3$ atunci E este mare

Expresia erorii pentru cazul general este:

$$E=do_1(1-do_2)(1-do_3)\{(sig[m(co_2-co_1)]+sig[m(co_3-co_1)]\} + (1-do_1)do_2(1-do_3)\{(sig[m(co_1-co_2)]+sig[m(co_3-co_2)]\} + (1-do_1)(1-do_2)do_3\{(sig[m(co_1-co_3)]+sig[m(co_2-co_3)]\} \quad (4.14)$$

4.3.2.3.Perceptronul Fuzzy Multistrat

Modelul care urmează reprezintă o rețea neurală tip MLP extinsă cu abilități de procesare de intrări-ieşiri fuzzy [Mit00]. Foloseşte hibridizare asociativă (fuzională şi de transformare -Figura 3.21).

Modelul fuzzy MLP include mulțimile vagi la nivelul reprezentării intrărilor şi ieşirilor, şi conține un singur strat ascuns. Poate lucra atât cu intrări numerice cât şi lingvistice. Astfel, dacă F_j este vectorul n-dimensional de caracteristici cu numărul de ordine j (din datele de antrenare),

$$F_j = [\mu_{low(F_{i1})}(F_i),...,\mu_{high(F_{in})}(F_i)] = [y_1^0, y_2^0,..., y_{3n}^0], \quad (4.15)$$

adică fiecare componentă a sa este trecută prin câte 3 valori lingvistice (mic, mediu, mare), definite de mulțimile fuzzy de tip π:

$$\pi(F_j;c,\lambda) = \begin{cases} 2\left(1-\frac{\| F_j - c \|}{\lambda}\right)^2, \frac{\lambda}{2} \le \| F_j - c \| \le \lambda \\ 1-2\left(\frac{\| F_j - c \|}{\lambda}\right)^2, 0 \le \| F_j - c \| \le \frac{\lambda}{2} \\ 0, altfel \end{cases} \quad (4.16)$$

(c, λ-se obțin din datele de antrenare).

Ieşirea oricărui neuron din stratul ascuns sau de ieşire se exprimă ca:

$$y_j^{h'} = \frac{1}{1+\exp(-\sum_i y_i^{h'-1} w_{ji}^{h'-1})} \quad (4.17)$$

unde $x_j^{h'} = \sum_i y_i^{h'-1} w_{ji}^{h'-1}$.

Reprezentarea ieşirilor

Din datele de antrenare se calculează media $o_k=[o_{k1},\ldots o_{kn}]$ şi varianța $v_k=[v_{k1},\ldots,v_{kn}]$ pentru fiecare clasă $C_k,k=1,l.$ Acestea folosesc la determinarea distanței ponderate de la exemplul de antrenare F_i la clasa C_k:

$$z_{ik} = \sqrt{\sum_{j=1}^{n}\left[\frac{F_{ij} - o_{kj}}{V_{kj}}\right]^2}\ , \quad (4.18)$$

şi, mai departe, la calculul gradului de apartenență a lui F_i în clasa C_k:

$$\mu_k(F_i) = \frac{1}{1+\left(\frac{z_{ik}}{f_d}\right)^{f_e}} \quad (4.19)$$

unde f_e , f_d sunt parametri aleşi în funcție de acuratețea dorită.

După ce rețeaua este antrenată cu backpropagation şi ajustată ("pruned") cu metoda de degradare a ponderilor ("weight decay"), se pot extrage reguli din ponderile rețelei, urmărind prin backtracking căile de pondere ridicată, de le stratul de ieşire spre cel de intrare, şi asociind o regulă cu fiecare submulțime de noduri ataşată (ce defineşte căile respective.)[Mit00]

Arhitectura a fost folosită cu succes de către autori în extragerea de reguli pentru diagnosticarea afecțiunilor hepatice [Mit00].

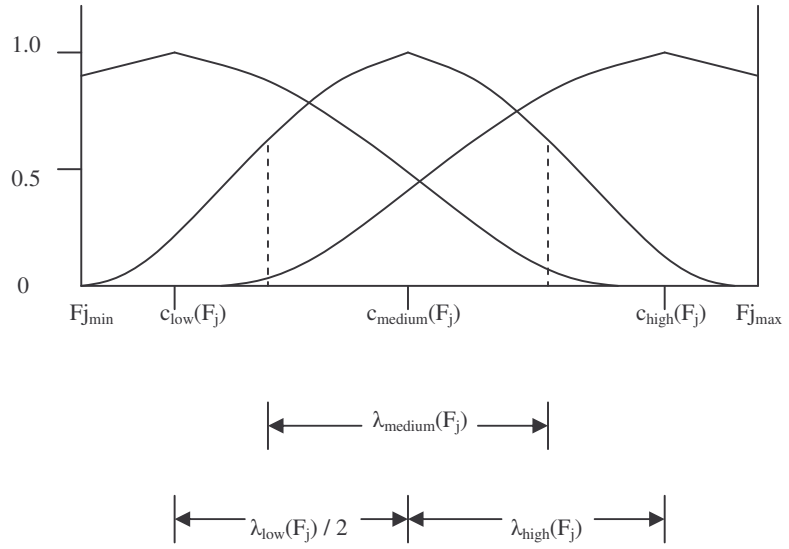


Figura 4.8. Definiția mulțimilor π

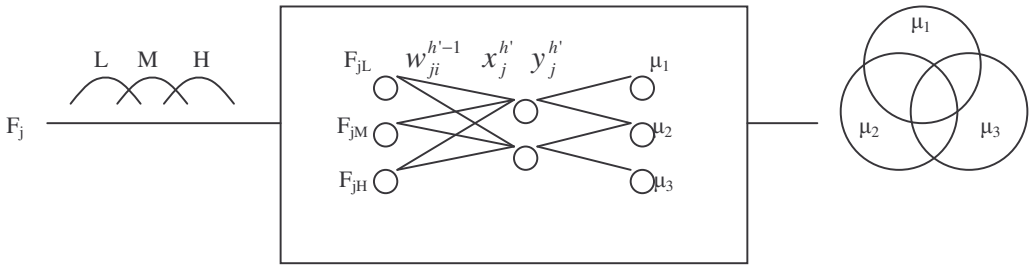


Figura 4.9.Schema generală a Fuzzy MLP

4.3.2.4.Neuroni fuzzy logici

Hibridizarea poate merge însă şi mai departe, în interiorul rețelei, la nivelul structurii neuronilor (hibridizare fuzională). O arhitectură interesantă de rețea neurală fuzzy, bazată pe *neuroni fuzzy logici* a fost propusă de Witold Pedrycz [Ped04] şi o prezentăm pe scurt în continuare împreună cu ideea unei posibile aplicații.

Neuroni fuzzy logici de agregare

Fie $x=[x_1,\ldots,x_n]$ vectorul de intrări într-un neuron şi $w=[w_1,\ldots,w_n]$ vectorul de ponderi asociat.

Neuronul de tip OR. Ieşirea sa este dată de:

$$y = \overset{n}{\underset{i=1}{S}}(x_i t w_i) \text{ (unde } S \text{ este o s-normă şi } t \text{ o t-normă)} \quad (4.20)$$

Acest y reprezintă valoarea de adevăr a expresiei logice:

$$(x_1 \text{ and } w_1) \text{ or } (x_2 \text{ and } w_2) \text{or} \ldots \text{ or } (x_n \text{ and } w_n).$$

Se observă că valori mici ale ponderilor scad impactul intrărilor respective asupra rezultatului final.

Neuronul de tip AND. Ieşirea acestuia se exprimă ca:

$$y = \overset{n}{\underset{i=1}{T}}(x_i s w_i) \quad \text{(s= s-normă, t=t-normă).} \quad (4.21)$$

De data aceasta, valori mari ale ponderilor reduc impactul intrării asupra rezultatului final.

Neuroni logici de referință

Folosesc la procesarea predicatelor logice de tipul: *similar*, *inclus în*, *domină*. Valoarea de adevăr a unui astfel de predicat este dată de satisfacerea unei expresii de tipul *P(x,a)*, unde a reprezintă o valoare de referință. (De exemplu *P(x,a)* atunci când *P=similar*, se interpretează "*x* este similar cu *a*").

Dacă **x** şi **a** sunt vectori atunci:

$$P(\mathbf{x};\mathbf{a}) = P(x_1,a_1) \text{ and } P(x_2,a_2) \text{ and } \ldots \text{ and } P(x_n,a_n). \quad (4.22)$$

Pentru a pondera relevanța variabilelor în satisfacerea unui predicat dat, expresia de mai sus se poate scrie:

$$P(\mathbf{x};\mathbf{w},\mathbf{a}) = [P(x_1,a_1) \text{ or } w_1] \text{ and } [P(x_2,a_2) \text{ or } w_2] \text{ and } \ldots \text{ and } [P(x_n,a_n) \text{ or } w_n]. \quad (4.23)$$

În general, ieşirea unui neuron de referință este:

$$y = \overset{n}{\underset{i=1}{T}}(REF(x_i, a_i)sw_i). \quad (4.24)$$

Dacă, spre exemplu, REF = SIM(ilar), acesta se poate defini ținând cont de relațiile:

$$\text{INCLus}(x,a) = x \rightarrow a; \quad (4.25)$$

$$\text{DOMină}(x,a) = a \rightarrow x; \quad (4.26)$$

$$\text{SIM}(x,a) = \text{INCL}(x,a) \text{t } \text{DOM}(x,a), \quad (4.27)$$

şi $a \rightarrow b$ poate fi modelată prin t-norma:

$$a \rightarrow b = \sup\{c \in [0,1] \mid \text{atc} \leq b\}, a,b \in [0,1].$$

Topologia generală a rețelei este prezentată în Figura 4.10, unde neuronii din stratul τ reprezintă modificatori lingvistici de tipul *foarte, puțin, mai mult sau mai puțin* (efect de diluare sau accentuare).

Arhitectura şi ponderile se pot antrena folosind algoritmi genetici. În final, rețeaua poate fi transformată într-o formulă logică transparentă, bine-structurată.

O posibilă aplicație a acestei arhitecturi se poate obține în diagnoza pe bază de funcții criteriu [Mun03], neuronii fuzzy logici servind la modelarea naturală a operatorilor şi mulțimilor fuzzy ce apar în construcția funcțiilor respective. Această idee poate constitui o direcție de studiu interesantă, în cadrul tehnicilor de integrare a cunoştințelor simbolice în structuri neurale. Cunoştințele simbolice integrate pot preveni blocarea în minime

locale şi îmbunătăţesc timpul de învăţare al reţelei, fiind totodată utile în faza de explicare a rezultatelor furnizate de reţea. În plus, structura de reţea neurală ar ajuta inferenţa prin avantajele procesării paralele şi robusteţii la forme incomplete şi distorsionate. Există totuşi limitări ale capacităţii unei reţele neurale de a simula un raţionament nemonoton, provenite tocmai din paralelismul reţelei şi din lipsa abilităţilor de interacţiune cu utilizatorul (reţeaua nu poate pune întrebări care să ghideze raţionamentul). Rămâne o problemă deschisă dacă şi cum ar putea fi inserată interactivitatea şi nonmonotonicitatea într-o structură hibridă neuro-fuzzy.

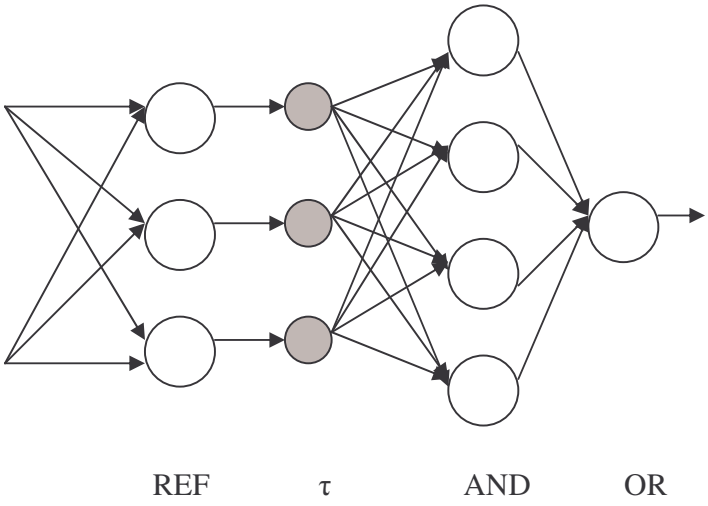


Figura 4.10. Topologia unei reţele cu neuroni fuzzy logici

4.4.Un exemplu de hibridizare combinativă în diagnoza medicală

4.4.1.Sistemul CHECK

Problema diagnozei pe bază de model este în general NP-completă. Se pot însă folosi hibridizări combinative, în care primul nivel al sistemului de diagnoză se ocupă de focalizarea raţionamentului, apelând la tehnici asociative de raţionament. Reducerea spaţiului de căutare în cazul diagnozei pe bază de model poate fi abordată şi numeric, folosind măsuri de probabilitate/posibilitate ([TorCon89 , Mun03]). Un exemplu de astfel de sistem hibrid este propus în continuare. Sistemul se încadrează în tipul de hibridizare în care logica formală este asistată de o tehnică inteligentă (simbolică în cazul de faţă), şi constituie o alternativă mult îmbunătăţită a sistemului CASNET.

În [TorCon89] ni se propune o organizare hibridă (CHECK) (=ce foloseşte atât cunoştinţe de profunzime cât şi raţionament asociativ), interesantă şi originală a cunoştinţelor pe 3 nivele: nivelul descrierii datelor (1), nivelul euristic (2)(folosit pentru inferenţă, include cunoştinţe de suprafaţă), nivelul modelului cauzal (3) (folosit pentru explicaţii-include cunoştinţe profunde). Sistemul este unul dintre cele mai complete şi aprofundate sisteme

de diagnoză şi a fost folosit cu rezultate bune în diagnosticarea afecţiunilor ficatului. Pe toate cele trei nivele, reprezentarea cunoştinţelor foloseşte cadre, cu sloturi specifice. La nivelul descrierii datelor există patru tipuri de informaţii (structurale, de control, prototipice, de interpretare). Nivelul euristic cuprinde cunoştinte structurale (de exemplu, o relaţie de ierarhizare între diagnostice -cazuri de bază/cazuri specializate=specifice), cunoştinţe prototipice (manifestări necesare şi suplimentare pentru un diagnostic), cunoştinţe de control (reguli de activare şi validare, ipoteze asociate şi alternative unei ipoteze date, specializări default).

Descrierile prototipice ale unui diagnostic sunt folosite în evaluarea gradului de compatibilitate al datelor observate cu profilul unei anumite boli, şi în asocierea unui rang de plauzibilitate fiecărei ipoteze de diagnostic. Evaluarea evidenţelor se face corectând evidenţa observată cu măsura relevanţei acesteia într-un anumit context, după formula:

$$E(H)=(((e(N) +_u e(S)) \circ_f [1-e(ER)]) +_u e(CR)) \quad (4.28)$$

unde:

$+_u$: $e_1 +_u e_2 = e_1 + (1-e_1)*e_1*e_2$; (varianta "unfair" a legii de combinare a evidenţelor Bernoulli folosită în Mycin: primul membru-care corespunde în 4.28 manifestărilor necesare -este privilegiat, acordându-i-se o pondere mai mare);

$\circ_f$: $e_1 \circ_f e_2 = e_1*e_2$ (adaptat semanticii regulilor de excludere: gradul de evidenţă global nu trebuie să descrească dacă evidenţa din regulile de excludere este nulă);

e(N): evidenţa din manifestările necesare; e(S): evidenţa din manifestările suplimentare; e(ER)=evidenţa din regulile de excludere; e(CR)=evidenţa din regulile de confirmare.

Această evaluare a evidenţelor este folosită de regulile de activare pentru instanţierea cadrului corespunzător unei ipoteze de diagnostic şi trecerea sa în agenda de lucru. Regulile de validare sunt apoi folosite pentru confirmarea /excluderea instanţei unui cadru(=a unei ipoteze). Algoritmul de diagnoză corespunde unei căutări în lărgime (în paralel) la nivelul euristic, ceea ce face posibil şi diagnosticul diferenţial.

Algoritm

1. Activarea mulţimii iniţiale de ipoteze =A;
2. Determinarea ipotezelor asociate care nu sunt în A → mulţimea C;
3. Activarea ipotezelor din C → C';
4. S=A ∪ C';
5. Validarea ipotezelor din S → mulţimea de ipoteze H;

6. Aplicarea unei funcții-prag;

7. Specializarea ipotezelor din H.

Nivelul cauzal cuprinde cunoştinţele de profunzime şi este folosit pentru confirmarea/infirmarea unei ipoteze generate la nivelul euristic, generarea de ipoteze alternative sau analiza datelor "neaşteptate"-poate fi folosit şi pentru interogare. Pentru a modela raționamentul în această subcomponentă, se foloseşte o abordare calitativă bazată pe logica nemonotonă. În această abordare, raționamentul logic nemonoton pe o rețea cauzală foloseşte *principiul rezoluției extins* pentru a găsi sursa unei inconsistențe. (Dacă o manifestare a unei stări lipseşte, se aplică abducția indirectă pentru a găsi explicația acestei observații inconsistente). Dezavantajele principale ale abordării sunt cele enunțate în Secțiunea 3.4.1. în paragraful dedicat versiunii duale a abducției.

Nodurile din structura rețelei cauzale (Figura 4.12.) sunt de patru tipuri:

- **ipoteze** (=situații (stări/combinații de stări) în care se poate afla sistemul la un moment dat); fiecare ipoteză este conectată cu un cadru care o reprezintă la nivelul euristic;

- **stări şi acțiuni**- stările au asociate variabile de stare (caracteristici ale stărilor) care se modifică prin funcțiile definite de acțiuni (evenimente ce determină tranziția dintr-o stare în alta, şi care reflectă legăturile cauzale dintre stări);

- **manifestări observabile** ale stărilor interne (cel mai adesea inaccesibile decât prin teste invazive/costisitoare);

- **cauze inițiale** =posibile cauze originare ale unei boli.

Relațiile care leagă aceste noduri în cadrul rețelei sunt, deasemenea, împărțite în mai multe categorii, în funcție de specializarea lor:

- arce cauzale; stare s→acțiune→stare t;

- arce de tip M (“Has As a **M**anifestation”) -conectează stările cu manifestările asociate;

- arce LOOP (bucle);

- arce de tip "sugerează...": leagă stările de ipoteze;

- arce de tip "definit ca...": leagă, deasemenea, stările de ipoteze, dar spre deosebire de precedentele, sunt certe: o ipoteză de diagnostic se defineşte ca prezența unei stări particulare (combinație de stări) în cadrul sistemului modelat.

Între arcele de acelaşi tip se pot folosi operatori logici de tip AND,OR. Unele arce se definesc de tip MAY, ca şi în AID, pentru a exprima incertitudinea asociată cu incompletitudinea modelului teoretic. În cadrul raţionamentului necesar pentru confirmarea unei stări trebuie stabilită prezenţa condiţiilor strict necesare pentru această stare, şi stabilirea unui grup de cauze iniţiale şi a unor drumuri cauzale de la acestea la starea respectivă. Inferenţa se face în principal înainte (forward), încercând să se instanţieze reţeaua pornind de la cauzele iniţiale. Raţionamentul ipotetic asociat reţelei se defineşte pe baza a patru elemente principale:

1. **Stări** = formule logice definite cu ajutorul unor predicate de aritate egală cu numărul de atribute al stării.

2. **Acţiuni =** funcţii ce permit exprimarea atributelor stării efect în termenii atributelor stării cauză.

Dacă S_1 are două atribute x,y şi S_2 are un singur atribut avem implicaţia cauzală (Figura 4.11):

$$S_1(x, y) \wedge C(x, y) \Rightarrow S_2(f(x, y)) \quad (4.29)$$

unde funcţia f modelează A.

3. Arce **"M"** (has as a manifestation): $S(x,y) \Rightarrow m(z)$;

4. Arce **may** (relaţii posibile-definesc calitativ incertitudinile din model):

$$S_1(x, y) \wedge \alpha \Rightarrow S_2(f(x, y)), \quad (4.30)$$

unde α este un literal ipotetic (nu există evidenţe pentru el).

Se transformă astfel graful asociat reţelei într-o mulţime de formule logice pe baza cărora se poate raţiona calitativ (Figura 4.12).

Specificarea teoriei cauzale propuse de Torasso şi Console este dată de cadrul care urmează.

Definiţie. O *specificare cauzală a unei teorii de diagnostic* este cadrul C=(DFS, OBS, CM), unde:
- DFS este o mulţime de literali de defecte sau ipoteze posibile,
- OBS este mulţimea manifestărilor observabile -negative sau pozitive-;
- CM (Causal Model) este o mulţime de axiome de forma:

$$d_1 \wedge ... \wedge d_n \rightarrow o \quad (4.31)$$

(axiome de anormalitate: implicația logică exprimă o relație de cauzalitate) (d_i: defecte sau stări anormale intermediare între o cauză rădăcină şi simptomele finale);

$$d_1 \wedge ... \wedge d_n \rightarrow d \quad (4.32)$$

(axiome de clasificare, după modelul CASNET: un defect este definit în termenii unei mulțimi de stări).

Relații cauzale incerte:

$$d_1 \wedge ... \wedge d_n \wedge \alpha_o \rightarrow o \quad (4.33)$$

$$d_1 \wedge ... \wedge d_n \wedge \alpha_d \rightarrow d \quad (4.34)$$

unde α este literalul "ipotezei de incompletitudine", la fel ca în relația 4.30 de mai sus.

Figura 4.11.Exemplu de arc cauzal

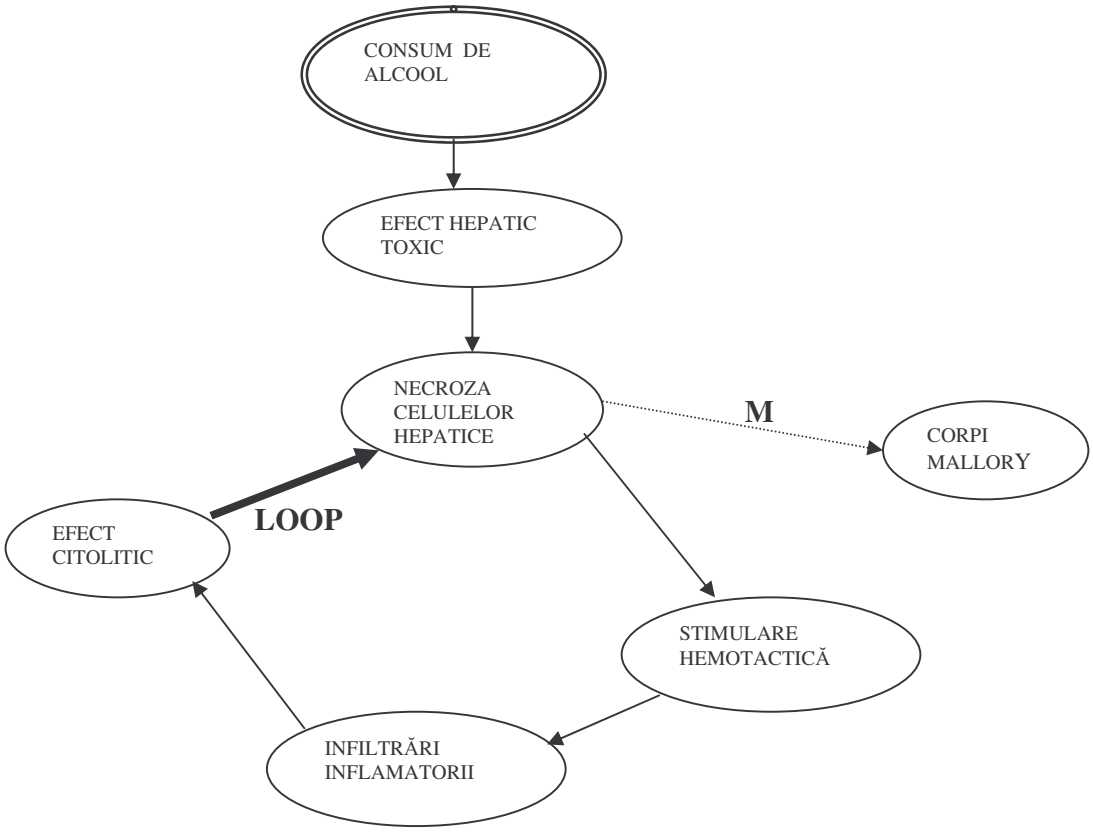


Figura 4.12. Rețea cauzală în CHECK

4.4.2 Comparație între sistemele CHECK şi CASNET

Sistemul CHECK reuşeşte să îmbunătățească, în mare parte, dezavantajele sistemului CASNET enunțate în Secțiunea 3.3.1.4. Pe de o parte, prin activarea pe bază de reguli a anumitor ipoteze, raționamentul se concentrează pe un submodel din modelul medical total. Acestui submodel i se aplică, abia în faza de generare de explicații, raționament nemonoton, care abordează net superior contradicțiile. În plus, detalierea superioară a modelului şi definirea explicită a mai multor tipuri de relații (de confirmare sau infirmare) va determina scăderea numărului de contradicții generate la pasul de explicare a concluziilor.

Tostuşi, încă nu se poate modela în CHECK situația în care o ipoteză e confirmată doar dacă *majoritatea* (criteriu vag) testelor sale sunt confirmate. Această problemă va fi rezolvată în DiaMed.

4.5.Sistemul hibrid DiaMed

În continuare, structura lucrării ține cont de obiectivul pe care ni l-am propus, şi anume construirea unui sistem hibrid original de tip combinativ, structurat pe două nivele, pe care l-am numit DiaMed (**Dia**gnoză **Med**icală). Ne-am oprit asupra hibridizării combinative pentru a evita două mari dezavantaje ale hibridizării neuro-fuzzy sau neuro-simbolice, şi anume: creşterea exponențială a dimensiunii problemei cu creşterea numărului de variabile ("curse of dimensionality"), şi dificultatea de a se plia pe dinamismul unei probleme de diagnoză medicală.

Arhitectura DiaMed seamănă cu cea din sistemul CHECK, dar metodele efective de realizare sunt diferite. Primul nivel implementează selecția ipotezelor cu o metodă eficientă din categoria raționamentului asociativ, apelând în acest scop la funcții de decizie fuzzy pentru definirea funcțiilor de discriminare între clase. Avantajul major al acestor funcții de decizie față de legea combinării evidențelor din CHECK stă în faptul că pot exprima cu acuratețe criterii vagi de mare diversitate (de exemplu, *majoritatea, cel puțin x din n, un număr semnificativ din* etc.)

Al doilea nivel utilizează modelul cauzal de profunzime restrâns la contextul ipotezelor selectate la primul pas, pentru a discrimina şi rafina diagnosticul final, şi, eventual, pentru înlăturarea contradicțiilor generate la primul nivel. Aplicarea algoritmilor DCSP pentru a controla constrângerile active la un moment dat are față de CASNET avantajul de a restrânge contextul, aşa cum făceau şi regulile de activare a ipotezelor (trigger) în CHECK. În plus, acest pas ia în considerare un model cât mai precis şi complet, în care situațiile excepționale să poată fi reprezentate eficient şi natural, şi schema de raționament să se încadreze în specificul nemonoton al unui raționament de diagnoză. În acest scop, am apelat în cadrul acestui al doilea nivel la metode logico-simbolice din domeniul sistemelor de argumentare şi la algoritmi CSP pentru rafinarea şi explicarea rezultatelor. Avantajul principal față de CHECK al acestei abordări a nonmonotonicității la nivelul rafinării ipotezelor în DiaMed stă în faptul că foloseşte o metodă tratabilă,

eficientă, spre deosebire de abordarea logică a abducției indirecte ce determină sursa eventualelor inconsistențe în CHECK.

4.6.Concluzii şi contribuții

Capitolul IV a realizat mai mult decât o simplă trecere în revistă a sistemelor hibride de diagnoză medicală de succes de până astăzi. Întrucât am dorit ca în acest capitol să încadrăm deja în contextul actual sistemul original DiaMed propus de lucrarea prezentă, cu avantajele sale, am simțit nevoia detalierii anumitor formalisme, pentru ca secțiunile ulterioare să permită o mai bună comparație. Al doilea obiectiv a fost deschiderea de noi perspective asupra aplicabilității anumitor tehnici în diagnoza medicală în special. Secțiunile următoare vor analiza în detaliu sistemul original DiaMed, cu cele două nivele ale sale: selectarea ipotezelor şi rafinarea mulțimii de diagnostice.

V. Selectarea ipotezelor de diagnostic prin decizie fuzzy în DiaMed

5.1.Introducere

Operația de selectare a ipotezelor foloseşte, de obicei, tehnici eficiente dar mai puțin precise şi aproape deloc transparente. Metodele cele mai populare de inteligență artificială ce folosesc cunoştințe superficiale pentru un raționament bazat pe asocieri sunt rețelele neurale [HayS98], sistemele bazate pe reguli[GiaRil94], anumite modele fuzzy[DubPra85]. Toate aceste tehnici modelează expertiza umană pe baza unor perechi intrări-ieşiri: corelația statistică fiind conceptul-cheie ce stă în spatele lor, indiferent de formalizare. (Pentru o sinteză a abordărilor posibile ale diagnozei medicale, statistice sau pe bază de model vezi [Lon01]).

Secțiunea curentă prezintă contribuțiile originale pentru faza de selectare a ipotezelor în cadrul diagnosticării, contribuții ce cuprind un model original bazat pe funcții de decizie fuzzy [Mun03, Mun05]. Abordarea este comparată cu tehnica cea mai apropiată, şi anume sistemele de inferență fuzzy. Puterea de discriminare a funcțiilor astfel construite este evaluată folosind indexul statistic C, rezultatele fiind foarte bune.

5.2.Sisteme de inferență fuzzy în diagnoza medicală

O problemă de diagnoză medicală se poate formaliza natural ca un proces de clasificare cu o mulțime de N clase (diagnostice) [CasFanMen03]:

$$\Delta=\{d_1,\ldots,d_N\} \quad (5.1)$$

Mulțimea de simptome pe baza căreia se face clasificarea se poate scrie ca un vector K-dimensional $o=(o_1,\ldots o_K)$, iar diagnosticarea se face pe baza unei funcții clasificator:

$$D:A \subseteq \Re^K \rightarrow \Delta, \quad (5.2)$$

unde A se poate defini printr-un hiperinterval:

$$A = \prod_{j=1}^{K} dom_j, \qquad o_j \in dom_j=[i_j,I_j]$$

Baza de reguli fuzzy ce stă în spatele funcției de clasificare D conține de obicei reguli de forma:

IF o is G_r THEN $\tilde{D}(x)$ is $d_1(v_{r1})$, …, $d_N(v_{rN})$, (5.3)

unde G_r este o relație fuzzy K-dimensională.

Gradul de apartenență al vectorului de măsurători o în G_r defineşte gradul de activare al regulii r (şi poate fi interpretat ca măsură a similarității între o şi vectorul prototip al mulțimii G_r).

Regulile de mai sus definesc funcția de clasificare :

$$\tilde{D}: A \rightarrow [0,1]^M \quad (5.4)$$

adică $\tilde{D}$ (o)=(v_1,…,v_N), unde fiecare v_j reprezintă gradul în care este prezent diagnosticul d_j la un pacient cu simptome date de vectorul o. Acest grad poate avea diferite semantici [KunSte99]:

- Cât de tipic este cazul o pentru diagnosticul d_j;
- Cât de gravă este boala d_j în cazul o;
- Suportul pentru ipoteza că d_j este adevăratul diagnostic în cazul o, dedus din evidențele disponibile;
- Probabilitatea că d_j este diagnosticul real pentru o.

Gradul de apartenență al lui o la diagnosticul d_j calculat pe baza a R reguli de tipul (5.3) se calculează cu formula:

$$v_j = \frac{\sum_{r=1}^{R} G_r(o) v_{rj}}{\sum_{r=1}^{R} G_r(o)} \quad (5.5)$$

(unde G_r(o) este gradul de apartenență al vectorului o la relația fuzzy G_r)

Decizia finală poate alege clasa cu grad maxim de realizare drept diagnostic, sau poate păstra în lista ipotezelor posibile toate diagnosticele al căror grad depăşeşte un prag dat.

5.3.Decizie fuzzy în diagnoza medicală

Fie $\Delta = \{d_1, ..., d_N\}$ un context restrâns de boli (clase/ipoteze de diagnostic) luate în considerare, şi $M = \{m_1, ..., m_K\}$ un set complet de manifestări ce le caraterizează. Considerăm deasemenea că Δ este o mulțime completă de cauze pentru *M*. Matricea

$WEIGHTS = (w_{ij})_{j=1..K}^{i=1..N}$ conține indicii de relevanță ai simptomelor în cadrul definițiilor claselor, calculați statistic sau furnizați de un expert.

Originalitatea abordării prezente constă în definirea claselor (bolilor) prin transformarea unui model de decizie fuzzy, după cum urmează:

- Fiecare simptom m_j este definit de o funcție fuzzy $M_j : dom_j \to [0,1]$, (dom_j reprezintă domeniul funcției M_j, şi poate fi discret sau continuu, -dependent de simptom), şi $A = \prod_{j=1}^{K} dom_j$ este mulțimea tuturor observațiilor posibile (Π este produsul cartezian; observațiile sunt luate în considerare pe domeniul tuturor simptomelor posibile, deşi doar o mică parte sunt folosite efectiv în practică, întrucât restul sunt 0- nu au fost observate încă sau nu sunt prezente la sistemul sub observație);

- Observațiile sunt reprezentate de câte un punct K-dimensional $o = (M_1(o),...,M_K(o))$; -acest vector se poate modifica pe parcurs, pe măsură ce noi informații sunt obținute prin testare;

- Diagnosticele se definesc prin intermediul funcțiilor de decizie fuzzy: $D_1^{w_1},...,D_N^{w_N}$, după cum urmează:

$$D_i^{w_i} = h_i(M_{i_1}(o),...,M_{in_i}(o)) \quad (5.6)$$, unde:

  - $Simptome(d_i) = \{m_{i_1},...,m_{in_i}\}$, şi $i_1,...i_{n_i} \in \{1,...,K\}$ reprezintă indicii simptomelor relevante pentru clasa de diagnostic d_i;
  - w_i este vectorul ponderilor simptomelor în cadrul definiției clasei d_i;
  - h_i este o funcție de agregare definită cu ajutorul unor operatori fuzzy ce modelează modul de raționare al unui expert uman.

Detaliem în continuare modul de calcul.

Fie "$\wedge$" , "$\vee$" o T-normă şi, respectiv, o S-normă[SouKay02]. Studiile desfăşurate în cadrul logicii fuzzy şi teoriei deciziei au demonstrat că agregarea conjunctivă şi disjunctivă sunt insuficiente pentru a modela corect comportamentul avut de experții umani în luarea unei decizii. Multe situații practice sugerează mai degrabă un mod de gândire compensator: proprietățile bune ale unor criterii compensează pentru proprietățile nesatisfăcătoare ale altora (diagnoza medicală este un exemplu foarte potrivit, întrucât observarea câtorva simptome dintr-o mare diversitate este suficientă pentru a diagnostica o anumită boală). Definim mai jos tehnicile specifice de agregare compensatorie pe care le vom folosi.

Definiție 5.1 [SouKay02]. Un operator ordonat ponderat de medie (*ordered weighted averaging (OWA))* a *m* variabile este o funcție $W : \Re^m \to \Re$ cu un vector asociat $w^T = (w_1,\ldots,w_m)$, unde ponderile w_i satisfac:

$$\sum_{i=1}^{m} w_i = 1, \quad w_i \in [0,1], i = \{1,...,m\} \quad (5.7)$$

astfel încât:

$$W_w(b_1,...,b_m) = \sum_{i=1}^{m} w_i b_i \ , \ (5.8)$$

folosind convenția $b_1 \le ... \le b_m$. (Re-ordonarea este o proprietate fundamentală.) ■

Să presupunem că avem nevoie să exprimăm faptul că *x* criterii de diagnostic dintr-un număr dat trebuie îndeplinite, unde *x* poate fi o valoare crisp sau un criteriu vag de tipul "majoritatea", "câteva" etc. Aceasta se poate obține prin alegerea unui vector de ponderi cu toate pozițiile nule, în afara a exact *x* poziții. Dacă, spre exemplu, w^T = (0 0 0 0.25 0.25), W_w calculat cu Ecuația 5.8 va depăși pragul de 0.5 doar dacă cel puțin 2 din cele 5 criterii (modelate, după cum am precizat, de funcțiile fuzzy M_j), sunt îndeplinite în grad maxim (=1). Pentru a simplifica expunerea, vom considera de acum înainte simptomele prezente sau absente (adică, dom_j={0,1}- studiul poate fi cu ușurință extins la cazul general).

Observație. Condiția (5.7) din Definiția 5.1 nu este esențială şi o putem abandona dacă dorim mai multă putere discriminatorie: dacă sunt îndeplinite 3 din 5 criterii, algoritmul trebuie să aibă sensibilitatea necesară să considere această situație mai avantajoasă decât dacă sunt îndeplinite doar 2 din 5, în loc să asigneze valoarea 1 în ambele cazuri.

Operatorii OWA sunt cazuri speciale de integrale fuzzy, acestea din urmă fiind mai potrivite pentru calcularea scorului claselor în situațiile în care "mai multe" nu este specificat exact, şi unde criteriile de diagnostic nu sunt precis definite (în diagnoza medicală ne vom referi la aceste boli drept boli de "tip B", spre deosebire de bolile de "tip A" ce dispun de criterii ca cel din paragraful precedent).

Definiție 5.2[SouKay02] Fie C={$c_1,\ldots,c_m$} mulțimea criteriilor de decizie. Fie P(C) mulțimea părților lui C. O *măsură fuzzy* pe C este o funcție g:P(C)→[0,1] care satisface:

1. g(0)=0, g(C)=1;
2. $A \subset B \subset C \Rightarrow g(A) \le g(B)$. (5.9) ■

Definiție 5.3.[SouKay02] Fiind dată o măsură fuzzy g pe C, *integrala* fuzzy *Sugeno* a funcției p:C→[0,1] în raport cu g este definită de:

$$S_g(p(c_1),...,p(c_m)) = \bigvee_{i=1}^{m}(p(c_{(i)}) \wedge g(A_{(i)})). \quad (5.10)$$

În ecuația (5.10) indexul inferior "$_{(i)}$" indică faptul că indicii au fost permutați astfel încât $0 \le p(c_{(1)}) \le ... \le p(c_{(m)}) \le 1$, şi $A_{(i)} = \{c_{(i)},...,c_{(m)}\}$. ■

Pentru bolile de tip B, fie C_B mulțimea simptomelor relevante, $A \subseteq C_B$ şi *g(A)* ponderea normalizată a submulțimii A (i.e. suma ponderilor simptomelor din A împărțită la suma tuturor ponderilor simptomelor din C_B). Considerăm $p(m_j)$ valoarea criteriului (simptomului) m_j la un anumit pacient (sistem) *p.* Această interpretare ne permite să dăm următoarea definiție.

Definiție 5.4. Fie $d_i \in \Delta$, *p* un pacient sub observație. S*corul bolii* d_i de *tip B* la pacientul dat *p*, asociat vectorului de observații *o* este definit ca:

$$Scor(p, d_i) = S_g(M_{i_1}(o),\ldots, M_{in_i}(o)) \quad (5.11)$$

(unde g este definit folosind w_i, iar măsura fuzzy a unei mulțimi de simptome este suma normalizată a ponderilor lor, şi S_g este integrala Sugeno definită mai sus). ■

Pentru bolile de tip A, trebuie să folosim o abordare uşor diferită pentru a include ponderile în model. Funcția de decizie are aici o nouă formă (vezi Ecuația 4.16), şi trebuie ca senzitivitatea ei față de criterii să fie în concordanță cu ponderea fiecărui criteriu. Aceasta se poate obține prin cerința ca funcția de decizie să satisfacă proprietăți suplimentare.

Definiție 5.5[SouKay02] O *funcție generator* a unui operator de agregare a mulțimilor fuzzy este o funcție $f:[0,1] \to \Re \subseteq [-\infty, \infty]$ continuă şi strict monotonă. Funcția generator se spune că este un *generator crescător* (*generator descrescător*) dacă este monoton crescătoare (descrescătoare) în intervalul [0,1]. ■

Definiție 5.6[SouKay02] Pseudo-inversa unui generator *f* este funcția:

$$f^{(-1)}:[-\infty,\infty] \to [0,1],$$

care satisface:

$$f^{(-1)}(x) = \begin{cases} 1, x \in [-\infty, f(1)) \\ f^{-1}(x), x \in [f(1), f(0)], \text{ pentru un generator descrescător, şi} \\ 0, x \in (f(0), \infty] \end{cases}$$

$$f^{(-1)}(x)=\begin{cases}0, x\in[-\infty, f(0))\\ f^{-1}(x), x\in[f(0), f(1)]\\ 1, x\in(f(1), \infty]\end{cases}$$ pentru un generator crescător (f^{1} este inversa obişnuită).

(5.12)

■

Definiție 5.7[SouKay02] O t-normă T este *archimedeană* dacă satisface T(a,a)<a. Similar, o t-conormă S este archimedeană dacă satisface S(a,a)>a. ■

Propoziție. [SouKay02] Un operator de agregare continuu ν(a,b) pentru a,b∈ [0,1] poate fi reprezentat prin intermediul funcției sale generator *f* ca:

$$\nu(a,b)=f^{(-1)}(k(f(a)+f(b))),\ \text{cu } k>0,$$

dacă operatorul respectiv este o t-normă archimedeană, o t-conormă archimedeană, un operator de medie generalizat (şi câteva altele).■

Deci reprezentarea pe bază de funcție generator a unei funcții de decizie D pe bază de t-norme/ t-conorme archimediene este:

$$D(a,b)=f^{-1}(f(a)+f(b)),\ (f \text{ este funcția generator}),$$

şi se poate extinde într-o funcție de decizie ponderată prin transformarea spațiului de decizie astfel:

$$D^{w}(a,b)=f^{-1}(w_a f(a)+w_b f(b)), w=(w_a, w_b);\ (5.13)\ \text{unde } w \text{ este vectorul de ponderi.}$$

Spre exemplu, dacă alegem T-norma T(a,b)=ab pentru a exprima operația de conjuncție, funcția generator a acestei t-norme este f(x)=-ln(x), si $f^{-1}(x)=e^{-x}$, astfel încât:

$$D^{w}(a,b)=e^{-(w_a.(-\ln a)+w_b.(-\ln b))}=a^{w_a}b^{w_b}\quad (5.14)$$

În plus, dacă D^{w} este funcția de decizie ponderată bazată pe o T-normă, funcția de decizie ponderată pe bază de co-norma dual are expresia:

$$D'^{w}(a,b)\overset{def}{=}1-D^{w}(1-a,1-b)=1-f^{-1}(w_a f(1-a)+w_b f(1-b))=1-e^{w_a.\ln(1-a)+w_b.\ln(1-b)}=$$
$$=1-(1-a)^{w_a}(1-b)^{w_b}\ (5.15).$$

Aşadar, dacă $d_i\in\Delta$ este o boală de tip A şi simptomele lui d_i sunt împărțite în n_{d_i} clase $C_1,...,C_{n_{d_i}}$, criteriul de diagnostic este definit de: $d_i\equiv h_i(W_1C_1,...,W_{n_d}C_{n_d})$, unde W_i este

un operator OWA pentru "cel puțin x din $|C_k|$" definit ca mai sus, şi h_i este funcția de agregare construită pe baza unei ierarhii de t-norme şi t-conorme.

Exemplu 5.1. Reumatismul Articular Acut (RAA) este o boală de tip A. Tabloul clinic, ponderile simptomelor (definite de un expert, $\in [0,1]$) şi criteriile de diagnostic sunt prezentate mai jos.

Criteriile Jones pentru diagnoza RAA [PopIon99]

1. Manifestări majore (C_1)
   - Cardită 0.7
   - Poliartrită 0.9
   - Coree 0.7
   - Eritem marginal 0.6
   - Noduli subcutanați 0.6
2. Manifestări minore (C_2)
   - Febră 0.4
   - Artralgie 0.6
   - Factor reumatoid seric anormal 0.8
   - Proteina C reactivă crescută 0.8
3. Infecție streptococică (C_3)
   - Titru crescut de ASLO 0.7
   - Streptococ din grupa A în faringe 0.6
   - Scarlatină 0.5

Criteriu de diagnostic: prezența a 2 criterii majore, sau a unui criteriu major şi 2 minore sugerează o mare probabilitate pentru RAA, dacă a fost demonstrată infecția streptococică.

Criteriul general de diagnostic se traduce ca:

$$D_{AAR} = (W^{11} C_1 \wedge W^3 C_3) \vee (W^{12} C_1 \wedge W^2 C_2 \wedge W^3 C_3) \quad (5.16),$$

unde W^k sunt operatori OWA folosiți în definirea criteriului de diagnostic şi bazați pe următoarele ponderi, respectiv:

$$w^{11} = (0\ 0\ 0\ 0.25\ 0.25), \text{ (cel puțin 2 din 5)}$$

w^{12}=(0 0 0 0 0.5), (cel puțin 1 din 5)

w^{2}=(0 0 0.25 0.25), (cel puțin 2 din 4)

w^{3}=(0 0 0.5) (cel puțin 1 din 3),

astfel încât, dacă $C_k = \{m_1^k,...,m_{n_k}^k\}$ este mulțimea manifestărilor din clasa C_k, putem scrie:

$$W^k C_k(o) \overset{not}{=} W^k(M_1^k(o),...,M_{n_K}^k(o)) = \sum_{j=1}^{n_k} w_j^k M_j^k(o) \quad (5.17),$$

$M_j^k(o)$ fiind valoarea observată a simptomului m_j^k (0 sau 1, după cum am convenit) la pacientul dat p, o fiind vectorul de observații.

Ponderile de mai sus nu trebuie confundate cu ponderile simptomelor **observate:**

$$w_{AAR} = (w_{AAR}^1, w_{AAR}^2, w_{AAR}^3) \quad (5.18),$$

partiționate după cele 3 clase de simptome (w_{AAR}^k este vectorul de ponderi pentru simptomele observate ce aparțin clasei k).

Am ales să lucrăm cu perechea de norme duale T(a,b)=ab şi S(a,b)= a+b-ab, astfel încât prin înlocuirea operatorului "$\vee$" în 5.16, putem scrie:

D_{AAR}=1-[1-($W^{11}C_1 \wedge W^3 C_3$)]^{w13}*[1-($W^{12}C_1 \wedge W^2 C_2 \wedge W^3 C_3$)]^{w123} (5.19),

unde w13=w(C_1) +w(C_3); w123=w(C_1) + w(C_2) + w(C_3), si w(C_i) $\overset{not}{=}$ w_i este suma ponderilor simptomelor observate din clasa C_i, normalizată cu suma ponderilor tuturor simptomelor observate (am omis o din $W^k C_k(o)$ pentru a simplifica notația).

Să presupunem că avem, într-o notație simplificată, o={coree, artralgie, factor reumatoid seric anormal, streptococ din grupa A în faringe}- datele unui pacient suspect de RAA.

Atunci D_{AAR}=1-[1-(($W^{11}C_1$)^{w1}* ($W^3 C_3$)^{w3})]^{w13}*[1-(($W^{12}C_1$)^{w1}*($W^2 C_2$)^{w2}* ($W^3 C_3$)^{w3})]^{w123}; (5.20)

w1=0.26, w2=0.5, w3=0.22;

$W^{11}C_1$=0.25, $W^{12}C_1$=0.5, $W^2 C_2$=0.5, $W^3 C_3$=0.5.

Rezultatul final este D_{AAR}(o)=0.62, ceea ce înseamnă că RAA va fi într-adevăr selectată ca ipoteză de diagnostic şi trecută printr-un proces de discriminare.

5.4.Evaluarea capacității de discriminare a unui test de diagnoză

În evaluarea pasului de selecție al sistemului de asistare a deciziilor medicale DiaMed, utilizăm indexul C (echivalentul ariei de sub curba ROC-Receiver Operator Characteristic). Acesta măsoară capacitatea de discriminare a sistemului, adică de a distinge între cazuri ce fac parte din clase diferite.

Presupunem că avem acces la rezultatul real al clasificării ("gold standard") – ce poate fi 0 (pacientul este sănătos) sau 1 (pacientul suferă de boala D). Ieşirea sistemului este un număr real între 0 şi 1. Practic, se alege un prag, (de exemplu 0.5) şi clasificarea propriu-zisă se face prin compararea cu acest prag (dacă scorul obținut este > 0.5, se consideră că este bolnav).

Definiții preliminarii. În populația utilizată pentru evaluarea testului, notăm cu:

1. RP= numărul de real pozitivi (cei pe care testul i-a găsit bolnavi de boala D, şi ei sunt într-adevăr bolnavi);

2. FN= numărul de fals negativi (cei care au boala dar testul i-a inclus la sănătoşi);

3. FP= numărul de fals pozitivi (sunt în realitate sănătoşi dar testul i-a găsit bolnavi);

4. RN= numărul de real negativi (cei care sunt sănătoşi şi au fost clasificați drept sănătoşi şi de către testul pe care dorim să-l evaluăm)

- Senzitivitatea: SN = RP/(RP+FN) (capacitatea de a detecta pe cei bolnavi)
- Specificitatea: SP = RN/(RN+FP) (capacitatea de a respinge pe cei sănătoşi)
- Valoarea predictivă pozitivă: VPP = RP/(RP+FP) (probabilitatea ca boala să fie prezentă real, dacă a fost depistată de test)
- Valoarea predictivă negativă:VPN = RN/(RN+FN) (probabilitatea ca boala să fie absentă, dacă n-a fost depistată de test)
- Acuratețea testului=RN +RP

În Figura 5.1. am reprezentat pe orizontală valoarea pragului, iar pe verticală este distribuția numărului de cazuri de pacienți ale căror date au dat la testare valoarea de pe orizontală.

Se observă că specificitatea va creşte dacă pragul scade, iar senzitivitatea va creşte dacă pragul creşte. Există aşadar un compromis între aceşti doi parametri. Dacă variem valoarea pragului vom obține perechi diferite de valori, ce se reprezintă ca în Figura 5.2. Dacă luam în considerare toate valorile posibile pentru prag (între 0 şi 1), obținem curba ROC (Figura 5.3).

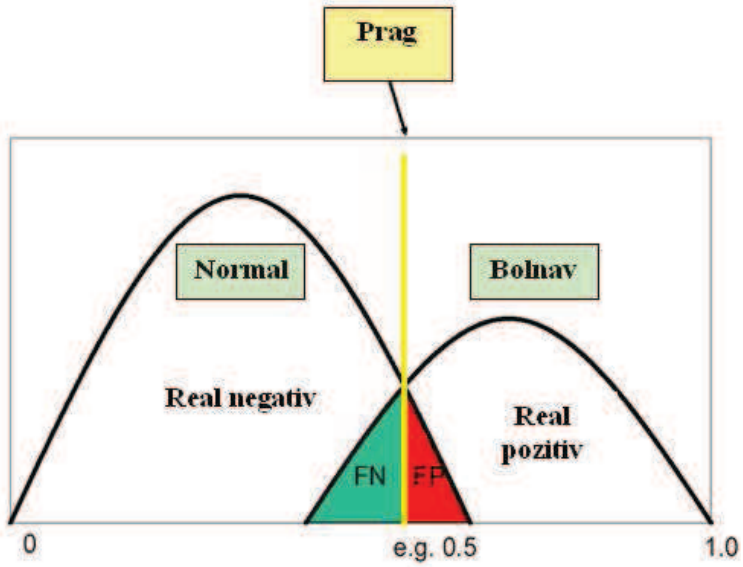


Figura 5.1.Exemplu de repartizare a celor patru clase de cazuri

Aria de sub această curbă este o estimare a puterii de discriminare a testului (este media senzitivității asupra tuturor specificităților). Dacă valoarea ariei este aproape de 0.5, testul este inutil, calitatea sa fiind cu atât mai bună cu cât aria se apropie de 1 (pentru 1 suntem în situația unei discriminări perfecte).

Prezentăm interpretarea ariei prin intermediul unui exemplu simplu.

	Sănătoşi (rezultatul real este 0)	**Bolnavi (rezultatul real este 1)**
Rezultatele testului	0.3 0.2 0.5 0.1 0.7	0.8 0.2 0.5 0.7 0.9

Se formează apoi toate perechile posibile 0-1, comparându-se estimările sistemului pentru fiecare pereche:

Sănătoşi	**Bolnavi**	
0.3	0.8	**concordant**
0.2	0.2	**discordant**
0.5	0.5	**concordant**
0.1	0.7	**concordant**
0.7	0.9	**concordant**

Perechea (0.3, 0.8) este concordantă întrucât unui caz real bolnav i-a fost asociat de către test un scor mai mare decât unui caz real sănătos. Se obțin în final 18 perechi concordante, 4 discordante şi 3 egalități. Avem:

$$Index\_C = \frac{Concordante + \frac{1}{2} Egalitati}{Total\_perechi} = 0.78 \quad (5.20)$$

adică s-a folosit un test destul de bun.

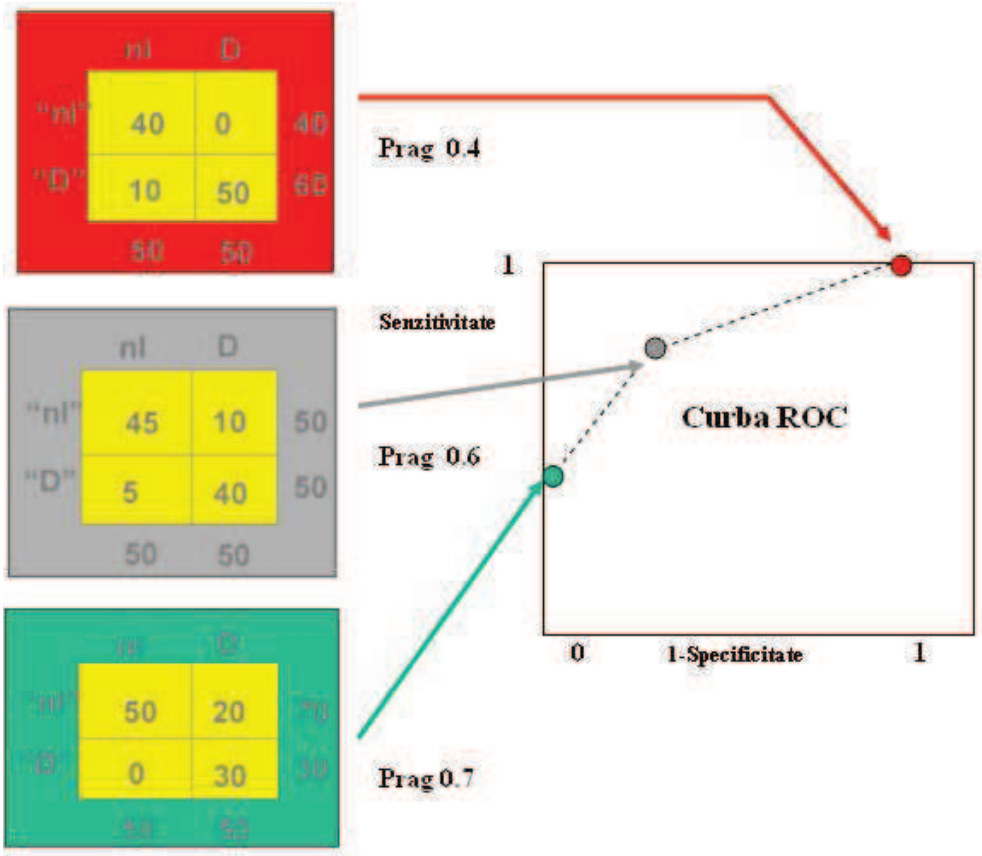


Figura 5.2.Calculul valorilor pentru curba ROC, pentru valori diferite ale pragului

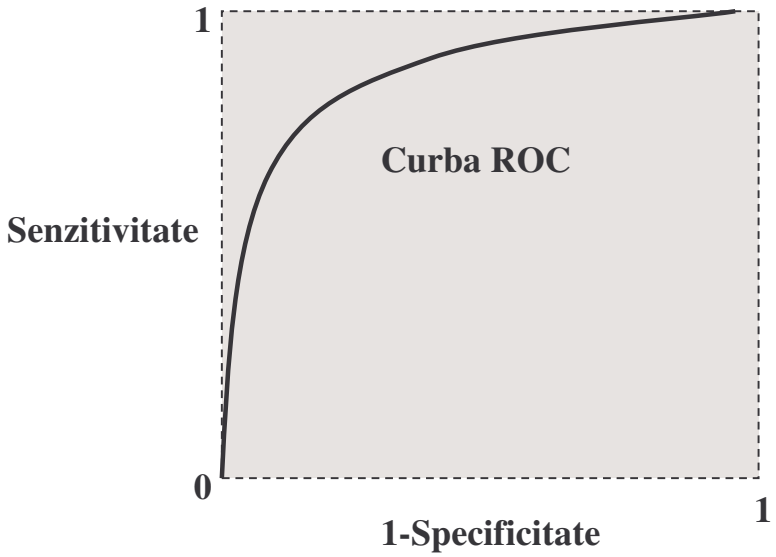


Figura 5.3.Curba ROC

5.5.Clasificare prin decizie fuzzy

Am ales pentru exemplificare o pereche de boli care pune de obicei proleme de recunoaştere (*ciroza* şi *insuficienţa cardiacă*) întrucât există situaţii când insuficienţa cardiacă se manifestă prin simptome asemănătoare cirozei şi este prin urmare confundată cu aceasta. Am comparat indexul C obţinut prin clasificarea cu o funcţie de decizie fuzzy construită după modelul descris în [Mun05], pentru perechi diferite de norme duale (utilizate în calculul integralei fuzzy). Indicele de calibrare din ultima coloană măsoară sensibilitatea calculului (este o măsură simplă de genul sumă de diferenţe, care stă la baza majorităţii metodelor de calibrare- i.e. aducere a scorului bolii cât mai aproape de probabilitatea reală).

S-NORMA	T-NORMA	INDEX C	INDICE DE CALIBRARE
max	min	0.9856	546.6659
a+b-ab	ab	0.9856	1186.5055
min(a+b, 1)	max(a+b-1, 0)	0.9856	1632.4998

Tabel 5.1. Indexul C şi calibrarea pentru testul "**ciroză**"

În urma acestui gen de experimente ne-am oprit asupra perechii (max, min) pentru sistemul final. Presupunem că superioritatea acestei perechi de norme a provenit din faptul că sunt mai puţin sensibile la ponderile simptomelor, care, probabil, nu au fost setate foarte corect. Au fost evidente avantajele abordării fuzzy: integrarea naturală a cunoştinţelor expert în model, modelarea facilă a incertitudinii.

S-NORMA	T-NORMA	INDEX C	INDICE DE CALIBRARE
max	min	0.956	392.1201
a+b-ab	ab	0.9088	908.2604
min(a+b, 1)	max(a+b-1, 0)	0.9064	1243.3226

Tabel 5.2. Indexul C şi calibrarea pentru testul "**insuficienţă cardiacă congestivă**"

Principalul dezavantaj pe care l-am întâlnit aici reprezintă o problemă încă deschisă în comunitatea ştiinţifică: lipsa unui indice riguros fundamentat ştiinţific de evaluare a puterii de discriminare în diagnosticul cu răspuns (posibil) multiplu. Dacă ar fi să abordăm problema în totalitate, nu doar pentru o anumită boală, cum ar trebui modificat/ adaptat indexul C? Cum s-ar redefini însăşi noţiunea de discriminare în noul context? La aceste întrebări deocamdată nu există un răspuns clar.

5.6.Concluzii şi contribuţii

Cheia reprezentării folosite mai sus este să privim simptomele drept criterii fuzzy ce definesc un diagnostic, în măsura în care sunt îndeplinite. Gradele de potrivire $D_i^{w_i}(o)$ între vectorul de observaţii şi criteriul complex ce defineşte diagnosticul d_i, conduc la o ierarhizare a ipotezelor de diagnostic, pe baza evidenţelor particulare ale unui caz dat. Mulţimea finală a diagnosticelor se determină prin aplicarea unui prag de semnificaţie, care ar trebui să fie altul pentru fiecare boală în parte (pentru a evita dezavantajele variabilităţii sumei totale de ponderi de la o boală la alta).

Considerăm că modelul de decizie fuzzy folosit de noi în DiaMed pentru selecţia ipotezelor de diagnostic constituie o alternativă viabilă şi mai eficientă a abordării fuzzy prezentate în secţiunea anterioară. Cu ajutorul său, fiecare bază de reguli fuzzy corespunzătoare unei anumite variabile de ieşire *y* se poate rescrie ca o funcţie de decizie fuzzy după modelul descris. Comparând cu abordarea din Secţiunea 5.2, în loc să agregăm scorul v_j al unei boli din R reguli, îl putem calcula direct pe baza funcţiei de decizie asociată bolii respective.

Astfel, dacă avem mai multe variabile de ieşire- (*m* ieşiri), selectarea ipotezelor de defect va avea nevoie doar de *m* funcţii de decizie fuzzy. În plus, aceste funcţii sunt formate cu operatori de agregare fuzzy mai complecşi, mai naturali, mai nuanţaţi şi mai uşor de interpretat şi citit (folosind, spre exemplu, operatorii compensatori, integralele fuzzy etc.), decât o bază formată în exclusivitate cu operatorul AND.

Un mare avantaj al funcţiilor de decizie fuzzy descrise mai sus este dat şi de faptul că sunt mai uşor de construit decât bazele de inferenţă fuzzy, în condiţiile în care există cunoştinţe expert suficient de clare şi de precise. Pentru aceasta însă este necesar să avem un model corespunzător adaptat: nu vom mai considera valoarea "mică" sau "mare" a

unui anumit element drept un defect în sine, ci este necesar să reprezentăm valoarea elementului ca o funcție de alte elemente din sistem, de care depinde direct sau indirect (vezi Figura 8.1).

În final, în abordarea noastră, în loc să agregăm grade de realizare diferite ale valorilor fuzzy (mare, mic etc) pentru o singură variabilă de ieşire, obținem direct un număr pe care îl considerăm mare sau mic nu atât raportat la o scară absolută, cât mai mult raportat la un anumit context (în relație cu valorile obținute de celelate variabile de ieşire)

În plus, avantajul major al funcțiilor de decizie definite mai sus față de legea combinării evidențelor din CHECK, stă în faptul că pot exprima cu acuratețe criterii vagi de mare diversitate (de exemplu, *majoritatea*, *cel puțin x din n*, *un număr semnificativ din* etc.)

Aşadar, principalele avantaje ale abordării constau în simplitatea şi eficiența modului de calcul (ce ia în considerare şi specificitatea fiecărui simptom în definițiile bolilor), şi în focalizarea rapidă a căutării spre direcții relevante. Dezavantajele ce rezultă din lipsa facilităților explicative şi din ignorarea deliberată a interacțiunilor posibile între ipoteze vor fi înlăturate de metodele prezentate în Capitolele VI, VIII.

VI.Sisteme de argumentare directă şi algoritmi de satisfacere a constrângerilor în modelarea raţionamentului nemonoton

6.1.Introducere

În secţiunea prezentă, aprofundăm tehnicile de raţionament nemonoton ce vor constitui suportul teoretic pentru modelarea nivelului de discriminare al sistemului hibrid DiaMed. Sunt introduse sistemele de argumentare directă, cu definirea unor noţiuni elementare, şi cu sublinierea avantajelor şi motivaţiilor care au susţinut dezvoltarea acestor sisteme (ce aparţin logicii informale). Este prezentată semantica argumentativă a admisibilităţii, pe care o folosim în DiaMed cu rolul unui mecanism de menţinere a consistenţei (aceasta din urmă constituind una din noţiunile necesare în definirea abducţiei). Pentru calculul efectiv al acestei semantici am optat pentru algoritmii specifici problemelor de satisfacre a constrângerilor (CSP), plecând de la faptul că admisibilitatea se poate rescrie în termenii verificării satisfiabilităţii unei formule logice, pentru care este natural să folosim algoritmii menţionaţi.

Secţiunea discută, deasemenea, versiunea dinamică a problemelor CSP (DCSP), mai potrivită unei probleme al cărei context se modifică dependent de teste, cum este diagnoza medicală. Ca implementare efectivă pentru o problemă de tip DCSP, am ales algoritmul backtracking dinamic al lui Ginsberg, întrucât este una din alternativele cele mai eficiente şi mai adecvate. Secţiunea prezintă şi o versiune de adaptare efectivă a backtracking-ului dinamic pentru DCSP [VerSchi], adaptare pe care o vom modifica la rândul nostru în cadrul nivelului de discriminare între diagnostice din DiaMed.

6.2.Sisteme de argumentare directă

6.2.1.Introducere

Logica de ordinul întâi, deşi reprezintă un formalism puternic de reprezentare, şi-a demonstrat limitările în privinţa multor probleme de Inteligenţă Artificială. Aceste limitări îşi au originea în două cauze principale: monotonicitatea şi capacitatea redusă de a reprezenta incertitudinea, de care este legat şi inconvenientul semidecidabilităţii.

Definiţie. Un mecanism de inferenţă este *monoton,* dacă pentru o teorie dată Δ avem:

$$T(\Delta \cup S) \supseteq T(\Delta), \quad (6.1)$$

oricare ar fi setul de propoziţii S ($T(\Delta)$ este mulţimea de propoziţii derivabile din Δ).

(O propoziţie adevarată într-o teorie rămâne adevărată după adăugarea de noi propoziţii la teoria respectivă).

Inferența nemonotonă este opusul celei monotone (întrucât cunoştințe noi pot contrazice ipoteze vechi), şi este mijlocul de modelare a incertitudinii cu mijloace ale logicii (formale sau informale) [Cla90]. Conceptul a devenit necesar în momentul în care s-a înțeles că incertitudinea probabilistă nu este suficientă pentru reprezentarea problemelor lumii reale. Spre exemplu:

"Cele mai multe păsări zboară."

este o propoziție probabilistă, bazată pe statistici calculate asupra unor populații de păsări. Se poate ca un sistem probabilist incert să rămână monoton, după modelul rețelelor bayesiene. Dacă însă vom dori să considerăm exemplul:

"Păsările zboară, dacă nu se poate demonstra contrariul.",

avem deja o afirmație pur logică, ce necesită o reprezentare distinctă a incertitudinii pe care o reprezintă.

Non-monotonicitatea poate avea expresii diferite în cadrul diverselor formalizări (unele dintre ele mai expresive, altele -mai eficiente), şi a fost abordată în programarea logică (de exemplu, completarea predicatelor, programarea logică generală, programarea logică disjunctivă, semanticile stabilă şi admisibilă -[BarGel94], [Dix55], [PerApAl93]), sau în circumscripția lui McCarthy [Cart80] (o formă mai generală de completare). Exemplele enumerate fac parte din "sistemele de argumentare indirectă" [Kon88]. Noțiunea de **argument** apare în cadrul lor implicit, în sensul general de "demonstrație –formală sau informală- a adevărului unei formule pe baza adevărului premizelor". Există însă sisteme în care argumentele sunt obiecte explicite ale formalismului – aşa numitele "sisteme de argumentare directă" [Kon88]. În cazul din urmă, non-monotonicitatea se reduce la o relație de atac între structuri monotone- argumentele ([BDKT97], [Pol02], [Vre92]). Vom justifica în continuare preferința noastră pentru acestea.

Tehnicile de completare a teoriei permit deducerea de ipoteze globale pornind de la ceea ce nu a putut fi dedus în cadrul teoriei. Pentru a reprezenta ipotezele implicite (o propoziție este implicit validă dacă nu există evidențe împotriva ei), se folosesc reprezentări mai complexe, -cum ar fi logica nemonotonă a lui McDermott şi Doyle ([DerDoy80][Der82]), sau logica implicită a lui Reiter [Rei80]. În conformitate cu aceste formalisme, o concluzie este acceptată dacă nu introduce inconsistențe la nivelul global al teoriei. În afara faptului că testarea consistenței este doar semi-decidabilă şi foarte costisitoare computațional, abordările enumerate suferă de lipsa unui mecanism determinist de selectare a unei extensii particulare (=o mulțime consistentă de concluzii), atunci când se generează mai multe. Rezolvarea unui conflict (atunci când există argumente pro şi contra pentru aceeaşi concluzie) este, aşadar, o sarcină dificilă pentru "sistemele de argumentare indirectă" [Kon88]. Sistemele de argumentare directă, în schimb, dispun de o reprezentare explicită a argumentelor, care reduce rezolvarea conflictelor la o relație de înfrângere între aceste structuri.

Semantica sistemelor de argumentare directă înlocuieşte teoria modelelor a lui Tarski (semantica teoretică pe bază de model) substituind *"adevărat"* cu *"justificat"*: o concluzie este justificată dacă există un argument neînfrânt ce o susţine în contextul evidenţelor actuale. Această înlocuire a teoriei modelelor se bazează pe concluziile mai multor cercetători (Pollock, Loui, Vreeswijk), că raţionamentul nemonoton are nevoie de o semantică diferită. După cum observau autorii în [PraVre], teoria modelelor a fost folosită în mod tradiţional pentru a defini înţelesul limbajelor logice pornind de la *realitate*: înţelesul unor simboluri logice se defineşte plecând de la cum arată lumea, dacă o expresie cu aceste simboluri este adevărată, şi defineşte consecinţa logică cercetând ce altceva este adevărat, pornind de la premizele date. Prin contrast, atunci când discutăm despre "adevărat în general, dacă nu sunt îndeplinite anumite excepţii", această clasificare normal/excepţional cuprinde deja condensată în ea, implicit, o parte de demonstraţie (de raţionament). Noţiunile de atac şi înfrângere, centrale în raţionamentul invalidabil, nu sunt 'propoziţionale' întrucât înţelesul lor nu este dat de corespondenţa unei propoziţii cu lumea [PraVre], ci de relaţiile între propoziţii, stabilite dependent de un context dinamic de evidenţe. S-a ajuns astfel la semantica argumentativă amintită.

Un avantaj în plus din punctul de vedere al lucrării de faţă stă în faptul că genul de raţionament de diagnoză de care avem nevoie presupune revizuirea iterativă a încrederilor, pentru care putem folosi natural formalismul sistemelor de argumentare directă (conceput special pentru simularea cât mai fidelă a raţionamentului uman în faţa unor situaţii complexe). Acest formalism este în plus adaptabil interactivităţii necesare pentru a lua în calcul testele efectuate secvenţial, generarea de ipoteze alternând cu obţinerea de noi informaţii.

Definiţie[Vre92]. Un *sistem de argumentare abstract A* este un triplet (L,R,≤), unde L este un limbaj, R este o mulţime de reguli de inferenţă, şi "≤" este o relaţie de ordine reflexivă şi tranzitivă pe mulţimea argumentelor.

Definiţie[GovMah00]**.** Un *argument* pentru literalul *p* este un arbore de demonstrare (folosind derivare monotonă) pentru *p*, construit pe baza regulilor de inferenţă din R.

Cele mai utilizate tehnici de rezolvare a conflictelor sunt: "principiul specificităţii" (Toulmin: "informaţia mai specifică suprascrie informaţia mai generală" [Tou58], metode dependente de domeniu, metode numerice (acestea din urmă asignează un numar –spre exemplu, tăria celei mai slabe legături- fiecărui argument [Pol02])).

Definiţie[Vre92]**.**O *mulţime de bază* este o submulţime finită compatibilă a lui L.

O mulţime de bază conţine "informaţie ireductibilă" –ea constituie punctul de plecare într-un raţionament forward, şi punctul final al raţionamentului backward.

Partea interesantă a sistemelor directe de argumentare, din punctul de vedere al diagnozei, este faptul că abducţia poate fi modelată mai eficient în cadrul lor decât într-un cadru ipotetico-deductiv. Noţiunea de explicaţie este înlocuită natural cu cea de argument, anularea ipotezelor cu cea de rezolvare a conflictelor între argumente, şi

observațiile inițiale pot fi considerate drept o mulțime de bază. Vom utiliza în lucrarea de față un cadru abductiv bazat pe argumente atât pentru fundamentarea şi discriminarea ipotezelor de diagnostic, cat şi pentru menținerea coerenței concluziilor finale.

Un cadru argumentativ pentru generarea de explicații a fost deja dezvoltat sub denumirea WOZ(Wizard of OZ) [ShaMus99], plecând de la structura argumentelor propusă de Toulmin [Tou58]. Componenta de generare a explicațiilor a fost integrată într-un sistem multiagent de decizie, cu aplicații în recomandările terapeutice, rezultatul final numindu-se ATHENA DSS [Sha00]– un instrument puternic de asistare a personalului medical în deciziile sale, şi singurul ce foloseşte această metodologie până acum, din cunoştințele noastre. Abordarea curentă duce ideea un pas mai departe: structura argumentelor a lui Toulmin nu este suficient de expresivă pentru a surprinde întreaga complexitate a raționamentului de diagnoză (spre exemplu, nu putem reprezenta excepțiile la excepții), astfel că vom propune în continuare o schemă argumentativă mai complexă. În plus, scopul principal al autorilor sistemului ATHENA DSS este implementarea bazată pe agenți, nu detaliile raționamentului nemonoton, acestea din urmă fiind tratate într-o manieră simplificată.

Ne propunem în continuare o succintă introducere a unor cadre de lucru argumentative, ale căror elemente ne-au inspirat în abordarea proprie.

6.2.2.Abordarea de punct fix

Sistemul de argumentație al lui Dung

Dung a fost unul din pionierii acestui domeniu. El a propus un sistem de argumentare care abstractizează structura internă a unui argument. Majoritatea abordărilor raționamentului nemonoton din inteligența artificială şi programarea logică pot fi privite drept cazuri particulare ale acestui sistem. Cadrul său conține o noțiune primitivă de argument şi o relație binară de înfrângere:

Definiție.[**Dun95**]

1. Un *cadru de argumentare* AF (“Argumentation Framework”) este o pereche

$$AF=<AR, atacuri>, \quad (6.2)$$

unde AR este o mulțime de argumente, şi *atacuri* este o relație binară pe AR: $atacuri \subseteq AR \times AR$;

2. Un AF este *finit* dacă fiecare argument din AR este înfrânt de către cel mult un număr finit de argumente;

3. O mulțime S de argumente este *fără conflicte* dacă nu există argumentele A, B în S astfel încât A atacă B.

Noţiunea de *acceptabilitate* este centrală în argumentare, întrucât defineşte de fapt relaţia argumentării cu raţionamentul nemonoton. Nu există deocamdată o metodă de calcul universală a acestei noţiuni.

Definiţie([Dun95]). Un argument A este *acceptabil* relativ la o mulţime S de argumente ddacă orice argument care înfrânge A este înfrânt de un argument din S.

Definiţie([Dun95]). O mulţime de argumente S fără conflicte este *admisibilă* ddacă fiecare argument din S este acceptabil în raport cu S.

Semantica admisibilă (credulă) îmbunătăţeşte mult semantica stabilă (sceptică- atacă tot ce nu conţine), care nu poate surprinde înţelesul intuitiv al unor cadre de argumentare, şi al cărei caracter *global* o face intratabilă în general. Prin contrast, extensii admisibile există pentru orice cadru de argumentare, indiferent de structura de atac asociată, şi, în plus, admisibilitatea este mai tratabilă computaţional întrucât este *locală*: se construieşte mai întâi o mulţime de ipoteze care implică monoton propoziţia pe care dorim să o justificăm, apoi se augmentează această mulţime cu "apărări" împotriva oricărui atac posibil. Pe baza acestor observaţii, propunem o definiţie originală şi naturală a unei mulţimi de diagnostic multiplu.

Definiţie. Un *diagnostic multiplu* este o mulţime admisibilă de ipoteze care acoperă toate observaţiile, şi este minimală cu această proprietate.

Vom folosi această definiţie în algoritmul din Capitolul VIII.

Exemplu. Fie 3 argumente A={leucemie}, B={nivel PT,PTT normal}, C={medicamente care afectează nivelul PT,PTT}. Relaţia de atac este dată de graful: A←B←C. Atunci A este acceptabil în raport cu C, C fiind o apărare a lui A faţă de atacul B. Deci {A,C} este o mulţime admisibilă, şi în acest caz "leucemie" este un diagnostic posibil în contextul definit de C.

Definiţie([Dun95])Semantica ground a lui Dung. Fie Args o multime de argumente ordonate cu ajutorul unei relaţii binare de atac, fie $S \subseteq$ Args. Atunci operatorul F este definit astfel:

$$F(S)=\{A \in \text{Args} | A \text{ este acceptabil în raport cu } S\}. \quad (6.3)$$

Definiţie([Dun95]). Un argument este *justificat* ddacă aparţine celui mai mic punct fix al operatorului F.

Propozitie(Dun95). Fie următoarea secvenţă de argumente:

- $F^0=0$;
- $F^{i+1}= \{A \in \text{Args} \mid A \text{ este acceptabil în raport cu } F^i\}$.

Atunci:

1. Toate argumentele din $\cup_{i=0}^{\infty}(F^i)$ sunt justificate.

2. Dacă fiecare argument este anulat de cel mult un număr finit de argumente, atunci un argument este justificat ddacă se află în $\cup_{i=0}^{\infty}(F^i)$.

6.2.3.Abordarea recursivă

Definiție.Argumente defensibile şi suprascrise. [PraVre]

- Un argument este *suprascris* ddacă nu este justificat şi fie se auto-anulează, fie unul dintre subargumentele sale proprii sunt anulate de un argument justificat.

- Un argument este *defensibil* ddacă nu este justificat şi nu este suprascris.

Definiție.Argumente justificate recursiv[PraVre]. Un argument A este *justificat* ddacă:

1. A nu se autoanulează;

2. toate subargumentele proprii ale lui A sunt justificate;

3. toate argumentele care anulează A se auto-anulează, sau au cel puțin un subargument propriu suprascris.

În abordarea credulă (care permite stări multiple pentru argumente), argumentele contradictorii de tărie egală vor induce două stări alternative.

Definiție[PraVre].O *asignare de stare* unei mulțimi X de argumente ordonate printr-o relație binară de înfrângere este asignarea fiecărui argument fie a stării "in", fie a stării "out", astfel încât să fie satisfacute condițiile:

1. Un argument este "in" dacă toate argumentele care îl anulează sunt "out";

2. Un argument este "out" dacă este anulat de un argument "in".

Remarcă. Atât în abordarea credulă, cât şi în cea sceptică, ciclurile impare de înfrângere nu au asignare de stare.

În [BDKT97] autorii propun un cadru abstract care permite o viziune unificată a diferitelor formalisme nemonotone în general, şi a sistemelor de argumentație în special.

Definiție[BDKT97]. Se numeşte *sistem deductiv* perechea (L,R), unde:

- L este un limbaj formal alcătuit dintr-o mulţime numărabilă de propoziţii şi
- R este o mulţime de reguli de inferenţă de forma $\frac{\alpha_1,...,\alpha_n}{\alpha}, \alpha_1,...,\alpha_n \in L, n \geq 0.$

Orice mulţime de propoziţii $T \subseteq L$ se numeşte o *teorie*; $Th(T)=\{\alpha \in L | T \vdash \alpha\}$.

Autorii argumentează că toate logicile nemonotone dezvoltate până în prezent pot fi de fapt privite ca metode de a extinde o teorie dată cu ipoteze, în condiţiile în care nu există raţiuni care să demonstreze contrariul acestor ipoteze (noţiunea de contrarietate fiind definită diferit, de la caz la caz).

Definiţie[BDKT97]. Fiind dat un sistem deductiv (L,R), un *cadru bazat pe ipoteze* în raport cu (L,R) este format de tuplul <T,Ab, $^-$>, unde:

- $T,Ab \subseteq L$, $Ab \neq 0$,
- "$^-$" este o relaţie din Ab în L ($\bar{\alpha}$ este contrariul lui α)

(Ab este mulţimea de ipoteze care poate extinde teoria T).

Definiţie[BDKT97]. Fiind dat un cadru bazat pe ipoteze <T,Ab, $^-$> şi $\Delta \subseteq Ab$, Δ se numeşte *fără conflicte* ddacă pentru toţi $\alpha \in Ab$, $T \cup \Delta \nvdash \alpha, \bar{\alpha}$.

Noţiunile fundamentale (atac, aparare, tipuri de semantici) definite mai sus se rescriu în cadrul acestui formalism după cum urmează.

Definiţie[BDKT97]. Atac împotriva unei formule α. Se spune că Δ *atacă* α (este un argument împotriva lui α) dacă există o derivare monotonă a lui $\neg\alpha$ din $T \cup \Delta$: $T \cup \Delta \vdash \neg\alpha$.

Avantajul este că folosind noţiunea de atac se pot defini semantici declarative şi procedurale teoretic-argumentative pentru diferite formalisme nemonotone, asemănătoare sau chiar echivalente cu cele din programarea logică, dar într-o exprimare mai simplă, mai intuitivă, şi care depăşesc unele din problemele de acolo.

Noţiunea duală celei de atac este cea de *apărare* faţă de un atac.

Definiţie[BDKT97]. O mulţime de ipoteze Δ *apără* o ipoteză α ddacă pentru orice Δ' mulţime închisă (din punctul de vedere al operaţiei de deducţie în (L,R)) de ipoteze, dacă Δ' atacă α, Δ atacă Δ' - Δ (Notaţie : Def(Δ)={α| Δ apără (Defends) α.})

Un sistem de ipoteze este plat (flat) dacă orice subset al său este închis (o ipoteză nu e deductibilă din altele). Programele logice şi teoriile implicite sunt plate.

Semantica admisibilă (credulă)[BDKT97]. O mulțime închisă de ipoteze $\Delta \subseteq Ab$ este *admisibilă* ddacă:

- Δ nu se atacă pe sine, (i.e. nu conține conflicte) şi
- pentru orice mulțime închisă de ipoteze $\Delta' \subseteq Ab$ dacă Δ' atacă Δ, atunci şi Δ atacă Δ' (se apără față de toate atacurile). ($\Delta \subseteq Def(\Delta)$)

Semantica admisibilă a fost introdusă de Dung pentru programarea logică şi este de preferat din punctul de vedere al teoriei argumentației (semantica stabilă fiind prea restrictivă, iar cea naivă -mulțimi maximale fără conflicte- prea liberală).

Semantica completă [BDKT97](credulă). Un set ipoteze este *complet* când conține tot ce apără. (Δ=Def(Δ))

Semantica preferată (credulă maximală). Un set de ipoteze se numeşte *preferat* dacă este maximal admisibil.

Semantica bine fundamentată[BDKT97](sceptică, minimalistă). Mulțimea de ipoteze bine-fundamentată a unui program logic este intersecția tuturor mulțimilor complete de ipoteze.

Semantica stabilă (sceptică)[BDKT97]. O mulțime de ipoteze este *stabilă* dacă este închisă, fără conflicte (nu se atacă pe sine) şi atacă orice ipoteză α pe care nu o conține. O extensie stabilă a unui cadru bazat pe ipoteze E este dată de: $E=\{q | P \cup \Delta_E \vdash q\}$, unde $\Delta_E=\{not\ p \in Ab |\ p \notin E\}$.

Teorema 1[BDKT97]. M este un model stabil al unui program logic P ddacă există o extensie stabilă E a cadrului bazat pe ipoteze corespunzător astfel încât $M=E \cap HB$ (HB este baza Herbrand a programului P: **H**erbrand **B**ase).

Un set stabil de ipoteze este întotdeauna şi preferat. Un set preferat şi plat de ipoteze este complet (stabil$\rightarrow$ preferat $\overset{flat}{\rightarrow}$ complet).

Teorema 2[BDKT97]. Pentru cadrele stratificate (adica plate şi graful de atacuri nu conține drumuri infinite-de exemplu, cicluri) semanticile stabilă, bine-fundamentată şi preferată există, coincid şi sunt unice.

Autorii propun în [DKT97] şi o procedură de calcul abstractă (poate fi aplicată la orice raționament de tip invalidabil ce poate fi definit în termeni de teoria argumentației) pentru semantica admisibilă, în forma unui program logic. Această procedură formalizează strategia informală prezentată în unele lucrări despre argumentare (concluziile trebuie susținute de o rațiune, rațiunea trebuie apărată de contraatacuri) şi este, de fapt, o adaptare

a unei proceduri dezvoltate iniţial pentru semantica stabilă (care este intratabilă în cazul general) .

Se consideră că A atacă Δ dacă atacă un $\alpha \in \Delta$.

Procedura de calcul pentru semantica admisibilă este dată în forma unui meta-program logic. Clauza de nivel superior este cea care defineşte predicatul admisibil: adm(Δ_0, Δ): Δ este un superset admisibil al lui Δ_0. Clauza de nivel inferior defineşte relaţia de apărare faţă de un atac: apără(D, Δ) (D atacă toate atacurile închise împotriva lui Δ =închisă).

Meta-programul logic care defineşte procedura este:

$$adm(\Delta_0, \Delta) \leftrightarrow [(\Delta_0 \subseteq \Delta) \wedge (\Delta \text{ admisibilă})]$$

$$apără(D, \Delta) \leftrightarrow \forall A \subseteq Ab[[(A \text{ atacă } \Delta) \wedge închisă(A)] \rightarrow D \text{ atacă } (A - \Delta)]$$

Aşadar: adm(Δ_0 , Δ) $\leftrightarrow$ [($\Delta_0 \subseteq \Delta$) $\wedge$ apără(Δ, Δ) $\wedge$ închisă(Δ)].

Ideea de interpretare este că Δ_0 trebuie extinsă la Δ care o apără. Dacă programului îi este prezentată o interogare de forma $\leftarrow$ adm(Δ_0, X), acesta adaugă repetat apărări mulţimii iniţiale până ajunge la o mulţime care se apără pe sine de orice atac posibil. Într-o problemă de diagnoză, Δ_0 poate fi considerat ca un argument (format din simptome) suport pentru un anumit diagnostic, iar Δ ar fi simptomele/stările intermediare care susţin diagnosticul respectiv şi îl apără în cadrul unui diagnostic diferenţial. Mulţimea de apărări adăugată pe parcursul procedurii ar defini natural planul de teste suplimentare necesare. Se observă că procedura nu este deterministă (construcţia apărărilor), ceea ce se încadrează intuitiv şi în specificul diagnozei: există multe configuraţii alternative care conduc la acelaşi diagnostic-este suficientă prezenţa uneia singure pentru confirmarea acestuia (în conformitate cu specificul credul al semanticii admisibile).

6.2.4.Rafinarea semanticii preferate folosind "apărări minimale"

Rafinarea "min-def" ("minimal defence") propusă în [CayDou02] a semanticii preferate pleacă de la intenţia de a modela situaţii în care există două tipuri de argumente: *restricţionate* si *nerestricţionate*, şi în utilizarea maximală a celor nerestricţionate şi minimală a celor restricţionate (spre exemplu, în aplicaţiile juridice, pot exista martori pentru care este periculos să se implice şi să depună mărturie –pentru că ar fi, de exemplu, ameninţaţi de mafie-; în cazul diagnozei medicale, ar fi natural să dorim restricţionarea argumentelor ce folosesc teste invazive/costisitoare).

Fie AF=(A,R) un cadru de argumentare.

Partiționarea mulțimii de argumente a unui cadru de argumentare[CayDou02].
Partiție de tipul I
Este situația în care mulțimea argumentelor A este nediferențiată, fiind formată doar din argumentele F ale unui agent dat şi mulțimea complementară A\F.

Partiție de tipul II.
Fie o mulțime $F=F_u \cup F_r$, $F_u \cap F_r=\phi$ (F_u = mulțimea argumentelor nerestricționate- unrestricted, F_r = mulțimea argumentelor restricționate- "restricted");

$$A= F_u \cup F_r \cup (A\setminus(F_u \cup F_r)); \quad (6.4)$$

Acest tip de cadru de argumentare se notează cu AF_{Fru}.

Definiție[CayDou02]. Fie AF_{Fru} un cadru de argumentare partiționat tip II. Fie S_1, S_2 două submulțimi ale lui F. S_2 este *mai bun* decât S_1 ($S_1 \prec S_2$) ddacă $S_{1u} \subset S_{2u}$, sau $S_{1u}=S_{2u}$ si $S_{2r} \subseteq S_{1r}$ (S_{ir}, S_{iu} reprezintă argumentele restricționate, respectiv nerestricționate din mulțimea de argumente S_i).

Definiție[CayDou02]. Fie $S \subseteq F$. S este *restricționat-admisibilă* ddacă:
1. S este admisibilă şi
2. $\forall x \in S_r, \exists y \in S_u$, astfel încât x este un apărător individual al lui y.

Definiție[CayDou02].S este o *extensie min-def* a cadrului AF_{Fru} ddacă S este $\prec$-maximală între mulțimile restricționat-admisibile ale lui AF_{Fru}.

Am prezentat aici aceste noțiuni întrucât ele au constituit sursa de inspirație în definirea şi utilizarea relației de ordine "$\lhd$" din Capitolul VIII.

6.3.Algoritmi DCSP

6.3.1.Introducere

Constrângerile sunt o metodă naturală de reprezentare a cunoştințelor în foarte multe domenii, după cum observa autorul în [MigI01]. O problemă bazată pe constrângeri conține trei componente principale:

Definiție[Gin96]. O problemă de satisfacere a constrângerilor (V,D,c) reprezintă o mulțime V de variabile; pentru fiecare $v \in V$, există o mulțime posibilă de valori D_v ale acestei variabile. c este o mulțime de constrângeri, fiecare fiind o pereche (W, K), unde $W=(v_1,\ldots v_k)$ este o submulțime ordonată a lui V, şi K este o submulțime a lui $D_{v1} \times \ldots \times D_{vk}$.

Constrângerile pot fi privite ca "o relație logică între variabile, exprimând o mulțime admisibilă de combinații de valori", şi reprezintă o tehnică fundamentală de inferență. Concret, ele pot lua forma unor egalități/inegalități. Raționamentul pe bază de

constrângeri este declarativ şi permite separarea reprezentării cunoştinţelor (exprimarea soluţiilor problemei sub formă de constrângeri), de rezolvarea efectivă a constrângerilor, ceea ce a permis abordarea problemei cu tehnici foarte variate.

Aplicaţiile metodelor CSP se întind de la recunoaşterea imaginilor sau probleme de diagnoză în inteligenţa artificială[Wal75], până la programarea logică (unde înlocuirea algoritmului de unificare al lui Robinson cu metodele de rezolvare a constrângerilor au dat naştere la Programarea Logică pe bază de Constrângeri –CLP(Constraint Logic Programming)- [JafLas87], cu toate derivatele sale ulterioare), şi până la rezolvarea de probleme combinatoriale NP-dificile în domeniile cercetărilor operaţionale şi matematicii discrete. Implicarea recentă a tehnicilor inteligente (reţele neurale Hopfield în care punctele de stabilitate corespund unor soluţii ale problemelor bazate pe constrângeri discrete [LeeTam], algoritmi genetici pentru focalizarea căutării soluţiilor, în care constrângerile se reprezintă ca funcţii de penalizare, reprezentarea fuzzy a constrângerilor pentru relaxarea problemelor fără soluţii şi obţinerea de soluţii parţiale [MigI01]ş.a.) reprezintă o garanţie a faptului că domeniul este încă în plină dezvoltare şi promite mai mult decât a reuşit deja să ofere, în special în probleme în care alte metode au eşuat, şi ariile sale de aplicabilitate continuă să se extindă.

6.3.2.Backtracking dinamic pentru CSP

Algoritmii "clasici" folosiţi în abordarea problemelor de satisfacere a constrângerilor se împart în două categorii: *algoritmi de preprocesare*, în care constrângerile sunt folosite activ pentru a reduce spaţiul de căutare prin restrângerea prealabilă a domeniului variabilelor (algoritmii de arc-consistenţă cu variantele de îmbunătăţire a nivelului de consistenţă prin considerarea simultană a unui grup de constrângeri- Path-consistency), şi *algoritmi de rezolvare* efectivă prin căutare, în care constrângerile sunt folosite pasiv, pentru testarea soluţiilor generate deja (aceştia din urmă sunt o paradigmă a tehnicilor "generează şi testează").

Una dintre metodele cele mai folosite de căutare sistematică este tehnica Backtracking, - în principiu o căutare în adâncime în graful soluţiilor. Unul dintre marile sale dezavantaje stă în faptul că depistarea unei inconsistenţe şi revenirea la variabilele asignate anterior pentru a le modifica se face "muncitoreşte", fără a lua în considerare care este variabila care a fost efectiv implicată în inconsistenţa respectivă. Metoda de backtracking controlat prin dependenţe ("dependency-directed backtracking"[StaSus77]) înlătură acest neajuns prin revenirea direct la sursa de conflict. Aceasta pastrează însă în continuare un mare neajuns: la momentul revenirii "se sare" peste soluţia construită de la punctul de eroare până la pasul curent, pierzându-se astfel o asignare parţială ce ar putea fi utilă până în final. Ginsberg [Gin96] a propus astfel o nouă îmbunătăţire – backtracking dinamic, ce structurează în mod dinamic căutarea: pentru fiecare valoare eliminată a unei variabile se reţine o listă de "nogoods" – ca în TMS-, adică variabilele ale căror valori curente sunt în conflict cu valoarea eliminată. Aceste liste se numesc "explicaţii eliminatorii", şi felul în care sunt efectiv folosite în algoritm determină salvarea de informaţii "nogood" numai pentru valorile curente ale variabilelor asignate;

cele care depind de asignări depăşite sunt şterse, îmbunătățind astfel semnificativ performanțele algoritmului:

Propoziție[Gin96]. Spațiul necesar pentru backjumping este $o(i^2v)$, unde i=|I|, este numărul variabilelor problemei şi v este numărul de valori pentru variabila cu cea mai mare mulțime de valori V_i.

Rezultatul pentru backjumping rămâne valabil şi pentru backtracking dinamic întrucât spațiul rămâne limitat de structura explicațiilor eliminatorii.

Vom prezenta algoritmul în detaliu în Secțiunea 6.3.5.

6.3.3.CSP pentru sisteme de argumentare

În[BesDou04] sunt prezentate trei metode de calcul a extensiilor admisibile. Ne-am oprit asupra celei bazate pe verificarea satisfiabilității, întrucât, după cum remarcă şi autorii, aceasta poate fi un punct de plecare în abordarea sistemelor de argumentare folosind tehnici de rezolvare a constrângerilor, iar noi dorim să adaptăm un algoritm DCSP la calculul acestor extensii.

Propoziție[BesDou04]**.** Fie (A,R) un cadru de argumentare. O mulțime $S \subseteq A$ este admisibilă ddacă formula:

$$\bigwedge_{a \in S} (a \wedge (\bigwedge_{b:(b,a)\in R} (\neg b \wedge (\bigvee_{c:(c,b)\in R} c)))) \wedge \bigwedge_{a \notin S} \neg a \; ; \; (6.5)$$

este satisfiabilă.

(Ultimul termen exprimă condiția de maximalitate, şi putem renunța la el dacă astfel ne apropiem mai bine de cerința problemei particulare pe care o avem în vedere).

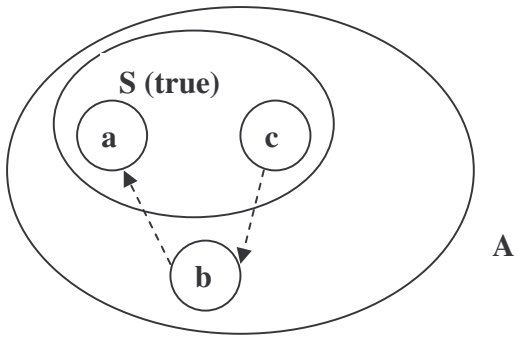


Figura 6.1. Reprezentarea grafică a relației 6.5. (săgețile simbolizează atacuri)

Dacă formula 6.5 este satisfiabilă, înseamnă că există o soluție a ei în cadrul sistemului. Algoritmii CSP pot calcula exact aceste soluții. Concret, dacă relațiile de atac ar fi

definite drept constrângeri asupra variabilelor implicate în atac, un algoritm CSP poate calcula extensiile posibile ale unui sistem de argumentare într-o manieră eficientă şi sistematică. O mulţime admisibilă de argumente se obţine prin filtrarea finală a soluţiilor cu relaţia 6.5, care reprezintă tot o constrângere, astfel că algoritmul CSP calculează în final exact mulţimile admisibile de argumente, ce vor fi interpretate ca soluţii posibile ale problemei de diagnoză: argumentele "in" fiind cele pentru care s-a asignat valoarea de adevăr *true*.

6.3.4.Algoritmi dinamici de satisfacere a constrângerilor

Se ştie că este esenţială capacitatea sistemelor de inteligenţă artificială de a raţiona pe baza unui mediu dinamic. Problemele clasice de satisfacere a constrângerilor nu sunt potrivite pentru probleme a căror structură se modifică dinamic (de exemplu, ca urmare a apariţiei unor cerinţe neprevăzute din partea utilizatorilor, sau prin apariţia de noi date ce trebuie încadrate în sistemul de constrângeri). Aceste situaţii pot fi tratate corect numai dacă se consideră că setul de constrângeri ce defineşte soluţia problemei se poate modifica dinamic (prin adăugare sau eliminare). S-a ajuns astfel la tehnica DCSP, cu variantele sale de reprezentare şi rezolvare [MigI01]. Ne vom opri pe scurt la varianta bazată pe activităţi, pe care o vom utiliza în cadrul aplicaţiei.

Definiţie[MigI01]. O problemă de Satisfacere a Constrângerilor Dinamică bazată pe Activităţi (aDCSP- activity-based Dynamic Constraint Satisfaction) constă dintr-o problemă CSP clasică şi un mecanism ce descrie cum sunt adăugate (şterse) variabilele din problemă, pe baza valorilor altor variabile (Context) (acest mecanism se poate reprezenta, la rândul său, tot prin constrângeri).

Astfel, o problemă aDCSP conţine două tipuri de constrângeri: constrângeri de *compatibilitate* (cele clasice), şi constrângeri de *activitate,* care specifică în ce condiţii o anumită variabilă şi domeniul său asociat devin relevante în construirea soluţiilor. În principal, constrângerile de activitate iau una din formele:

$$c_1 \in C_a : x_i = d_{i_a}, x_j = d_{j_b}, \ldots \rightarrow activ(x_k)\ ; \quad (6.6)$$

$$c_2 \in C_a : x_i = d_{i_a}, x_j = d_{j_b}, \ldots \rightarrow inactiv(x_k)\ ; \quad (6.7)$$

$$c_3 \in C_a : activ(x_i), activ(x_j), \ldots \rightarrow activ(x_k)\ ; \quad (6.8)$$

$$c_4 \in C_a : activ(x_i), activ(x_j), \ldots \rightarrow inactiv(x_k)\ . \quad (6.9)$$

Un caz particular de modificare a structurii problemei este situaţia când aceasta se schimbă nu ca urmare a unor intervenţii externe, ci chiar ca rezultat al deciziilor luate în cursul rezolvării[MigI01]. Astfel, în abordarea noastră, mecanismul dinamic ce modifică structura problemei este legat de testarea dinamică de simptome, testare ce reactualizează continuu contextul problemei.

6.3.5.Algoritmul Backtracking Dinamic pentru Probleme Dinamice de Satisfacere a Constrângerilor [VerSchi]

Este o variantă potrivită de abordare a problemelor DCSP bazate pe activități, pe care o vom implementa într-o formă adaptată în cadrul aplicației.

dbt(csp)
V=mulțimea de variabile a problemei CSP;
return **dbt-variabile(ϕ,V)**.

backward-checking(V,v)
-verifică valoarea curentă a variabilei v față de valorile variabilelor din V şi întoarce prima constrângere încălcată sau *succes* (dacă nu există o astfel de constrângere);

crează-explicație-eliminatorie(v,val,V)
-reține mulțimea de variabile V ca o explicație pentru eliminarea valorii 'val' din domeniul variabilei v;

şterge-explicații-eliminatorii(v,V)
-pentru toate variabilele din V şterge explicațiile eliminatorii care conțin variabila v şi întoarce valorile corespunzătoare din domeniile lor - trebuie reactualizat domeniul pentru $V_4 \cup V_2$;

dbt-valoare(V1,v,val)
v←val;
c=**backward-checking(V1,v)**;
if (c=*succes*) then return *succes*;
else { V_3←mulțimea de variabile din c;
deasignează(v);
crează-explicație-eliminatorie(v,val, V_3-{v});
return *failure*; }

dbt-variabilă(V_1, v,d)
if(d=ϕ) then return *failure*
else {fie *val*$\in d$ o valoare din domeniu;
if (**dbt-valoare(V_1 ,v,val)**=*succes*) then return succes
else return **dbt-variabilă(V_1 , v, d-{val})**};

dbt-variabile (V_1,V_2)
//V_1 este mulțimea variabilelor asignate, V_2 – a celor neasignate;
if(V_2=ϕ) then return *succes*;
else {v← o variabilă din V_2;
d= domeniul curent al variabilei v;

if (**dbt-variabilă(V1,v,d)**=*failure*) then return **dbt-bt-variabilă(V_1,V_2,v)**;
else return **dbt-variabile($V_1 \cup$ {v}, V_2-{v})**;}

dbt-bt-variabilă (V_1, V_2, v)
$V_3 \leftarrow$ mulțimea de conflicte a lui v (= uniunea explicațiilor eliminatorii ale valorilor eliminate ale lui v);
if ($V_3 = \phi$) then return *failure*
else { v'←ultima variabilă $\in V_3$ din V_1;
val'←valoarea curentă a lui v';
$V_4 \leftarrow$mulțimea variabilelor de după v' din V_1;
deasignează(v');
crează-explicație-eliminatorie(v',val', V_3-{v'});
şterge-explicație-eliminatorii(v', $V_4 \cup V_2$);
dbt-variabile (V_1-{v'}, $V_2 \cup$ {v'})}

şterge-explicațiile-eliminatorii-de-variabile(V_3,V)
-pentru fiecare variabilă din V, şterge explicațiile eliminatorii care conțin variabile din V_3 (deasignate);

şterge-explicațiile-eliminatorii-de-constrângeri(RC,V)
-pentru fiecare variabilă din V, şterge explicațiile eliminatorii care conțin variabile care intră în componența unor constrângeri înlăturate;

şterge-asignări(V_1,AC)
fie c$\in$ AC; V(c)=variabilele din constrângerea c;
-dacă asignările variabilelor din V_1 invalidează c, deasignează ultima variabilă a lui V(c) din V_1 şi crează explicațiile eliminatorii corespunzătoare;
-întoarce mulțimea V_3 a variabilelor care au fost deasignate;

ddbt(V_1, V_2, AC, RC)
//V_1=mulțimea ordonată a variabilelor asignate la căutarea anterioară;
//V_2=mulțimea variabilelor neasignate la căutarea anterioară;
//AC= mulțimea constrângerilor adăugate (Added Constraints);
//RC= mulțimea constrângerilor şterse (Removed Constraints);

$V = V_1 \cup V_2$;
$V_3 \leftarrow$**şterge-asignări(V_1, AC);**
şterge-explicațiile-eliminatorii-de-variabile(V_3,V);
şterge-explicațiile-eliminatorii-de-constrângeri(RC,V).
return **dbt-variabile(V_1-V_3, $V_2 \cup V_3$).**

6.4.Concluzii şi contribuţii

Capitolul curent a detaliat bazele teoretice ale abordării originale din această lucrare. Am prezentat sistemele de argumentare directă ca formalism modern de reprezentare, adecvat abordării abducţiei, şi pe care îl vom folosi la rândul nostru în cadrul sistemului DiaMed. Introducerea sistemelor de argumentare pentru modelarea nonmonotonicităţii ne este necesară întrucât modelul medical va conţine şi relaţii asimetrice de atac -ilustrate de situaţia concretă a bolilor mascate, descrisă în Capitolul 2 (o relaţie de atac exclusiv simetrică se poate reduce la o problemă de consistenţă). Am introdus semantica specific argumentativă a admisibilităţii întrucât o vom utiliza atât în definirea originală a unui diagnostic multiplu, cât şi pentru a modela menţinerea consistenţei pe parcursul raţionamentului. Paragraful cheie al capitolului este cel din subsecţiunea 6.2.3, în care se vede cum se reduce calculul semanticii admisibile la o problemă de satisfiabilitate, pentru a cărei implementare algoritmii specifici problemelor de satisfacere a constrângerilor (CSP) sunt o opţiune naturală. Între versiunile posibile ale acestor algoritmi, ne-am oprit asupra backtracking-ului dinamic pentru DCSP [VerSchi] pentru eficienţa şi adecvarea sa la specificul unei probleme de diagnoză medicală.

VII.Relaţia abducţiei cu sistemele de argumentare, tehnicile CSP şi sistemele TMS

7.1.Introducere

Secţiunea prezentă îşi propune să fie o dezvoltare mai aprofundată a unor idei din secţiunile precedente. Pentru început, vom încerca să descifrăm legătura programării logice (domeniu bine conturat şi studiat, în special datorită caracterului său procedural, orientat spre calcul, cu aplicabilitate imediată), cu abducţia. Aceasta ne permite, pe de o parte, să arătăm cum s-a ajuns de la semantica stabilă la cea admisibilă în calculul abducţiei (semantica admisibilă, specifică argumentaţiei, a rezolvat anumite probleme când a fost folosită în programarea logică), şi, pe de altă parte, cum o procedură clasică de abordare a abducţiei în programarea logică îşi găseşte natural translatarea în sistemele de argumentare. Concluzia demersului ar fi că sistemele de argumentare au cel puţin aceeaşi putere de reprezentare şi sunt chiar mai adecvate sub anumite aspecte decât programarea logică, cel puţin când aplicaţia este abducţia.

Ne-am propus, deasemenea, să dezvăluim relaţia abducţiei cu sistemele TMS- o alternativă de manevrare nemonotonă a dependenţelor logice- şi cu problemele de satisfacere a constrângerilor (CSP). Intenţia ultimelor subcapitole este de a arăta avantajele sistemelor de argumentare directă şi ale problemelor CSP asupra sistemelor TMS, din punctul de vedere al aplicaţiei noastre (diagnoza medicală) .

7.2.Abducţia în Programarea Logică

În această secţiune vom prezenta cum şi de ce s-a ajuns de la semantica stabilă la semantica admisibilă pentru raţionamentul abductiv. Atât utilizarea semanticii stabile pentru negaţia ca imposiblitate de a demonstra (naf) în programele logice normale cât şi a semanticii stabile generalizate pentru ALP determină ca procedurile de calcul abductive utilizate să nu fie valide decât pentru un tip special de programe logice.

Ideea principală a secţiunii este aşadar că semantica admisibilă (pe care şi noi o utilizăm în cadrul aplicaţiei) este validă pentru procedurile de calcul abductiv în general, fiind prin urmare o abordare adecvată a diagnozei (care este exemplul cel mai întâlnit de abducţie).

Orice program logic P poate fi privit drept cadru abductiv (vezi şi Secţiunea 3.4.1) prin transformarea [EshKo88, EshKo89]:

$$P \rightarrow <P^*, A^*, I^*>, \quad (7.1)$$

unde: P*=teorie, A*= predicate abductibile, I*= constrângeri de integritate.

1. A*={p*} (p* introdus pentru pentru fiecare predicat p din P);
2. P*=P, unde fiecare literal negativ ~p(t) a fost înlocuit cu p*;
3. I*= constrângeri de integritate de forma:

$$\forall x \neg [p(x) \wedge p^*(x)] \quad (7.2)$$

$$\forall x [p(x) \vee p^*(x)] \quad (7.3)$$

(Constrângerea 7.3 forțează p* când p nu e susținut adevărat).

Semantica asociată cadrului abductiv <P*, A*, I*> în termeni de extensii $P^* \cup \Delta$, unde $\Delta \subseteq A^*$, dă o semantică pentru programul inițial P, şi anume:

> *P→Q (concluzia Q e valabilă raportat la P) ddacă interogarea Q* are o explicație abductivă în <P*, A*, I*> (Q* este Q în care ~p(t) se înlocuieşte cu p*(t).*

Definiție. Abducția în logică [KKoT98]. Fie G=scop (goal/query: ceea ce necesită explicație), Δ= *explicație (extensie) abductivă* pentru G, T= teoria domeniului. Atunci:
1. $T \cup \Delta \models G$;
2. $T \cup \Delta$ consistentă.

Semnificația I:* $P^* \cup \Delta$ extensie a P* (o teorie Horn) satisface I* dacă orice p atom fără variabile satisface:

1. $P^* \cup \Delta \not\models p \wedge p^*$ (7.4)
2. $P^* \cup \Delta \models p$ sau $(P^* \cup \Delta \models p^*)$. (7.5)

Constrângerile de integritate I* forțează menținerea consistenței în cadrul procesului de generare de extensii.

Rezultat[GelLi88]. Transformarea de mai sus determină o corespondență unu la unu între modelele stabile ale programului logic P şi extensiile abductive (program + explicația abductivă pentru o anumită interogare Q) ale lui P*.

Definiție. Model stabil al unui program logic general P [KKoT98]. Presupunem că toate clauzele lui P sunt fără variabile. Pentru orice mulțime M de atomi fără variabile, fie P_M programul Horn obținut prin ştergerea din P a:
1. fiecărei reguli ce conține un literal negativ ~A, cu $A \in M$;
2. tuturor literalilor negativi din regulile rămase.

Dacă modelul (Herbrand) minimal al lui P_M coincide cu M, atunci M este un *model stabil* al lui P.■

În [EshKo89], autorii au propus o *procedură de calcul abductivă pentru programarea logică*- o metodă efectivă de calcul a extensiilor abductive (şi deci a modelelor stabile) ce îmbină două faze:

1.faza abductivă (generează ipoteze);

2.faza de testare a consistentei (verifică incremental dacă ipotezele verifică constrângerile de integritate ale programului)[2].

În această procedură, abducția este folosită pentru tratarea NAF ("Negation As Failure": negația ca imposibilitate de a demonstra): când se ajunge la un literal ce nu poate fi susținut din premize (date) şi program, acesta este considerat ipoteză (presupus adevărat).

Această metodă de calcul abductivă extinde SLDNF (Single Linear Derivation with Negation as Failure) şi e validă pentru programele logice consistente la ordine, însă nu e validă în general. Dung a argumentat ca semantica stabilă poartă de fapt răspunderea pentru această invaliditate. El a propus în schimb o altă semantică abductivă (semantica *preferată*) pentru NAF, fără constrângerile de integritate disjunctive [Dung91]. Pentru semantica preferată propusă de el procedura de calcul abductivă s-a dovedit validă în general.

Rezolvarea propusă de Dung (şi ideea de la baza semanticii preferate) a constat în înlocuirea constrângerii de integritate disjunctivă cu cerința ca Δ (mulțimea de ipoteze negative =explicația abductivă) să fie maximală, depăşind însă deficiențele noțiunii de maximalitate[3] de până la el. Astfel, Dung a introdus o noțiune mai subtilă de maximalitate. Programului P îi asociază cadrul abductiv <P*,A*, I*>, unde din I* au dispărut constrângerile conjuctive:

$$\text{I*: } \forall x, \neg[p(x) \wedge p^*(x)]; \quad (7.6)$$

Definiție[KKoT98]. Fie Δ, E= mulțimi de ipoteze (negative), $\Delta, E \subseteq A^*$. Spunem că E *atacă* Δ (relativ la <P*,A*, I*>) dacă:

1. $P^* \cup E \vdash p$,
2. $p^* \in \Delta$.

Definiție[KKoT98]. O extensie $P^* \cup \Delta$ a lui P* e *preferată* dacă:

1. $P^* \cup \Delta$ e consistent cu I* şi
2. Δ e maximală cu proprietatea că pentru orice atac E împotriva lui Δ, Δ atacă E (Δ se apără față de toate atacurile).

Semantica preferată rezolvă inclusiv exemplele de genul EX.

O extensie *admisibilă* îndeplineşte aceleaşi condiții ca una preferată, cu deosebirea că nu e maximală.

[2] cele două faze se regăsesc, ca idee generală, şi în abordarea originală din DiaMed.

[3] Spre exemplu, dacă avem p←~q, doar {q*} e intuitiv corectă din cele două extensii maximale {p*} şi {q*}, deoarece {p*} e atacată de {q*} fără să se apere (EX).

Semantica admisibilă este mai potrivită pentru procedura de calcul abductivă, întrucât, după cum am menționat deja, pentru semantica admisibilă procedura e validă. Acest avantaj a dus la extinderea cadrului de lucru pentru ALP (Abductive Logic Programming) [EshKo88], ELP (Extended Logic Programming), prin definirea corespunzătoare a noi noțiuni de atac.

Programarea Logică Abductivă (ALP)
ALP este o extensie a programării logice pentru a trata abducția în general, nu doar abducția pentru NAF (în sensul că există o clasă specială de predicate abductibile).

Presupunând că predicatele abductibile nu sunt definite în cadrul teoriei T (nu apar în capul clauzelor), se garantează că toate explicațiile vor fi de bază (acesta este un neajuns de reprezentare pentru problema noastră, şi un argument în plus în favoarea abordării noastre) (la noi asumpțiile/ipotezele nu sunt independente, astfel că procedura de calcul abductivă e prea restrictivă; abordarea noastră, deşi aproximativă (calculează cu aproximație semantica admisibilă), înlătură acest neajuns).

Pentru ALP, Kakas & Mancarella au propus semantica stabilă generalizată [KaMan90], ce extinde semantica modelului stabil. Modelele stabile generalizate combină folosirea abducției pentru raționament implicit (de forma NAF) cu utilizarea pentru alte forme de raționament ipotetic.

Fie <P,A,I> un cadru abductiv; P un program logic general (clauze Horn +NAF).

Definiție[KKoT98].Fie $\Delta \subseteq A$; $M(\Delta)$ este un *model stabil generalizat* al <P,A,I> ddacă:

1.$M(\Delta)$ e model stabil al $P \cup \Delta$;
2.$M(\Delta) \models I$.

Definiție[KKoT98].O *explicație abductivă* a interogării Q în <P,A,I> este orice subset Δ al lui A astfel încât:

1. $M(\Delta)$ este model stabil generalizat al <P,A,I>, şi
2.$M(\Delta) \models Q$.

Procedura de calcul abductivă pentru ALP: când se adaugă o ipoteză nouă la extensia Δ, procedura rezolvă ipoteza față de toate constrângerile de integritate ce o conțin, şi apoi raționează înapoi de la rezolventă. Din nou, procedura nu este validă pentru semantica stabilă generalizată. O trăsătură importantă a procedurii de calcul abductive stă în evitarea unei verificări a integrității generală (nu foloseşte regulile din program pentru raționament forward; foloseşte doar constrângerile $\neg(q \wedge q^*)$ pentru a raționa backward de la rezolventă; în general, raționamentul forward nu este de dorit întrucât nu focalizează căutarea, nefiind orientat spre scop).

Ceea ce lipseşte abordărilor bazate pe elementele prezentate succint mai sus este o procedură de calcul constructivă şi eficientă (adică tocmai contribuția noastră la

problemă). În plus, procedura abductivă nu este completă: după ce găseşte o explicaţie se opreşte, ceea ce deasemenea este un dezavantaj din punctul nostru de vedere, întrucât suntem interesaţi să comparăm între ele toate explicaţiile posibile.

Programarea logică extinsă (ELP)
Foloseşte o formă explicită de negaţie [KKoT98], atunci când definiţia unui predicat e incompletă (pentru a defini instanţele negative ale predicatului în loc să le inferăm implicit folosind NAF). NAF pentru informaţia negativă e sigură numai sub ipoteza **cwa**("**c**losed **w**orld **a**ssumption" - lume închisă); - nu e potrivită pentru predicatele abductibile (care pot fi incomplete, şi deci "deschise").

7.3.Sistemele de argumentare directă şi abducţia

În Secţiunea 6.1. am făcut o introducere a sistemelor de argumentare directă. În această secţiune, vom sublinia, o dată în plus, avantajele admisibilităţii asupra "stabilităţii", privite însă dintr-o perspectivă nouă, şi anume din cea a sistemelor de argumentare.

Am vazut că invaliditatea procedurilor prezentate mai sus este depăşită prin introducerea unei semantici (admisibilă) care, de fapt, este specifică teoriei argumentaţiei directe. De aici rezultă şi avantajul major al sistemelor de argumentaţie asupra programării logice în abordarea abducţiei.

Interpretarea argumentativă a procedurii de calcul abductive [KKoT98]
Cele două faze principale ale procedurii de calcul abductive din secţiunea alocată programării logice pot fi privite prin prisma argumentaţiei astfel: pasul I construieşte o mulţime de ipoteze pentru rezolvarea scopului iniţial; iar pasul II augmentează mulţimea construită la I cu toate apărările necesare. După cum s-a menţionat, procedura de calcul abductivă e validă pentru semantica admisibilă.

Problema validităţii pentru ALP poate fi deasemenea tratată prin introducerea unei semantici specifice argumentaţiei, care să trateze constrângerile de integritate şi NAF uniform via o noţiune corespunzător adaptată de "atac".

O interpretare argumentativ teoretică a procedurii de calcul abductiv pentru ALP
Definiţie[KKoT98].

- *Atacuri via NAF*: E *atacă* Δ *via NAF* relativ la $<P^*, A \cup A^*, I \cup I^*>$, dacă
  $P^* \cup E \vdash p$, pentru $p^* \in \Delta$, sau $a^* \in E$, pentru *a* abductibil $\in \Delta$;

- *Atacuri via constrângeri de integritate*: E *atacă* Δ via constrângerea de integritate $\neg (L1 \wedge \ldots \wedge Ln) \in I$ relativ la $<P^*, A \cup A^*, I \cup I^*>$, dacă
  $P^* \cup E \vdash L_1, \ldots, L_{i-1}, L_{i+1}, \ldots, L_n$, pentru $L_i \in \Delta$.

Semantica argumentativ-teoretică pentru ALP
Definiţie[KKoT98]. O mulţime de ipoteze Δ este *KM-admisibilă* dacă:

Pentru orice atac E împotriva lui Δ, Δ atacă (E-Δ) doar via NAF (doar atacurile via NAF pot să contra-atace).

Semantica argumentativ teoretică pentru ELP [KKoT98]
ELP se pot transforma în ALP astfel: se redenumeşte negaţia explicită a unui predicat p al programului: $\neg$p cu p' şi se adaugă la constrângerile de integritate următoarele constrângeri:

$$\forall x, \neg[p(x) \wedge p'(x)]. \quad (7.7)$$

Definiţie [KaManDun94].
E *atacă* o mulţime nevidă Δ *via negaţie explicită* relativ la un program P' dacă:

- P' $\cup$ E $\vdash$ l
- P' $\cup$ Δ $\vdash$l', pentru un literal l.

(adică evidenţa pozitivă are prioritate faţă de evidenţa negativă, o evidenţă pozitivă atacă o evidenţă negativă).

Definiţie[KKoT98]. O mulţime de ipoteze Δ este *D-admisibilă* dacă:

1) D nu se atacă pe sine, via NAF sau via negaţie explicită;
2) Pentru orice atac E împotriva lui Δ, via negaţie explicită sau NAF, Δ atacă E via NAF.

7.4. Calculul abducţiei folosind TMS. Legătura între CSP şi abducţie

Prezentăm în cele ce urmează o abordare procedurală diferită a abducţiei. Este vorba de sistemele TMS (Truth Maintenance System-sistem de menţinere a adevărului), sisteme cu scopul general de a manevra dependenţele logice şi logica nemonotonă. Motivaţia principală a acestor sisteme stă în necesitatea de a trata procedural menţinerea adevărului şi revizuirea încrederilor. După cum definea McAllester [McAll80], un TMS este un "sistem independent de domeniu pentru menţinerea consistenţei şi bunei –fundamentări (well-foundedness) a unei mulţimi de încrederi".

După cum am văzut la procedura de calcul abductivă, explicaţia abductivă D era reţinută şi putea folosi în revizuirea ulterioară a încrederilor, exact ca justificările din sistemele TMS. Acestea din urmă dispun însă de un mecanism mai bine închegat de respingere pe baza justificărilor memorate.

În sistemele TMS predomină, aşadar, aspectul procedural şi sunt alcătuite, în general, din două module separate ce interacţionează pe parcursul calculului:

1)PS(problem solver): un rezolvitor de probleme dependent de domeniu care efectuează inferenţe (utilizând logica de ordinul întâi);

2)sistemul TMS propriu-zis: un sistem independent de domeniu care memorează inferenţele (de obicei utilizează inferenţă propoziţională). Acest modul are scopul de a determina ce propoziţii se pot deriva pe baza justificărilor, fără încălcarea constrângerilor de integritate de tip "nogood"("nogoods" =mulţimi inconsistente de ipoteze).

Paralela între sistemele TMS şi ALP (Programarea Logica Abductiva) este foarte strânsă: Satoh şi Iwayama au demonstrat în [SatIwa91] ca sistemul JTMS al lui Doyle [Doy79] calculează, de fapt, semantica stabilă generalizată (ce stă şi la baza ALP; faptul că sistemul este legat de semantica stabilă reprezintă însă un dezavantaj al său din punctul de vedere al diagnozei, după cum am văzut). În plus, se poate identifica imediat o echivalenţă între termenii pe care cele două abordări îi folosesc:

Abductive Logic Programming –TMS:

- Clauza propoziţională Horn= Justificare
- Constrângere de integritate=Nogood; (o constrângere de integritate not(L1 şi L2) = nogood(L1,L2));
- Ipoteză abductibilă=Asumpţie;
- Mulţime acceptabilă de ipoteze=Context.

Un dezavantaj al sistemelor TMS faţă de ALP rezultă din avantajul constrângerilor de integritate "clasice" faţă de cele de tip "nogood": există multe mulţimi de ipoteze ce se pot ataca via aceeaşi constrângere de integritate, mulţimi ce în TMS ar fi fost reţinute explicit; în schimb, o constrângere "clasică" este reprezentanta unei mulţimi largi de posibile atacuri (care nu e nevoie sa fie reţinute explicit) -atacuri între tot atâtea mulţimi posibile ce ar conţine variabilele din constrângere.

Există însă şi avantaje ale TMS faţă de programarea logică. Unul dintre acestea vine din ideea de a folosi **structuri de dependenţă** în raţionament şi rezolvare de probleme, structuri de dependenţă utilizate pentru a face sistemele de raţionament mai incrementale (adică pentru a păstra maximum din soluţia construită până la un anumit moment) în timpul retragerii (retraction) şi backtracking-ului, şi pentru a facilita mai bine explicaţiile (care erau mai rudimentare în programarea logică). În plus, prin exploatarea acestor dependenţe s-a evitat backtracking-ul inutil, redescoperirea contradicţiilor sau a inferenţelor (memorate în modulul TMS). Aceste structuri de dependenţă sunt descoperite (construite) dinamic pe parcursul calculului de către modulul de rezolvare (PS), folosind logica de ordinul întâi, şi apoi memorate.

În plus, după cum s-a mai menţionat, TMS este un modul folosit de sistemele deductive pentru a menţine relaţiile logice între încrederile manipulate [McAll80]; aceste relaţii sunt baza modificării incrementale a structurii de încrederi când se modifică premizele, oferind un mecanism de "context" mai flexibil decât în programarea logică.

Relaţiile memorate între încrederi pot fi deasemenea folosite pentru a găsi direct sursa contradicţiilor, îmbunătăţind eficienţa backtracking-ului (DDB:Dependency Directed Backtracking- backtracking controlat prin dependenţe).

Doyle a identificat 4 trăsături fundamentale ce caracterizează un sistem TMS în general:

1. Efectuează deducţie propoziţională pornind de la un set de premize;
2. Menţine justificări pentru a explica rezultatele deducţiilor sale;
3. Îşi actualizează incremental încrederile când se adaugă/şterg premize;

4. Realizează backtracking controlat prin dependențe (=când apar contradicții folosește justificările înregistrate pentru a găsi premizele ce stau la baza contradicției.)

Prezentăm informal, spre exemplificare, sistemul TMS propus de McAllester [McAll78].

TMS-McAllester

În acest sistem, propozițiile pot fi adevărate, false sau cu valoare necunoscută de adevăr (în loc de *adevărat*, *fals*, *in*, *out* ca la TMS-Doyle sau ATMS- de Kleer [deKleer86]). Adevărul unei propoziții este o asumpție (ipoteză), care se reprezintă prin marcarea corespunzătoare a propoziției; când se descoperă că ipoteza a generat o contradicție, valoarea de adevăr e retrasă automat.

1.Propagarea propozițională a constrângerilor
Cele mai vechi şi răspândite tehnici de propagare propozițională a constrângerilor pleacă de la algoritmul original de rezolvare a problemei satisfiabilității booleene(SAT) Davis-Putnam, bazat pe rezoluție. Au urmat apoi DPLL (o transformare a algoritmului original, bazată pe căutare). O dezvoltare aparte au avut-o algoritmii BCP (Boolean Constraint Propagation), ce realizează propagare unitară ("unit propagation") pentru a îngusta spațiul de cautare prin cautare înainte ("look-ahead").

Orice sistem de propagare a constrângerilor are un set de celule ce pot lua valorile true-false, şi o mulțime de constrângeri ce restrâng combinațiile posibile ale acestor valori. Sistemul TMS [McAll78] realizează deducție propozițională pe baza valorilor anterior determinate (premize) şi o *singură* constrângere, aleasă din mulțimea totală a constrângerilor. Acest mod de lucru are dezavantajul *incompletitudinii*: nu toate deducțiile ce urmează logic dintr-un set de constrângeri sunt efectuate.

În TMS, toate constrângerile (definite ca relații logice) iau forma unor clauze disjunctive de forma $(\vee$(p.false)(q.true)), şi sunt derivate din axiomele de bază ale logicii propoziționale. Premizele propoziționale reprezintă asignări de valori de adevăr ale aserțiunilor. Cum orice aserțiune se defineşte folosind simbolurile logice (or, and, $\rightarrow$, not), toate constrângerile clauzale relevante se generează pe baza axiomelor legate de simbolurile logice prezente în aserțiunile respective. Avantajul acestei reprezentări este că dacă se adaugă incremental premize şi constrângeri la TMS, se poate face deducție pe baza lor fără nici un efort suplimentar.

2.Justificări (Suport)
Orice deducție se face pe baza unei singure clauze + valorile de adevăr ale aserțiunilor acelei clauze; deci o justificare pentru valoarea de adevăr dedusă se poate reprezenta ca un pointer către clauza folosită pentru a deduce valoarea respectivă (prin căutare recursivă se construiesc justificări extinse =structuri suport ale ipotezelor.) Este necesar ca aceste structuri suport să fie bine fundamentate (să nu avem dependențe circulare).

Dacă s-a asertat o valoare de adevăr (premiză) contradictorie cu o valoare de adevăr dedusă deja de sistem, trebuie să facem *retracţie*.

3. Retracţie

O operaţie fundamentală în menţinerea adevărului este reactualizarea incrementală a bazei de aserţiuni când unele premize sunt retrase. Toate valorile de adevăr ce depind de premiza ştearsă (=consecinţele sale) se şterg; (se verifică pentru clauzele ce conţin orice aserţiune cu valoarea de adevăr retrasă ce valori susţin, aceste valori sunt retrase recursiv). Pentru aserţiunile cu valoarea de adevăr retrasă se verifică dacă putem deduce o valoare pentru ele în alt mod.

4.Backtracking şi respingere

O *contradicţie* este o clauză a TMS cu toţi termenii falşi (ex: ($\vee$(p.false)(q.true)) e contradicţie dacă p=true şi q=false). Pentru ca mecanismul de backtracking să nu interfereze cu deducţia şi retracţia, procesarea contradicţiilor se face în afara acestor procese.

Când apare o contradicţie, sistemul întreabă care din premizele ce stau la baza contradicţiei trebuie retrase (dependency directed backtracking). Când o premiză de la baza unei contradicţii e retrasă e important să deducem negaţia ei pentru a preveni reapariţia ulterioară a contradicţiei (Local Clause Resolution): se adaugă o clauză ce conţine negaţiile tuturor premizelor de la baza contradicţiei; deci se va deduce negaţia uneia din premize când toate celelate sunt crezute (operaţia are la bază o procedură sofisticată de creare de constrângeri locale ce au rolul de a da justificări mai scurte şi mai structurate).

Aserţiuni de verosimilitate (Likelihood assertions)

Un controler al premizelor ar trebui să poată să folosească deducţiile efectuate de TMS pentru a face mai multe tipuri pozitive de control al premizelor: (very-likely p) =aserţiune de verosimilitate. O problemă deschisă este cum poate un sistem "înţelege" şi utiliza aceste aserţiuni. Aici sistemul de argumentare în forma folosită de aplicaţia noastră are un avantaj din faptul că memorează explicit dependenţe utile în partea de raţionament, şi aceste dependenţe sunt selectate dinamic cu ajutorul unor funcţii de decizie fuzzy, ce înlocuiesc practic partea rezolvitorului de probleme (PS), evitând astfel dezavantajele logicii de ordinul întâi. În plus, dependenţele memorate explicit permit localizarea raţionamentului (şi deci restrângerea spaţiului de căutare) încă de la început, fiind în acelaşi timp un suport pentru proprietatea de localitate a admisibilităţii. Pe scurt, un avantaj al argumentelor faţă de sistemele TMS stă în faptul că primele reţin structuri ce sintetizează o parte din inferenţele rezolvitorului.

Controlerul de premize poate efectua multe reveniri inutile (în backtracking). Dacă numărul de ipoteze e mare şi interacţiunile dintre ele sunt complexe, acest fapt se transformă într-un mare dezavantaj al sistemului TMS prezentat.

ATMS

Sistemul ATMS reprezintă o facilitate foarte generală pentru toate tipurile de raționament implicit; avantajul său principal față de alte sisteme TMS stă în faptul că toate soluțiile posibile (de obicei mutual inconsistente) sau soluțiile parțiale sunt direct disponibile PS (problem solver); (efectuează căutare în lărgime), deci spre deosebire de TMS-ul lui McAllester va lucra fără retracție, backtracking, context-switching. ATMS oferă deci posibilitatea lucrului cu ipoteze contradictorii simultan, uşurând astfel diagnosticul diferențial. ATMS înglobează şi o formă de NAF: ipotezele sunt asumate în lipsa evidenței pentru contrarii.

Notații şi noțiuni fundamentale ale sistemului ATMS:

- *Asumpție*= ipoteză de lucru (dată primitivă pe care se bazează derivarea tuturor celorlalte date);
- ATMS: asumpțiile sunt crezute în lipsa evidenței pentru contrarii;
- *Mediu* (environment)= mulțime de ipoteze; un mediu inconsistent se numeşte *nogood*.
- *Justificare*=x_1 and…and $x_n \rightarrow$n; construite de PS, descriu cum depind datele unele de altele într-un singur pas; (regulile nu se reapelează pe aceleaşi date, odată apelate sunt reținute de TMS sub forma unei justificări);
- *Eticheta* unui nod (construită de TMS): descrie cum depind datele unele de altele în mai mulți paşi; (formată din acele medii în care nodul este susținut, adică are statutul "in");
- *Context*=mulțimea formată de ipotezele unui mediu consistent + toate nodurile derivabile din aceste ipoteze;

TMS trebuie să răspundă la întrebări de genul: dacă un context e inconsistent; dacă un anumit nod este valabil într-un context dat. Algoritmii TMS reactualizează consistent la fiecare pas etichetele nodurilor astfel încât să răspundă eficient la întrebări de acest gen.

Complexitatea PS din ATMS depinde de: numărul de reguli executate(1) şi de numărul de contexte luate în considerare pe parcursul căutării (2). Eficiența ATMS provine din faptul că acesta organizează căutarea astfel încât să găsească întâi cele mai generale inferențe; în acest scop, consumatorii din mediile mai mici sunt executați primii, şi dintre ei, primii se apelează cei pentru detectarea inconsistențelor. Programatorul (scheduler) de consumatori alege în mod repetat cel mai mic mediu ce are consumatori neexecutați şi rulează unul din aceşti consumatori până ce toți consumatorii tuturor mediilor consistente sunt rulați. Dacă ATMS găseşte un mediu inconsistent se opreşte (nu mai explorează mediul respectiv, şi nici vreun superset al său), avantajul fiind că nu explorează medii inconsistente neminimale. Combinația regulilor rezolutive din ATMS cu backtracking produce "cea mai bună schemă de DDB". Principalul dezavantaj este dificultatea de depanare.

Comparație CSP-ATMS

Constrângerile sunt o metodă naturală de reprezentare a cunoştințelor în foarte multe domenii, după cum observa autorul în [MigI01]. O problemă bazată pe constrângeri conține trei componente principale:

Definiție[Gin96]. O problemă de satisfacere a constrângerilor (V,D,c) reprezintă o mulțime V de variabile; pentru fiecare v∈ V, există o mulțime posibilă de valori D_v ale acestei variabile. c este o mulțime de constrângeri, fiecare fiind o pereche (W, K), unde W=(v_1,…v_k) este o submulțime ordonată a lui V, şi K este o submulțime a lui $D_{v1}\times \ldots \times D_{vk}$.

O problemă comună CSP şi ATMS [deKleer89] este cum poate fi controlată căutarea pentru a evita subspațiile determinate anterior ca inconsistente ("thrashing"). În CSP se utilizează backtracking şi preprocesarea constrângerilor-pentru a avea mai puține reveniri din backtracking şi a determina în prealabil stările (combinațiile de valori) inconsistente; însă preprocesarea este costisitoare în sine. ATMS este folosit întotdeauna interactiv cu un motor de inferențe (care efectuează backtracking), ATMS fiind modulul care forțează consistența locală de genul celei din CSP, având aceeaşi motivație cu algoritmii de consistență locală de acolo. În concluzie, atât CSP cât şi ATMS folosesc intuiții similare pentru a evita revenirea în regiuni determinate deja ca inconsistente; şi au aceleaşi compromisuri computaționale. Cheia acestui paralelism stă în faptul că atât algoritmii de consistență a nodurilor/arcelor/căilor din CSP cât şi algoritmii ATMS sunt acoperiți de câteva reguli de rezoluție propozițională, prezentate în continuare.

Codificarea propozițională a CSP

Fie:

- variabila x_i, cu domeniul D_i;
- $\{a_{ij}\}$ = mulțimea valorilor posibile pentru variabila x_i;
- C=mulțimi de clauze.

Se poate codifica unic rețeaua de constrângeri CN (**C**onstraint **N**etwork) în mulțimea de clauze D(CN) astfel:

- Orice variabilă x_i trebuie să ia o valoare din domeniul său $D_i=\{a_{ij}\}_{j\in I}$:

$$(x_i : a_{i1}) \vee \ldots \vee (x_i : a_{in_i}); \quad (7.8)$$

- Orice variabilă poate lua cel mult o valoare la un moment dat:

$$\forall (a_{ij} \neq a_{ik}) : \neg x_i : a_{ij} \vee \neg x_i : a_{ik} \quad (7.9)$$

- Fie o constrângere C_Z, fie $[t_1,\ldots,t_m]$ un tuplu de asignări variabilelor din Z care falsifică C_Z; exprimăm faptul că nu trebuie să existe combinații de asignări care să încalce vreo constrângere prin clauza:

$$\neg(t_1 \wedge \ldots \wedge t_m)\,. \quad (7.10)$$

Reguli de inferență propoziționale

Toate tehnicile de consistență locală din ATMS şi CSP pot fi privite drept implementarea câtorva reguli de rezoluție de bază. Acestea sunt:

- H_0: şterge clauzele subsumate din C (întrucât sunt redundante);
- H_3: Regula rezoluției unitare:

$$\frac{\neg A \qquad A \vee A_1 \vee \ldots \vee A_m}{A_1 \vee \ldots \vee A_m}$$

- H_5 (generalizare H_3):

Fie $N(\beta)$= clauză negativă cu simbolul β şi $N(\alpha_i)$ o mulțime de clauze negative. Atunci:

$$\frac{A_1 \vee \ldots \vee A_m \qquad N(\alpha_i), A_i \in \alpha_i, A_{j\neq i} \notin \alpha_i, \forall i}{N(\bigcup_i [\alpha_i - \{A_i\}])\,.}$$

În ceea ce priveşte diferențele, ATMS sunt incrementali, în timp ce consistența locală în CSP nu e incrementală (decât în versiunea DCSP); în plus, ATMS sunt mai generali (pot rezolva direct orice task exprimat propozițional) astfel că se poate trece de la o problemă CSP la una modelată prin ATMS întotdeauna, dar invers nu. ATMS este mai general ca CSP şi pentru că, în plus, construieşte explicații, face diagnostic diferențial, îmbină formularea şi rezolvarea (în abordarea CSP, doar versiunea DCSP se permite reconstrucția dinamică a problemei de constrângeri). Mai mult, ATMS poate îmbina rezolvarea unei CSP cu alte tipuri de raționament propozițional: calculul consistenței locale poate colabora uşor cu backtracking şi se poate utiliza backtracking ca o tehnică de control [deKleer86b] pentru a limita evaluări nenecesare ale predicatelor constrângerilor.

Dacă algoritmii de consistență locală în CSP aveau rolul de a reduce numărul de reveniri în backtracking, consistența locală pentru ATMS joacă un rol mai central: este fundamental pentru ATMS să răspundă eficient la interogări despre consistența unei mulțimi de simboluri. Iar dacă în CSP algoritmii de consistență şterg incremental violări ale constrângerilor prin îngustarea minimală a constrângerilor, în ATMS restrângerea spațiului de căutare se face prin construcția de constrângeri de integritate de tip *"nogood"*

deduse din C (N este mulțimea de constrângeri "nogood"; D este domeniul unei variabile, format din alegerile posibile, fiecare simbol din D fiind o ipoteză).

După cum se va vedea în secțiunea următoare, asemănările menționate (tratate aici la suprafață) au ca suport riguros faptul că atât ATMS cât şi CSP pot fi privite drept cazuri particulare ale Sistemului de Management a Clauzelor (CMS) propus de către de Kleer.

Calculul abducției prin TMS

Menționam mai sus că punctul de legătură cel mai important între TMS şi programarea logică abductivă vine din faptul că semantica stabilă generalizată se poate calcula prin intermediul unui JTMS. În acest scop, Satoh şi Iwayama [SatIwa91] au transformat un cadru abductiv într-un program logic normal cu constrângeri de integritate (pentru modelarea observațiilor) (modele stabile generalizate ale cadrului abductiv= modele stabile ale programului logic normal). Apoi, pentru programele normale cu constrângeri de integritate au extins o procedură ce calculează extensiile ground ale unui TMS (Doyle) într-o procedură (TMS) pentru calculul abducției (un model stabil al unui program logic fără constrângeri de integritate este echivalent cu o extensie ground a unui TMS). Calculul este bottom–up; deşi nu este orientat spre scop, scopurile pot fi reprezentate şi tratate drept constrângeri de integritate. Comparativ cu alte abordări care întâi generează modelele stabile şi abia apoi testează constrângerile de integritate, un avantaj al metodei Satoh–Iwayama stă în faptul că foloseşte constrângerile de integritate dinamic în timpul procesului de generare a modelelor stabile, pentru a reduce mai eficient spațiul de căutare.

O abordare mai riguroasă a legăturii între TMS şi abducție (şi CSP-abducție) găsim la de Kleer [deKleer89b], în sistemul general CMS (Clause Management System- sistem de management a clauzelor). CMS sunt utile pentru raționamentul abductiv în special şi pentru eficientizarea căutării în general.

Definiție[deKleer89b]. O clauză S este un suport pentru C în raport cu $\sum$ ddacă

$\Sigma \not\models S$ ($\Sigma \cup \{\neg S\}$ este satisfiabilă);

$\Sigma \models S \cup C$ ($\Leftrightarrow \Sigma \models \neg S \supset C$ sau $\neg S \rightarrow C \equiv S \cup C$)

$\sum$ este un set de clauze trimise către CMS de către Reasoner; C= query.

Rolul CMS-ului este de a găsi toate clauzele suport minimale pentru C în raport cu $\sum$. Minimalitatea este importantă pentru Reasoner din două motive: 1) Abducție; 2) Căutare eficientă. CMS-ul este un fel de "cache" inteligent care poate fi exploatat să organizeze şi să controleze căutarea, astfel că mare parte din calculul Reasoner-ului poate fi evitat (înainte de a face un calcul, acesta verifică dacă acesta a mai fost realizat anterior). Înainte de a alege o valoare pentru o anumită variabilă, Reasoner-ul interoghează întâi CMS pentru a vedea dacă variabila e determinată de alegerile curente; dacă nu e determinată- se alege o valoare ce poate fi adăugată consistent la mulțimea curentă de opțiuni.

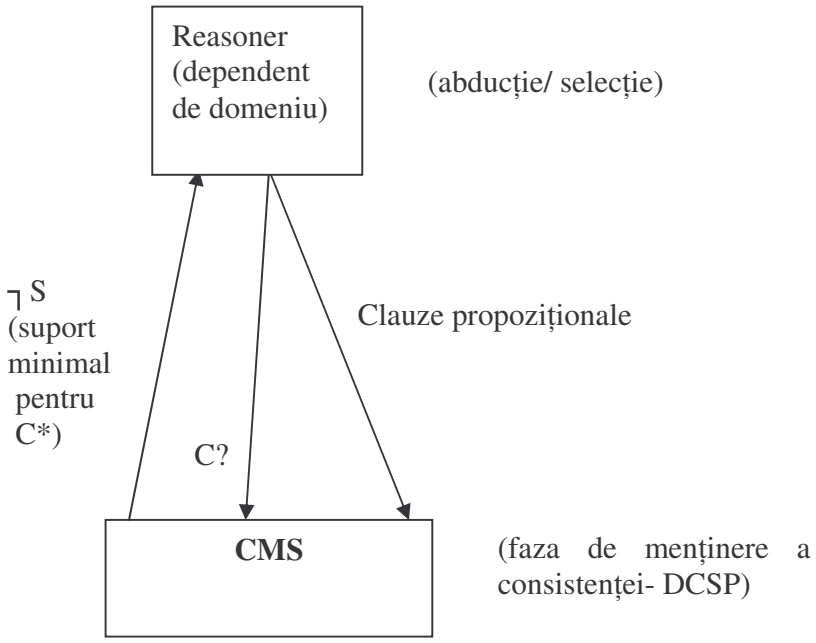
Definiție[deKleer89b]. *Prim implicant* al unei mulțimi ∑ de clauze este o clauză (disjuncție de literali) C :

1. ∑|=C;
2. Nu există C' submulțime proprie a lui C astfel încât ∑|=C'.

Teoremă[deKleer89b].Caracterizarea CMS. Fie ∑= mulțime de clauze; C={l} o clauză unitară; Atunci S este suport clauzal minimal pentru C în raport cu ∑ ddacă

Există un prim implicant Π al lui ∑ astfel încât l ∈ Π, S= Π-{l}

Partea interesantă a CMS stă în faptul că atât ATMS cât şi CSP sunt cazuri speciale ale sale [ReiKle87].CMS-ul este o generalizare ATMS: în ATMS, CMS este restricționat la clauze Horn (=justificările) şi clauze negative (inconsistențele de tip "nogood"), iar C sunt întotdeauna literali.



*: explicații abductive minimale pentru C

Caracterizarea ATMS[deKleer89b]. Fie J= o mulțime de justificări trimise către ATMS de către Reasoner, {α} o interogare (query) (α= simbol propozițional, ipoteză sau nu).
Atunci răspunsurile la această interogare sunt date de:

$\{ A_1 \wedge ... \wedge A_k \mid k \geq 0,$ şi $\neg A_1 \vee ... \vee \neg A_k \vee \alpha$ este un prim implicant al lui J$\}$ (7.11)

(abducție: ipoteză= explicație (ipoteza explicativă)).

Caracterizarea CSP[deKleer89b]. O ipoteză poate fi un simbol propoziţional ce face parte dintr-o soluţie propusă pentru CSP (o valoare pentru o variabilă din CSP).

7.5.CSP în calculul semanticii admisibile din argumentaţie

Am văzut până aici avantajele sistemelor de argumentare directă asupra programării logice şi TMS, în abordarea abducţiei şi consistenţei, cât şi avantajele semanticii admisibile pe care ne bazăm în definiţia diagnosticului. Am arătat deasemenea în secţiunea anterioară legătura riguroasă care există între algoritmii CSP şi abducţie.

Capitolul VI a introdus bazele teoretice ale utilizării algoritmilor CSP în calculul semanticii admisibile. Am arătat acolo cum putem folosi backtracking dinamic pentru DCSP pentru a aproxima semantica admisibilă (o variantă net superioară [vezi rezultatul de complexitate backtracking dinamic] algoritmilor exponenţiali bazaţi pe calculul părţilor unei mulţimi care au fost dezvoltaţi până acum pentru această semantică-[Doutre02], în afară de avantajul modelării dinamice a spaţiului de căutare).

Am văzut, în plus, în secţiunile precedente, că în comparaţie cu ATMS sau LTMS, ţinând cont de criteriul dimensiunii bazelor de inconsistenţe (*nogoods*), metoda backtracking dinamic are un avantaj evident întrucât reţine doar inconsistenţele asociate cu valorile actuale ale variabilelor (celelate sunt eliminate), fiind orientat spre descoperirea unei singure soluţii (poate fi însă extins cu uşurinţă, dacă specificul problemei o cere). În concluzie, nu numai că algoritmii DCSP sunt o abordare procedurală mai eficientă decât TMS pentru abducţie, dar aceştia sunt în plus adecvaţi pentru ceea ce avem nevoie în aplicaţia noastră, adică:

- implementarea admisibilităţii;
- lucrul simultan cu mai multe soluţii alternative (pentru a fi mai uşor comparate în vederea diagnosticului diferenţial);
- incrementalitate;
- adaptare la un mediu dinamic de evidenţe;
- eficienţi în general.

În încercarea de a descoperi o tehnică de căutare care să se mişte liber în spaţiul de căutare şi să memoreze ce s-a cercetat şi ce nu, autorii din [GinMcAll94] au încercat îmbinarea GSAT şi backtracking dinamic. GSAT este o cunoscută metodă de căutare nesistematică ce foloseşte o tehnică locală de tip hill-climbing. Este un fel de algoritm genetic care porneşte de la o asignare completă, trece prin soluţii vecine (ce diferă pe o singură poziţie) încercând să minimizeze numărul de constrângeri încălcate, în căutarea unei eventuale soluţii. Backtracking dinamic, în schimb, face o căutare sistematică, folosind o cantitate polinomială de informaţie pentru a îngusta spaţiul de căutare. Am menţionat această abordare aici, întrucât ni se pare apropiată de ceea ce încercăm şi noi să realizăm: pasul de selecţie al algoritmului corespunde GSAT- are scopul de a limita

spațiul de căutare, dar poate fi implementat prin intermediul unor tehnici inteligente care, spre deosebire de GSAT, să evite blocarea în minime locale [Mun05].

7.6.Alte avantaje ale argumentelor asupra TMS

În această secțiune am dori să subliniem încă un aspect în ceea ce priveşte avantajele abordării structurilor cauzale localizate pentru reprezentarea argumentelor (ca cele din abordarea pe care o propunem), față de abordarea TMS. Este vorba despre o suită de lucrări ale lui de Kleer & comp [deKleer90], în care autorii arată cum poate beneficia un TMS de localitatea din reprezentarea cunoştințelor pentru a-şi controla şi restrânge inferențele. Autorii pleacă de la ideea că mulți rezolvitori AI sunt inerent locali: fiecare parte componentă a problemei are un model de comportament fixat; şi o mare parte a raționamentului poate fi văzută ca propagare (când se inferează un semnal nou, modelele componentelor implicate sunt consultate pentru noi inferențe posibile), TMS fiind potrivit pentru această propagare. Se ştie în plus că Propagarea Booleană a Constrângerilor (BCP -Boolean Constraint Propagation) este eficientă dar incompletă pe CNF, deci TMS-urile sunt incomplete logic când sunt astfel folosite ("blind spots" – apar deoarece localitatea prezentă în modelul original se pierde în TMS). Pe de altă parte, satisfiabilitatea propozițională este NP-completă, dar costul completitudinii logice poate fi redus mult prin exploatarea localității. Astfel, local, în cadrul fiecărui modul, TMS este complet logic dacă se foloseşte BCP pe formule logice (BCP: completă pe formule logice, dar mai costisitoare).

Prin transformarea formulelor logice în CNF se pierde informația legată de localitate:

din $\neg x \vee y \vee z$ şi $x \vee y \vee z$ şi $y=F$, BCP nu deduce $z=T$.

Propunerea facută de către de Kleer este ca mulțimea de formule (i.e. combinații de clauze) ce reprezintă un model local să fie transmisă TMS ca un singur modul, şi TMS-ul să folosească o metodă de inferență completă local pe module. Extrapolând, localitatea este exploatată în modelul nostru, prin faptul că DCSP lucrează la un moment dat pe contextul asociat doar cu ipotezele de diagnostic selectate, context conținut într-un model cauzal local.

7.7.Concluzii şi contribuții

Am văzut în capitolul de față că atât abordările CSP, cât şi sistemele TMS reprezintă două formalisme înrudite pentru calculul abducției, şi am prezentat avantajele şi dezavantajele lor şi domeniul de aplicabilitate. Am justificat, deasemenea, de ce am ales algoritmi specifici CSP în favoarea TMS (ca implementare pentru o problemă cu constrângeri), şi de ce am ales sistemele de argumentare în favoarea sistemelor TMS. Prima parte a capitolului s-a dorit o justificare a faptului că sistemele de argumentare directă reprezintă un formalism cel puțin la fel de bun ca programarea logică în problema abducției (şi, prin urmare, merită atenția cercetătorilor în domeniu), şi a arătat, în plus, meritul semanticii admisibile față de cea stabilă prin prisma programării logice.

O contribuție importantă a capitolului stă în încercarea de sistematizare a ideilor ce stau la baza formalismelor enumerate mai sus, cu scopul de a descoperi mecanismele elementare ce apropie între ele abordări distincte, dar alăturate pe criteriul aplicabilității la aceeaşi problemă.

Ni s-a părut interesant să relevăm legătura între CSP (tehnica pe care am ales-o şi noi pentru implementare) şi abducție în general, şi să detaliem, măcar parțial, unele mecanisme ale sistemelor TMS. Acestea din urmă au fost alese atât pentru că reprezintă o modalitate notabilă de abordare a raționamentului nemonoton, cât şi pentru faptul că am considerat util să aducem în prim plan, pentru comparație, o alternativă de exploatare a conceptului de "localitate", întâlnit atât în rețelele cauzale cât şi în sistemele de argumentare directă.

Ideile prezentate ne-au servit, în final, la accentuarea avantajelor utilizării de algoritmi CSP pentru semantica admisibilă din argumentație.

VIII. Argumentarea deciziei de diagnostic în DiaMed

Capitolul de față descrie în detaliu algoritmul asociat cu nivelul de discriminare şi explicare al sistemului hibrid DiaMed [MunDum06a], algoritm care este o îmbunătățire a versiunii prezentate în Capitolul VI, astfel încât dinamica constrângerilor să fie determinată practic de rezultatele testelor medicale. Un exemplu relevant încheie capitolul, cu prezentarea facilităților oferite de program.

8.1.Reprezentarea cunoştinţelor

Modelul pe care îl propunem conține asocieri cauzale clase-caracteristici reprezentate cu ajutorul următoarelor componente.

Clasele de diagnostic (de exemplu bolile, în cazul medical) sunt modelate în abordarea noastră sub forma unor *rețele cauzale* de tip special, cu mai multe tipuri de noduri şi arce, şi care rezumă modelul cauzal profund al modelului. Astfel, există trei tipuri de noduri:

- noduri rădăcină, asociate claselor (ipotezelor de diagnostic), corespund unor cauze profunde, primare, ale manifestărilor observate (redate cu roşu, cele cu roz sunt folosite pentru reprezentarea complicațiilor);

- noduri corespunzătoare manifestărilor (simptomelor) de profunzime (inaccesibile sau accesibile cu dificultate, prin teste invazive, costisitoare, consumatoare de timp;

- noduri ce corespund unor simptome superficiale, direct/ uşor observabile.

Nodurile asociate simptomelor (observate sau nu) sunt de două tipuri: **necesare** sau **suplimentare**. Cele colorate în albastru reprezintă manifestările profunde, mai greu accesibile, iar cele în verde sunt manifestările superficiale, accesibile prin observații directe asupra pacientului. Infirmarea la testare a unui simptom necesar atrage după sine eliminarea completă a ipotezei de diagnostic corespunzătoare.

Arcele ce leagă nodurile în cadrul rețelei cauzale sunt de asemenea de mai multe tipuri:

1. Implicaţii necesare: prezența cauzei atrage întotdeauna după sine apariția consecinței;

2. Implicaţii posibile: cauza *poate* atrage după sine apariția consecinței, dar nu este obligatoriu; (observăm că această incertitudine îşi are de fapt originea într-o incompletitudine a modelului: există anumite elemente/condiții suplimentare care influențează validitatea implicației dar care nu au fost modelate explicit);

3. Atacuri bi- sau unidirecţionale): modelează, în principiu, elemente ce nu pot apare simultan în cadrul aceluiaşi sistem .

Fiecare clasă de diagnostic este definită de câte o astfel de rețea cauzală, ce conține toate elementele posibile relaționate cu clasa respectivă, organizate pe nivele din ce în ce mai superficiale, si deci mai accesibile observației directe asupra sistemului. Nodurile intermediare între rădăcină şi frunze sunt, de obicei, fie inaccesibile, fie greu accesibile (prin teste costisitoare, de lungă durată şi/sau invazive), şi gradul de accesibilitate creşte pe măsură ce ne apropiem de frunze (Figura 8.1)

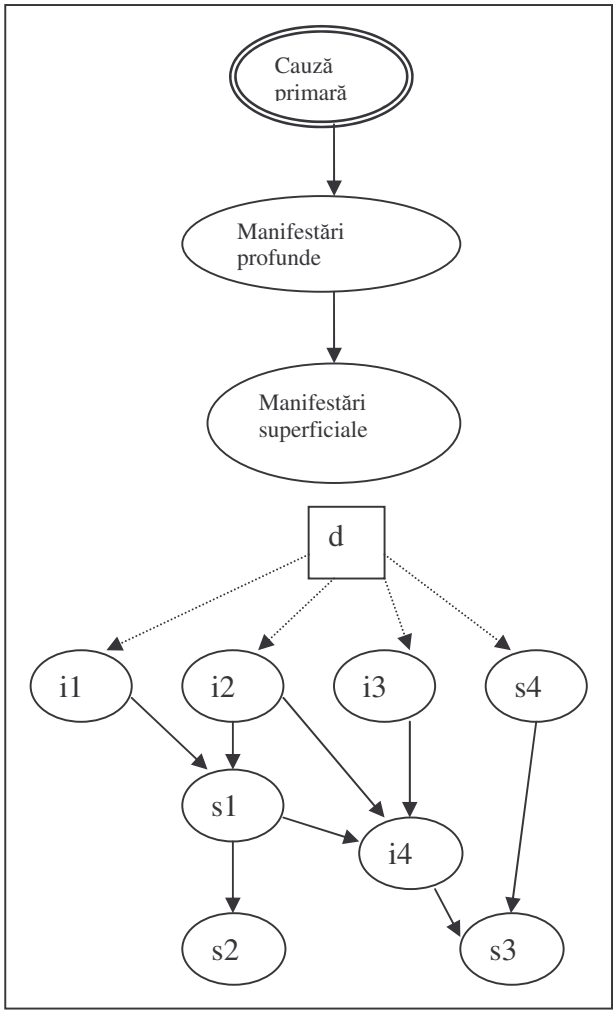


Figura 8.1.Structura generală a unei rețele cauzale şi un exemplu de subrețea (d-boală, i-noduri intermediare inaccesibile, s -simptome testabile)

Definiție. Se numeşte *argument* asociat unei anumite clase o instanță a rețelei cauzale ce defineşte clasa. O instanță a unei rețele cauzale este o submulțime a nodurilor sale ce conține cel puțin o manifestare observată (restul fiind presupuse adevărate).

Propunem în continuare o definiție originală şi naturală a unui diagnostic multiplu.

Definiție. Un *diagnostic multiplu* (i.e. o mulțime nevidă de boli posibile la un pacient dat) este o mulțime admisibilă[4] de ipoteze[5] care acoperă toate observațiile, şi este minimală cu această proprietate.

Exemplu. Fie 3 argumente A={leucemie}, B={nivel PT,PTT normal}, C={medicamente care afectează nivelul PT,PTT}. Relația de atac este dată de graful: A←B←C. Atunci A este acceptabil în raport cu C, C fiind o apărare a lui A față de atacul B. Deci {A,C} este o mulțime admisibilă, şi în acest caz "leucemie" este un diagnostic posibil în contextul definit de C.

Observații. Dung a observat că relațiile de atac între argumente în raționamentul implicit depind doar de asumpțiile (ipotezele) pe care se bazează acele argumente, şi a propus o definiție a argumentului ca fiind o deducție ale cărei premize sunt toate asumpții; în plus, argumentul ***a*** atacă argumentul ***b*** dacă ***a*** atacă o asumpție pe care ***b*** se bazează. (Un argument atacă o asumpție ***α*** când concluzia lui ***a*** este contrariul ***not α*** al lui ***α***).

Observații.

- Un argument corespunde configurației particulare de simptome prezentă în cadrul cazului observat, prin care clasa se manifestă, şi care poate varia de la pacient la pacient.
- Merită să comparăm în acest punct anumite aspecte pentru a elimina eventualele confuzii: dacă rețelele bayesiene şi cauzale din Capitolul III sunt structuri suport ce sintetizează paşi de inferență statistică, şi inferența se relizează exclusiv pe baza acestor structuri, după anumiți algoritmi probabilişti, rețelele cauzale din abordarea noastră sunt un mod de reprezentare compactă a unor module din modelul medical, inferența realizându-se nemonoton cu ajutorul relației de atac. Un argument poate fi privit aici drept o instanță a unei clase a modelului, instanță ce este adaugată/ ştearsă dinamic din contextul curent ținând cont de atacuri.

Definiție. *Graful de atac abstractizat* este, simplu, graful de atac luat în considerare între argumentele ce conțin nodurile ce se atacă (presupunem că nu există noduri ce se atacă în cadrul aceluiaşi argument).

Atacurile folosite concret în modelul nostru sunt în principiu de trei tipuri. În primul rând se atacă două cauze alternative pentru acelaşi efect (cum se vede şi în Figura 8.2.), în ideea că doar una dintre ele a provocat efectul respectiv şi este nefolositor să le considerăm pe ambele "in". Situațiile care ar infirma acest model sunt probabil foarte

[4] A se vedea definiția din Secțiunea 6.2.2.
[5] considerăm drept ipoteză orice boală activă, ce poate fi asimilată cu argumentul ce o susține.

rare, dacă există. În cadrul celor 30 de boli alese pentru exemplificare şi al exemplelor rulate noi nu am întâlnit o astfel de infirmare.

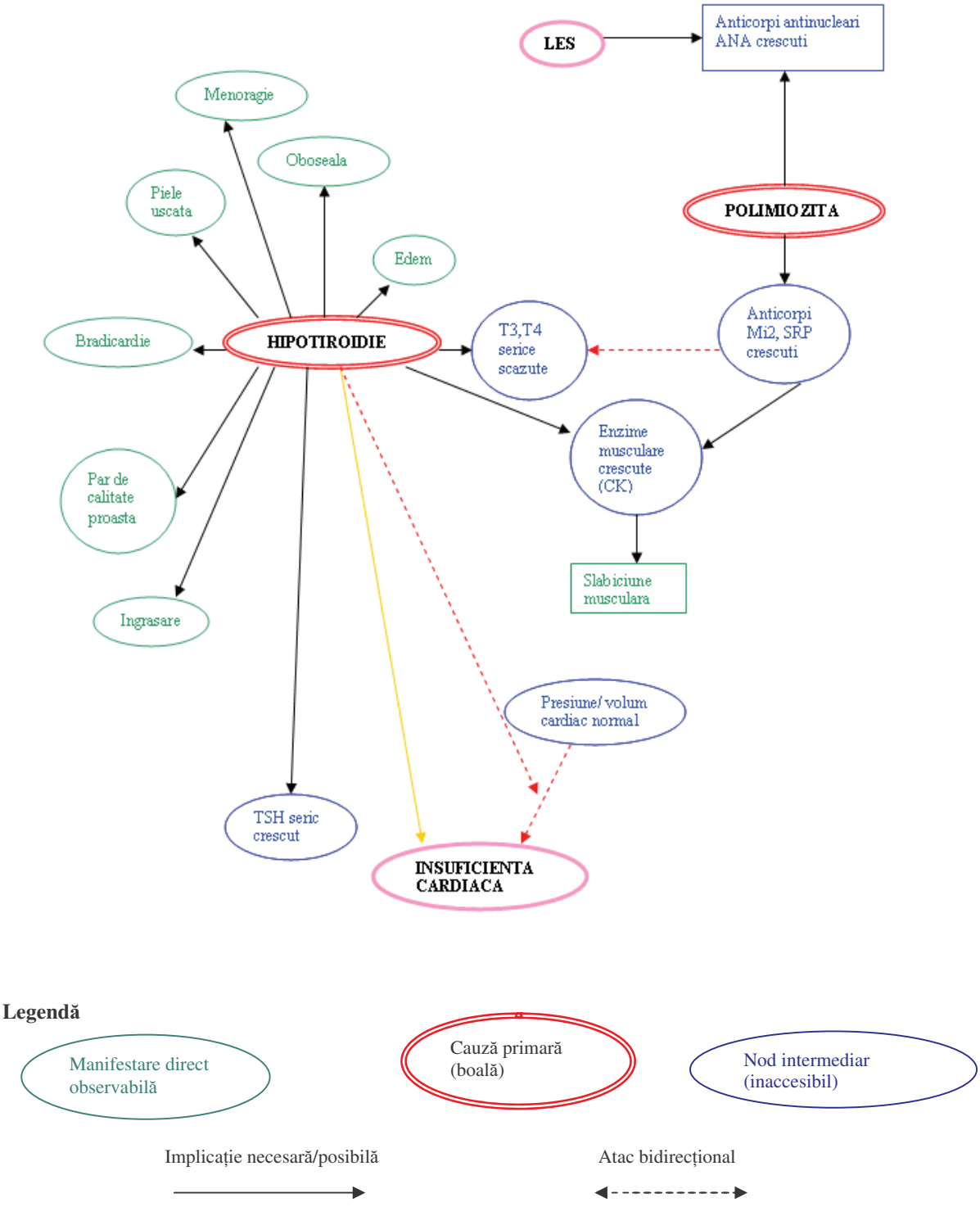


Figura 8.2. Exemplu de dependențe cauzale

Al doilea tip de atac este descris de următoarea situație: dacă o boală ***b*** se poate manifesta sub forma unei boli ***a***, astfel că deseori ***b*** se confundă cu ***a***, atunci considerăm că ***b*** atacă ***a***. Ideea este că inițial se presupune ***a***, dar teste suplimentare infirmă ***a*** şi confirmă ***b***. Acest atac sintetizează de fapt experiența practică a experților asupra unei anumite ordini temporale posibile de apariție a ipotezelor în cadrul raționamentului. Spre exemplu, insuficiența cardiacă atacă ciroza, deoarece uneori insuficiența se confundă cu ciroza.

În sfârşit, al treilea tip de atac este definit de inconsistența logică: dacă în insuficiența cardiacă este necesar ca presiunea şi volumul cardiac să fie afectate, am considerat că "presiune şi volum cardiac normal" atacă direct (prin abstractizare) insuficiența cardiacă. Există însă excepții care complică această situație; spre exemplu, în hipotiroidie avem insuficiență cardiacă fără ca presiunea sau volumul să fie afectate (mecanismul de producere fiind diferit: hipocalcemia) aşa că am considerat că hipotiroidia atacă direct "presiune volum cardiac normal", în contexul "insuficiență cardiacă". Altfel spus, hipotiroidia este o apărare a insuficienței cardiace în fața "presiune/volum cardiac normal" (vezi şi graful de atac din Figura 8.6). De fapt, în această excepție, avem tot atac de tipul "cauză alternativă", aşa că nu l-am tratat separat.

8.2.Modelarea problemei în cadrul DCSP

Notații.

- Ipoteze de diagnostic: $d_1,\ldots,d_N$ (N clase (boli) în total);
- Rețelele cauzale asociate ipotezelor de diagnostic: $C_1,\ldots,C_N$;
- Argumente: $A_1,\ldots A_L$ (L instanțieri posibile ale rețelelor cauzale);
- **Ip**=$\{d_1,\ldots,d_t\}$ -mulțimea de ipoteze selectate, eventual ordonate descrescător după ponderea primită la selecție;
- *Manifestări_Confirmate*=$\{mc_1,\ldots,mc_k\}$ mulțimea manifestărilor confirmate (true);
- *Manifestări_Infirmate*=$\{mi_1,\ldots,mi_p\}$ mulțimea manifestărilor infirmate (false);
- *Context= Manifestări_Confirmate* $\cup$ *Manifestări_Infirmate* -mulțimea totală de simptome confirmate sau infirmate prin testare;
- G_a - graful de atac abstractizat restricționat la mulțimea **Ip**;
- Ne vom folosi de o partiționare a mulțimii **Ip** în: *Ipoteze_infirmate, Ipoteze_considerate, Ipoteze_noi.*

Aşadar, pornind de la *Manifestări_confirmate,* se face selecția prin decizie fuzzy (faza abductivă a generării de ipoteze) şi se ajunge la mulțimea de ipoteze active:$\{d_1,\ldots,d_t\}$, cu rețelele cauzale asociate $C_1,\ldots,C_t$. Într-un prim pas, se calculează tot ceea ce este legat de manifestările confirmate/ infirmate prin relații deterministe (de exemplu, un nod atacat al cărui atac este *true* şi atacul nu e defensibil, va fi sigur *fals*). Nodurile rămase din rețelele activate netestate încă formează mulțimea de variabile active *Var_Active*=$\{V_1,\ldots,V_p\}$ pe care se aplică algoritmul backtracking dinamic pentru CSP dinamice [VerSchi] într-o formă adaptată (faza de testare a consistenței). Variabilele active sunt asociate de obicei cu noduri defensibile: au statutul "in" (presupus adevărat) sau "out" (presupus fals) la un

moment dat, dar pot fi reinstaurate/ contrazise pe parcurs. Constrângerile active ale algoritmului CSP -*Constr_Active*- sunt cele în care este implicată cel puțin o variabilă activă. După cum s-a definit, un diagnostic multiplu este o soluție minimală- adică o mulțime admisibilă, complet acoperitoare şi minimală. (Practic, în implementare am simplificat lucrurile prin introducerea în *Var_Active* direct a ipotezelor de diagnostic, acolo unde a fost posibil).

Într-o reprezentare sistemică, avem:

- **Intrări**= Simptome observate;
- **Stare sistem**= Contextul selectat de boli;
- **Ieşiri**= soluțiile posibile, i.e. combinații admisibile (acoperitoare sau nu) de ipoteze.

Evidența nodurilor testate se ține cu ajutorul mulțimilor *Manifestări_confirmate* (conține ipoteze de diagnostic sau simptome ce le susțin) – utilizate pentru explicația finală şi *Manifestări_infirmate* - utilizate pentru a nu reactiva prin selecție ipoteze/ noduri înfrânte deja.

Toate cele trei tipuri de implicații ce pot apare într-o rețea cauzală sunt reprezentate drept constrângeri binare (reprezentare inspirată din BCP- Boolean Constraint Propagation), astfel:

- Implicații necesare: $a \rightarrow b$: $C_{ab}=\{00,11,01\}$;
- Atacuri: $a \mapsto b$: $C_{atac}=\{10, 01,00\}$;
- Implicațiile posibile nu restrâng cu nimic domeniul posibil de valori pentru perechea de variabile implicate, astfel că acesta va fi {00,01,10,11}.

Fie:
Cons(v)= constrângerile în care apare variabila v;
Var(c)= variabilele ce apar în constrângerea c.

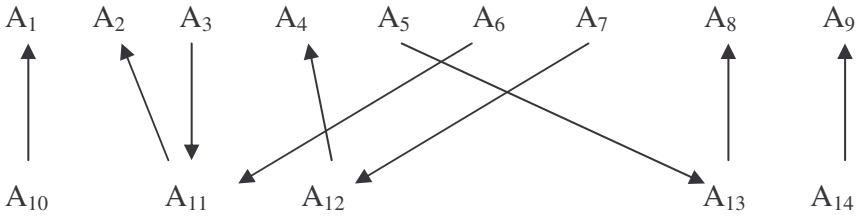
Mulțimea constrângerilor curente *Constr_Active* se defineşte natural:

$$Constr\_Active = \bigcup_{v \in Var\_Active} Cons(v). \quad (8.1)$$

Definiție. Se numeşte *soluție* a problemei o asignare completă şi consistentă de valori de adevăr pentru toate variabilele active (activate prin selectarea anumitor ipoteze), şi care acoperă toate manifestările confirmate. O soluție cu un număr minim de noduri-ipoteze se numeşte soluție *minimală*. Aceasta corespunde definiției diagnosticului multiplu de mai sus.

Mai multe soluții (nu neapărat minimale)- se pot ierarhiza după *gradul de determinare,*un indice original descris în continuare.

Grad de determinare
Facem presupunerea (naturală, dealtfel,) că nu există noduri care să se atace în interiorul aceluiaşi argument. Aceasta înseamnă că putem abstractiza relația de atac de la nivelul simptomelor la nivelul ipotezelor. Astfel, avem un graf de atac aciclic G_a de forma (spre exemplu):



Definiție. Se numeşte *grad de determinare* al unui argument A numărul:

$$\det(A) = \left\lfloor \frac{|\, noduri\_necesare_c(A)\,|}{|\, noduri\_necesare_a(A)\,|} \right\rfloor +_u \left\lfloor \frac{|\,\text{noduri\_suplimentare}_c(A)\,|}{|\,\text{noduri\_suplimentare}_a(A)\,|} \right\rfloor, \quad (8.2)$$

unde $necesare_c\ (A)$ reprezintă mulțimea nodurilor necesare confirmate din argumentul A, iar $necesare_a(A)$ este mulțimea nodurilor sale necesare asumate (similar pentru "suplimentare"), iar "$+_u$" este varianta "unfair" a legii combinării evidențelor a lui Bernoulli:

$$e_1 +_u e_2 = e_1 + (1 - e_1) * e_1 * e_2 .$$

Definiție. Dacă S_1, S_2 sunt submulțimi de argumente, definim relația de ordine "$\lhd$" astfel: $S_1 \lhd S2$ ddacă $\det(S_2) > \det(S_1)$ sau $\det(S_2) = \det(S_1)$ şi S_2 minimală (conține mai puține argumente-ipoteze decât S_1). (Ne-am inspirat aici din noțiunea de "minimal-defence" din Definiția din Secțiunea 6.1.4).

Gradul de determinare poate fi efectiv folosit în algoritm pentru compararea calității soluțiilor alternative.

8.3.Algoritm

1.Se ordonează *Var_Active* pornind de la variabilele cele mai constrânse (aceasta reprezintă o euristică binecunoscută pentru eficientizarea algoritmilor CSP). Întrucât atacul este constrângerea cea mai restrictivă, se poziționează pe primele locuri variabilele implicate în atacuri;

2.Am adaptat algoritmul de backtracking dinamic [VerSchi] pentru mulțimea *Var_Active* [Verfaillie, Schiex], cu constrângerile asociate- *Constr_Active*, pentru calculul diagnosticului, după cum urmează.

(Noduri *In* sau *Out*, =noduri defensibile (ce pot fi ulterior atacate)).

Inițializări
Fie $V_1=\phi$ mulțimea de variabile asignate, V_2=*Var_Active* mulțimea variabilelor neasignate; se inițializează *Domeniile* variabilelor cu mulțimea *{in,out}*.

dbt(csp)
V=mulțimea de variabile a problemei CSP;
return **dbt-variabile(ϕ,V)**.

backward-checking(V,v,val)
-verifică valoarea curentă *val* a variabilei *v* față de valorile variabilelor din V (= V_1 curent reunit, eventual, cu variabilele fixate din asignarea inadmisibilă ce se încearcă a fi extinsă) şi întoarce prima constrângere încălcată (c) sau *succes* (dacă nu există o astfel de constrângere);
-concret, se verifică dacă valoarea *val* este admisibilă în contextul actual astfel:

1. dacă *val*="in", *val* este admisibilă dacă nu atacă (e atacată de) un nod asignat deja ca fiind "in";
2. dacă se încearcă atribuirea valorii *val*="out" nodului curent n şi dacă el atacă un nod m "in", trebuie să existe un nod "in" care atacă nodul n;

crează-explicație-eliminatorie(v,val,V)
-reține mulțimea de variabile V (o lista de variabile cu valorile lor asociate) ca o explicație pentru eliminarea valorii *val* din domeniul variabilei *v*;

şterge-explicații-eliminatorii(v,V)
-pentru toate variabilele din V şterge explicațiile eliminatorii care conțin variabila v-trebuie reactualizat domeniul pentru $V_4 \cup V_2$, V_4 de la pasul **(*)**;

dbt-valoare(V1,v,val)
-verifică dacă în contextul definit de V_1 (variabile asignate deja), valoarea *val* pentru variabila *v* este admisibilă:
v←val;
c=**backward-checking(V1,v)**;
if (c=*succes*) then return *succes*;
else {V_3←mulțimea de variabile din constrângerea c;
crează-explicație-eliminatorie(v,val, V_3);
return *failure*; }

dbt-variabile (V_1,V_2)
//V_1 este mulțimea variabilelor asignate, V_2 – a celor neasignate;

if($V_2=\phi$) then {if (V_2 este o acoperire completă şi admisibilă a simptomelor prezente)
afişează soluţia;
Soluţii=Soluţii $\cup$ Soluţia_curentă;

- aici se propune utilizatorului o ierarhizare a soluţiilor în funcţie de gradul de determinare, însoţite de explicaţiile corespunzătoare (argumentele asociate fiecărei ipoteze); utilizatorul are în plus posibilitatea de a opri algoritmul în acest punct;

if (V_2 inadmisibilă) se reţine într-o listă asignarea curentă pentru a putea fi eventual extinsă ulterior; variabilele sale vor fi fixate, rolul său fiind limitat la verificarea admisibilităţii;}

else {găsit=false;
v← o variabilă din V_2;
d= domeniul curent al variabilei v;
val← o valoare din d;
if (**dbt-valoare(V_1,v,val)**) then
{găsit=true; v←val; Asignare.add(v,val);
$V_1=V_1\cup\{v\}$; $V_2=V_2-\{v\}$;
dbt-variabile($V_1\cup\{v\}$, $V_2-\{v\}$);}
if (**găsit**=*false*) then **dbt-bt-variabilă(V_1,V_2)**;
(nu mai am valori pentru v, trebuie să deasignez un conflict al său)
}//end else

dbt-bt-variabilă (V_1, V_2, v)
V_3← mulţimea de conflicte a lui v (= uniunea explicaţiilor eliminatorii ale valorilor eliminate ale lui v);
if ($V_3=\phi$) then {//nu există soluţii pentru variabilele curente;
date-noi=testează();
-se reia pasul de selecţie cu mulţimea reactualizată de teste (cele vechi + cele nou testate în cadrul pasului de discriminare curent):
creaza_Contexte (date);

AC=constrângerile ce definesc noul context de ipoteze;
RC=constrângerile asociate cu contextele în care un nod necesar a fost atacat de nodurile corespunzătoare simptomelor nou testate;

Reactualizare constrângeri pe baza testării:

Nodurile accesibile se testează şi:

- Se şterg din *Var_Active;*
- Dacă sunt confirmate şi înfrâng un nod necesar al unei ipoteze, toate constrângerile asociate ipotezei respective se şterg (se şterge din *Var_Active* şi nodula sociat ipotezei însăşi);

- Dacă sunt infirmate şi sunt noduri necesare pentru o ipoteză, această ipoteză se şterge din *Var_Active* şi constrângerile asociate ei se şterg din *Constr_Active*.

(Am definit astfel în mod natural şi un plan de testare original semi-automat pentru procesul de diagnoză).

ddbt(V_1,V_2, AC, RC);}

else { ($V_3 \neq \phi$):

Se revine şi se deasignează ultima variabilă din V_3 (adică ultima ipoteză facută care e în conflict cu v, variabila pentru care nu se mai găsesc soluții).

v'←ultima variabilă ∈ V_3 din V_1;
val'←valoarea curentă a lui v';
V_4←mulțimea variabilelor de după v' din V_1;
deasignează(v');
crează-explicație-eliminatorie(v',val', V_3-{v'});
şterge-explicații-eliminatorii(v', $V_4 \cup V_2$); **(*)**
dbt-variabile (V_1-{v'}, $V_2 \cup$ {v'})}

8.4.Exemplu

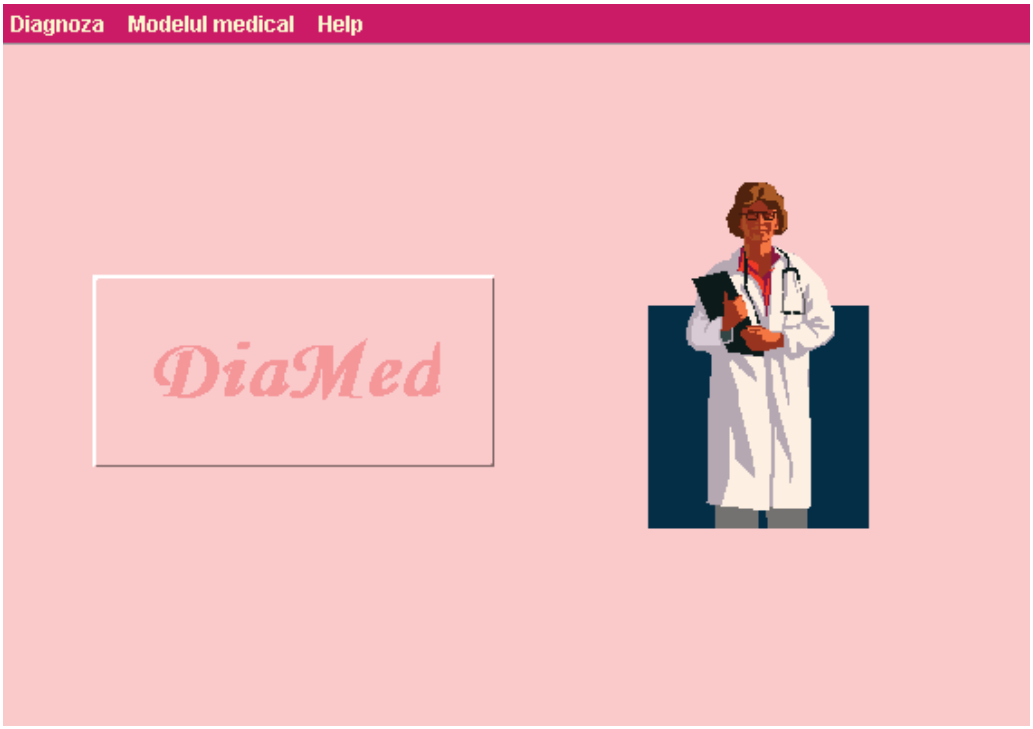


Figura 8.3. Programul DiaMed

Să presupunem că am observat inițial următoarele simptome (vezi planşele I şi VI):

Manifestări_confirmate={anorexie, aritmii, ascită, dispnee, edem, oboseală, slăbiciune musculară, anemie de boală cronică}. La o primă rulare, sistemul selectează prin decizie fuzzy următoarele ipoteze (boli), cu scor mai mare sau egal decât pragul ales (0.2):

- Angină pectorală 0.51 (BI1)
- Ciroză 0.22 (BH1)
- Hipotiroidie 0.25 (BE3)
- Infarct miocardic 0.26 (BI2)
- Insuficiență cardiacă 0.29 (BI3)
- Miocardită 0.70 (BI4)
- Pericardită 0.34 (BI5)

(Am trecut în paranteză codificarea prescurtată a bolii folosită în implementarea practică). În lista de ipoteze mai este selectat automat de către algoritm un nod cu statut special: "presiune/ volum cardiac normal" (SI32). Statutul special al nodului este dat de faptul că acesta apare în relațiile de atac în care sunt implicate ipotezele selectate (fără să fie simptom al vreuneia din boli). (Figura 8.4 prezintă graful de atac asociat contextului selectat).

Prin algoritmul CSP se obțin trei soluții admisibile minimal acoperitoare, şi anume:

- Hipotiroidie, insuficiență cardiacă, miocardită(BE3, BI3, BI4)
- Hipotiroidie, insuficiență cardiacă, infarct miocardic(BE3, BI3, BI2)
- Hipotiroidie, insuficiență cardiacă, angină pectorală (BI1, BI3, BE3)

Presupunem că utilizatorul (medicul) nu este mulțumit de rezultat şi vrea să completeze investigațiile cu noi teste. Aici el are posibilitatea de a completa în program simptome nou observate şi poate infirma simptome necesare ale bolilor active la acest moment pentru a mai restrânge, eventual, contextul. În acest scop, o listă de simptome necesare posibile în contextul dat este construită automat de către program, şi acesta este singurul indiciu automatizat care să ajute testarea. Simptomele necesare au acest statut preferențial față de cele suplimentare pentru că doar prin infirmarea lor se poate obține o restrângere a spațiului de lucru. (Figura 8.4)

În urma noilor observații, Manifestări_infirmate={"dureri piept ce cedează la nitroglicerină", "dureri piept ce nu cedează la nitroglicerină", "ritm de galop", "frecătură pericardică"}, ceea ce exclude din lista de ipoteze posibile nodurile {BI1, BI2, BI4, BI5}- adică boli pentru care anumite manifestări necesare lipsesc la pacientul observat. (A se vedea şi Anexele 1-3). Manifestările infirmate susțin faptul *"presiune/volum cardiac normal"*, ce apără ipoteza *„ciroză"*. Se descoperă în plus, la testare, prezența de "anticorpi Mi2, SRP crescuți". Acest simptom va adăuga în lista de ipoteze posibile

"Polimiozita" (BREUM3), ce a obținut un scor de 0.45. Graful de atac a devenit cel din Figura 8.6.

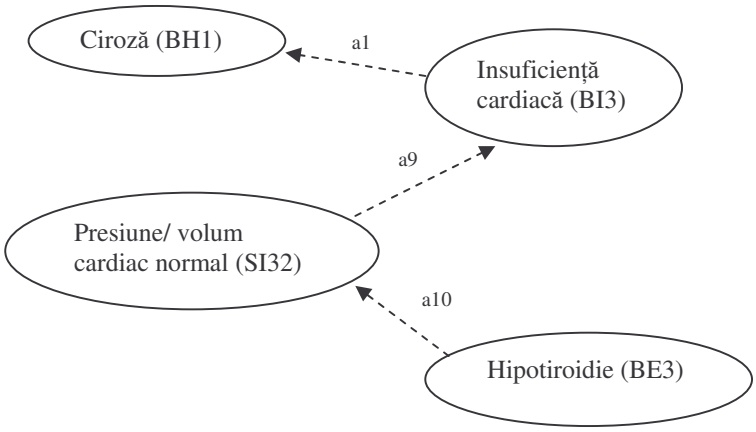


Figura 8.4.Graful de atac pentru primul context selectat

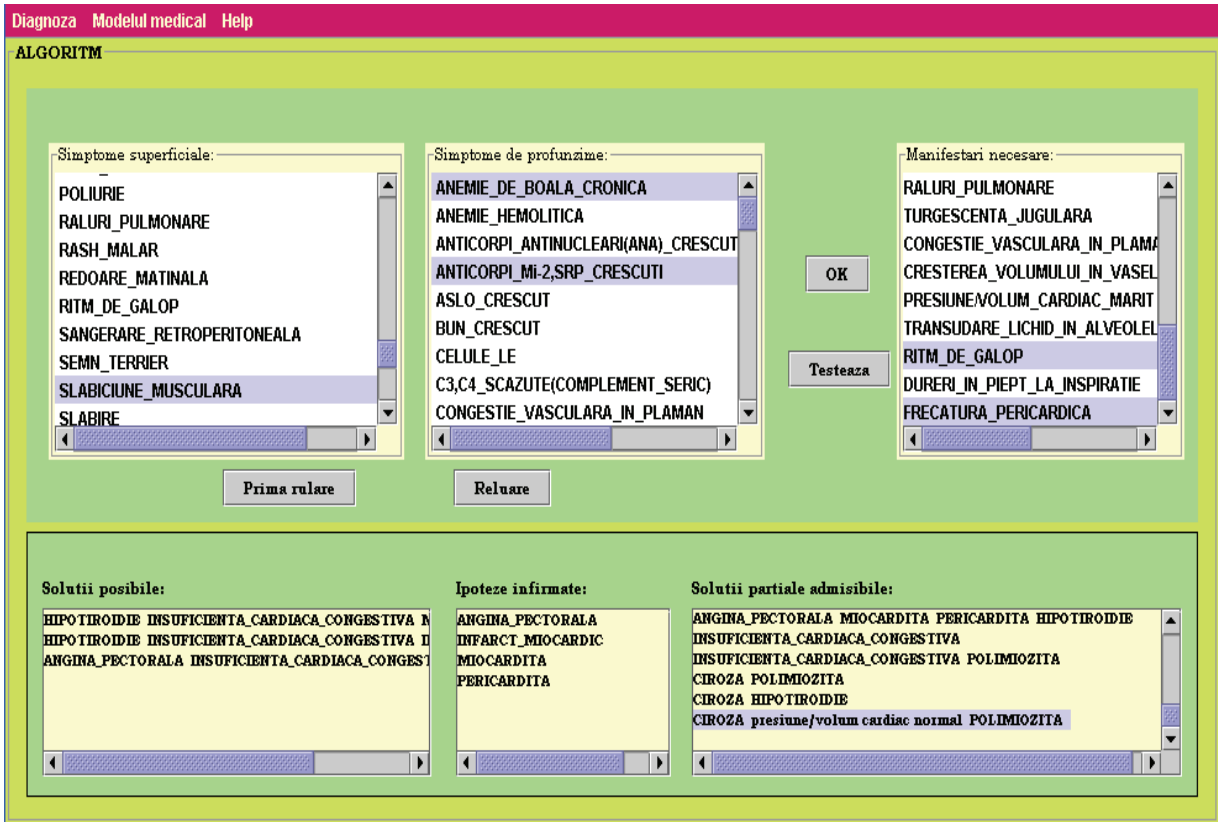


Figura 8.5. Exemplu (Anexa 1, planşele I, VI)

La reluarea algoritmului, se încearcă extinderea asignărilor inadmisibile de la pasul precedent cu apărări. Astfel, algoritmul preia, spre exemplu, asignarea inadmisibilă {“ciroza” “in”, “presiune/ volum cardiac normal” “in”} şi obţine în noul context o asignare admisibilă dar neacoperitoare: {“ciroza” “in”, “presiune/ volum cardiac normal” “in”, “polimiozita” “in”}, adică nodul BREUM3 este o apărare a mulţimii {SI32, BH1}, faţă de posibilele {BI3, BE3} activate de evidenţe.

Concluzia este că am reuşit să simulăm dinamica spaţiului de căutare în funcţie de datele noi şi infirmările ipotezelor anterioare foarte aproape de modul de raţionament al unui expert uman (acesta are totuşi ultimul cuvânt în decizia de diagnostic).

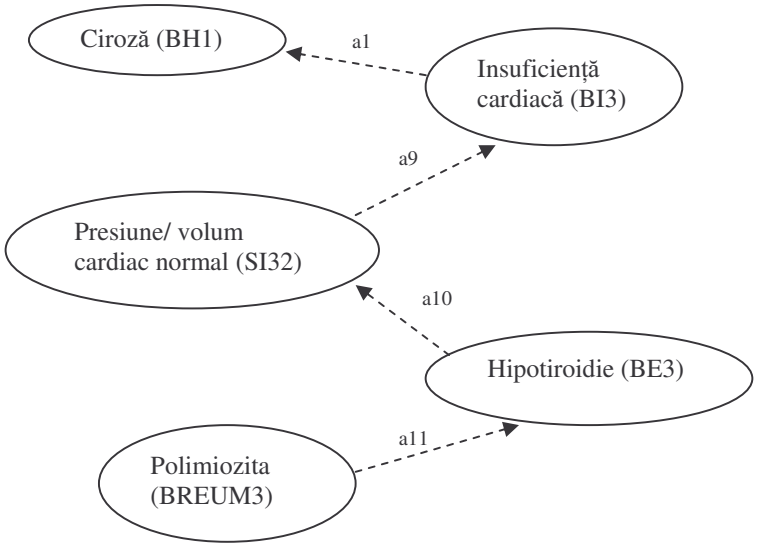


Figura 8.6.Graful de atac pentru al doilea context selectat

În secţiunea “Soluţii posibile” apar combinaţii admisibile de boli complet acoperitoare ale simptomelor, iar în secţiunea “Soluţii parţiale admisibile” sunt combinaţii admisibile neacoperitoare dar cu scor ridicat. Acest scor a fost folosit în cadrul algoritmului ca o euristică foarte utilă în eliminarea combinaţiilor mai puţin probabile, ceea ce a redus timpul de calcul de câteva ori (euristica abandonează asignările de scor scăzut, scorul fiind proporţional cu ponderea simptomelor acoperite în cadrul bolilor asignate ca “in”).

8.5.Concluzii şi contribuţii

Abordarea DCSP din algoritmul de mai sus reprezintă o traducere eficientă a mecanismului de remodelare dinamică a contextului de lucru pe baza evidenţelor aduse de procesul de testare, permiţând lucrul pe secţiuni limitate ale modelului. Adăugarea şi ştergerea de ipoteze ca urmare a selecţiei reluate dupa re-completarea testelor stă la baza

constrângerilor de activitate, adăugând/ştergând din variabilele problemei, pe baza unor reguli dependente de domeniu. Constrângerile de activitate asociate problemei aDCSP de activare/inactivare a variabilelor sunt conținute implicit în funcțiile de decizie fuzzy şi instanțele activate ale rețelelor cauzale (i.e argumentele).

Backtracking dinamic pentru DCSP este o metodă incrementală, ce pastrează subsoluția validă la tranziția spre alte porțiuni ale modelului. În plus, ni se oferă flexibilitate: ipotezele sunt generate chiar atunci când nu avem informații suficiente, dar avem posibilitatea să reconsiderăm aceste ipoteze dacă suntem confruntați cu necesitatea păstrării coerenței în fața a noi evidențe;

Prin algoritmul de mai sus s-a arătat, deasemenea, cum poate fi folosit DCSP pentru calculul (aproximativ al) semanticii admisibile; mai precis, CSP este o tehnică validă de determinare a admisibilității unei mulțimi de argumente, iar versiunea dinamică permite determinarea contextuală acestei proprietăți ce este, în mod natural, dependentă de context; avem aşadar o metodă constructivă şi eficientă de calcul aproximativ a semanticii admisibile.

O altă contribuție originală este dată de definiția naturală şi originală a diagnosticului multiplu folosind semantica admisibilă a argumentelor, şi de indicele original de calcul a gradului de determinare al unui argument. S-a propus pentru argumente o definiție originală, adaptată domeniului studiat, care structurează informația, grupând bolile pe nuclee posibile de interacțiune.

În câteva cuvinte, mecanismul de ştergere şi adăugare de constrângeri folosit surprinde cu fidelitate, într-o manieră sistematică şi eficientă în acelaşi timp, mecanismul de retragere şi adăugare de încrederi propriu raționamentului nemonoton. Avantajul principal față de sistemul CHECK al acestei abordări a nonmonotonicității la nivelul rafinării ipotezelor în DiaMed, stă în faptul că foloseşte o metodă tratabilă, eficientă, spre deosebire de abordarea logică a abducției indirecte ce determină sursa eventualelor inconsistențe în CHECK.

Am prezentat, aşadar, o abordare originală, pe bază de logica argumentării şi algoritmi DCSP a raționamentului nemonoton în general (şi care poate fi utilă în discriminarea şi rafinarea mulțimii de ipoteze de diagnostic în special). Aşa cum observau autorii în [BreDix97], "cercetările ulterioare în domeniul raționamentului nemonoton trebuie să pună mai mult accentul pe aspectele computaționale", întrucât acest domeniu "va avea un impact de durată asupra domeniului Inteligenței Artificiale în general, numai dacă va pune la dispoziție instrumente utile oamenilor în rezolvarea problemelor lumii reale". Acest citat subliniază importanța şi semnificația algoritmului prezentat în Secțiunea 8.3. În plus, am făcut legătura în această lucrare între semantica admisibilă şi diagnosticele multiple, justificând apropierea la nivel intuitiv a celor două noțiuni, şi am arătat cum semantica admisibilă se poate calcula prin algoritmi CSP. Am arătat, mai mult, cum varianta dinamică a acestor algoritmi se poate adapta pentru a pune la dispoziție un instrument deopotrivă eficient şi sistematic pentru raționamentul nemonoton. Prezentarea,

deşi poate fi utilizată pentru diagnoză în general, s-a orientat spre specificul diagnozei medicale, pe care am avut-o în vedere în cadrul sistemului DiaMed.

IX.Aplicaţii ale algoritmului DiaMed în diagnoza sistemelor dinamice

9.1.Introducere

Ne-am propus în capitolul curent să adaptăm abordarea DiaMed pentru diagnoza sistemelor dinamice şi să o comparăm cu alte abordări înrudite ale acestei probleme [MunDum06c]. Termenii de referinţă ai acestei comparaţii sunt sistemele de inferenţă fuzzy (în forma propusă de Frank în [Fran96]), şi algoritmul lui Koscielny [KoSedZac99] de ajustare dinamică a bazei de reguli fuzzy pentru diagnoză, folosit pentru eficientizarea calculului în cazul sistemelor complexe, de dimensiuni mari.

Am stabilit deja în Secţiunea 3.4.2 că diagnoza sistemelor dinamice presupune două etape principale, distincte: detectarea defectelor şi diagnoza defectelor. Am exemplificat la momentul respectiv câteva abordări posibile ele etapei de detectare a defectelor (prin generarea de reziduuri). Ne vom concentra aici numai asupra etapei de diagnoză a defectelor, întrucât aceasta este etapa în care abordarea DiaMed îşi poate dovedi aplicabilitatea şi utilitatea.

9.2.Decizie fuzzy sau sisteme de inferenţă fuzzy

a)Sisteme de inferenţă fuzzy în modelarea sistemelor şi diagnoza defectelor

Subsecţiunea prezentă ilustrează cum se poate utiliza logica fuzzy în motorul de inferenţe decizional pentru diagnoza defectelor.

Un model analitic precis şi fidel al unui sistem dinamic va conduce la inconvenienţa ca orice eroare de modelare să afecteze performanţele schemei de detectare şi izolare a defectelor (FDI), în special în cazul sistemelor nelineare şi incerte (adică majoritatea sistemelor reale) [Cal01]. Această problemă poate fi evitată prin aplicarea regulilor şi logicii fuzzy, înlocuind astfel un model exact prin descrierea comportamentului sistemului sub forma unor relaţii de tipul *if-then*. Logica fuzzy are în plus avantajul reprezentării naturale a informaţiilor simbolice, lucru de foarte mare ajutor în multe situaţii.

Frank [Fran96] propune o abordare bazată pe logica fuzzy pentru evaluarea reziduurilor, pentru a izola defectele. Fie $R=\{r_1, r_2,\ldots, r_m\}$ mulţimea de reziduuri. Fiecare reziduu r_i, (i=1,…m), este descris prin intermediul mulţimilor fuzzy $\{r_{i1}, r_{i2},\ldots,r_{is}\}$, ale căror funcţii de apartenenţă se pot identifica fie prin cunoştinţele unui expert, fie prin învăţare cu reţele neurale. Relaţiile cauzale între reziduuri şi defecte se pot exprima prin reguli *dacă-atunci* de forma:

$$DACĂ\ (efect= r_{ip})ŞI\ \ldots\ ŞI\ (efect= r_{jq})\ ATUNCI\ (cauza= defectul\ k) \quad (9.1)$$

Ieşirea clasificatorului fuzzy este vectorul de defecte *F*. Procesul de inferenţă fuzzy asociază fiecărei componente F_l, $l=1,\ldots,n$, o valoare între 0 şi 1 ce indică gradul în care sistemul monitorizat e normal (componenta corespunzătoare fiind F_0) sau e afectat de către defectul l, $l=1,\ldots,n$. Dacă se presupune că sistemul poate fi afectat doar de un defect la un moment dat, vectorul de defecte va conţine doar o componentă cu o valoare mai mare decât un prag dat, corespunzătoare defectului ce afectează sistemul. Dacă se lucrează cu ipoteza defectelor multiple, atunci toate componentele vectorului de clasificare ce depăşesc pragul sunt indicatori ai defectelor apărute în sistem. Avantajul folosirii acestui clasificator fuzzy rezidă în faptul că regulile fuzzy oferă detalii asupra mapării reziduurilor într-o anumită stare de defect. Dezavantajul apare în cazul sistemelor complexe, întrucât atunci numărul de reguli devine foarte mare. O soluţie a acestei probleme este propusă în [KoSedZac99]. Menţionăm, deasemenea, că în abordarea lui Frank este necesară câte o regulă pentru fiecare valoare anormală a aceleiaşi variabile. Acest dezavantaj este îmbunătăţit semnificativ de metoda din secţiunea următoare (practic o adaptare a etapei de selecţie a ipotezelor din DiaMed).

b)Decizie fuzzy pentru selectarea simptomelor

Modelul de decizie fuzzy folosit în DiaMed poate fi adaptat în diagnoza sistemelor dinamice fără modificări notabile, după cum se observă în continuare.

Cheia reprezentării din DiaMed este să definim defectele $F = \{f_1,...,f_N\}$ cu ajutorul unor funcţii de decizie ponderate ($F_1^{w_1},...,F_N^{w_N}$) [Mun05]:

$$F_i^{w_i} = h_i(R_{i_1}(o),...,R_{in_i}(o)) \quad (9.2)\text{ , unde:}$$

- $Simptome(f_i) = \{r_{i_1},...,r_{in_i}\}$, şi $i_1,...i_{n_i} \in \{1,...,K\}$ sunt indicii reziduurilor relevante pentru defectul f_i;
- $R_j : dom_j \to [0,1]$, este funcţia fuzzy ce defineşte reziduul r_j (dom_j este domeniul funcţiei R_j),
- w_i este vectorul ponderilor simptomelor în definiţia defectului f_i ;
- h_i este o funcţie de agregare bazată pe operatori fuzzy ce exprimă modul de raţionare al unui expert uman; şi
- $o = (R_1(o),...,R_K(o))$ este punctul K-dimensional al observaţiilor pentru sistemul monitorizat.

Ideea ce stă la baza diagnozei este aceeaşi cu cea prezentată în Capitolul V: se calculează indicii $F_i^{w_i}(o)$ care măsoară gradul de potrivire al observaţiilor cu criteriul complex ce descrie defectul f_i, se compară cu un prag, şi apoi se ierarhizează ipotezele selectate pe baza indicilor. Reluăm într-o notaţie modificată definiţia unui astfel de indice din Capitolul V.

Definiţie. Fie $f_i \in F$, şi *s* sistemul monitorizat. S*corul defectului* f_i la sistemul *s* este:

$$Scor(s, f_i)=S_g(M_{i_1}(o),\ldots, M_{in_i}(o)) \quad (9.3)$$

(semnificaţia notaţiilor a rămas aceeaşi). ■

Se moşteneşte astfel avantajul focalizării rapide a căutării pe direcţiile relevante, printr-o tehnică de calcul deopotrivă simplă şi eficientă, dar care ignoră interacţiunile posibile între ipotezele de diagnostic şi nu este foarte transparentă, dezavantaj depăşit de metoda descrisă în Secţiunea 9.3.

9.3.Studiu comparativ a două abordări pentru sistemele complexe

a)Un algoritm dinamic pentru un sistem complex

Autorii lucrării [KoSedZac99] propun o metodă de a evita explozia exponenţială a spaţiului de reprezentare din abordarea Frank. Rezumăm în continuare ideile lor.

Definiţie[KoSedZac99]. Un sistem de izolare a defectelor (FIS: Fault Isolation System) este un cvadruplu <F, R, V, φ>, unde:

- F este mulţimea de stări $F=\{f_0, f_1,\ldots, f_K\}$, f_0 este starea normală şi restul sunt stările de defect;
- $R=\{r_1,\ldots,r_J\}$ este mulţimea reziduurilor (simptomele);
- $V = \bigcup_{r_j \in R} V_j$, unde $V_j=\{v_{ji}\}$, $(i \in I_j)$, este variabila lingvistică ce descrie reziduul r_j, cu valorile sale fuzzy;
- $\varphi(f_k, r_j)=V_{kj}$ reprezintă mulţimea valorilor lingvistice ale reziduului r_j în cadrul defectului f_k (pot exista mai multe reguli pentru acelaşi defect).

Dacă luăm ca valori fuzzy posibile (N+, P, N- negativ –, pozitiv= normal, negativ+) regula corespunzătoare defectului f_k ar putea fi, spre exemplu, de forma:

$$DACĂ\ (v_1 = N+)\ şi\ldots şi\ (v_j = P)\ şi\ldots şi\ (v_J = N-)\ ATUNCI\ defect\ f_k \quad (9.4)$$

unde v_j sunt valorile fuzzy cu grad de realizare maxim pentru fiecare reziduu r_j.

Gradul de activare al fiecărei reguli este calculat ca un indice ce măsoară apropierea între un reziduu r_j şi valorile sale obţinute pentru un anumit defect f_k, diagnosticul final fiind format din defectele cu grad de activare maxim. Pentru diagnoza sistemelor complexe, abordarea este foarte greu de aplicat practic din cauza dimensiunii mulţimii defectelor şi reziduurilor. [KoSedZac99] propune restrângerea dinamică a mulţimii de reziduuri cu care se lucrează efectiv la o submulţime R*, în funcţie de o mulţime de defecte posibile la un moment dat F*.

1)Testarea ciclică a simptomelor.
Procedura de izolare a defectelor este lansată când un reziduu r_{j0} (posibil ataşat la o alarmă) are o valoare anormală ce depăşeşte un anumit prag T:

$$\exists r_j^0, (v_{ji} \neq P) \wedge (\mu_{ji} > T),$$

unde v_{ji} este valoarea fuzzy a reziduului r_j, iar μ_{ij} este gradul de realizare al valorii fuzzy v_{ji} pentru valoarea concretă a lui r_{j0}.

Inițializări:

- $R^*=R_N^*(m=0)=r_{j0}$ (R* este mulțimea de testare, adică mulțimea reziduurilor interesante într-un context de defecte, iar R_N^* este mulțimea reziduurilor despre care se ştie că au o valoare fuzzy anormală ce depăşeşte T când e calculată pentru valoarea concretă a reziduului respectiv;

- $F^*=\emptyset$.

2) Se crează FIS, ce include defectul curent.*

a) $F^* = \{f_k \in F \mid \exists r_j \in R_N^*(m) \cap \varphi(f_k, r_j) \neq P\}$
(f_k= defecte pentru care există un reziduu deja selectat, ce apare cu o valoare anormală în definiția lui f_k, $\varphi(f_k,r_j)$= valoarea fuzzy cu care apare r_j în definiția f_k);

b) $R^* = \{r_j \in R \mid \exists f_k \in F^*(m) \cap \varphi(f_k, r_j) \neq P\}$
(se extinde mulțimea de reziduuri cu reziduurile ce apar cu valori anormale în definiția defectelor selectate deja);

c) $R_N^* = \{r_j \in R^* \mid (v_{ji} \neq P) \wedge (\mu_{ji} > T)\}$.

Paşii 1 şi 2 se repetă cât timp RN* se mai modifică. În final, FIS*=<F*,R*, V, φ*> (φ* este restricția lui φ la noile mulțimi).

b)Rafinarea deciziei de diagnostic în DiaMed şi diagnoza sistemelor complexe

Al doilea nivel al sistemului hibrid DiaMed (folosit pentru discriminare şi explicare) reprezintă o alternativă a abordării [KoSedZac99] în evitarea exploziei exponențiale a spațiului de reprezentare. Deşi necesită un model mai detaliat şi mai bine structurat al sistemului observat, are avantajul unor mai bune facilități de explicare a deciziei.

Abordarea DiaMed este mai potrivită pentru procesele tehnice pentru care se cunosc, măcar parțial, relații de bază între defecte şi simptome, şi aceste relații se pot reprezenta sub forma unor relații cauzale:

defect →evenimente →simptome

Este suficient să reprezentăm adecvat sistemul (după schema din Figura 8.1), pentru a putea implementa diagnoza cu ajutorul procedurii descrise de Secţiunea 8.3, şi asupra căreia nu mai revenim întrucât adaptarea la noua problemă nu presupune practic nici o modificare aici.

Am subliniat la momentul potrivit faptul că algoritmul propus oferea o schemă dinamică şi eficientă de abordare a raţionamentului nemonoton, cu explicaţiile deciziilor înglobate în structura argumentelor. Avantajul său principal faţă de abordarea lui Koscielny a fost subliniat deja: acesta constă în nivelul superior de transparenţă a deciziei de diagnostic, bazată pe un model mai detaliat al sistemului observat. În plus, structura modelului pe care lucrăm permite o schemă mai inteligentă de restricţionare dinamică la un sub-model posibil din spaţiul total de cautare, iar admisibilitatea este o alegere mai naturală pentru reprezentarea diagnosticelor în situaţii reale.

9.4.Concluzii şi contribuţii

Am confruntat în capitolul curent modelul hibrid DiaMed cu problema diagnozei sistemelor dinamice. Am folosit sistemele de inferenţă fuzzy ca termen de referinţă al comparaţiei, întrucât ele reprezintă alternativa de diagnostic cea mai apropiată de decizia fuzzy. Dezavantajul DiaMed (necesitatea unui model mai complet al structurii şi interacţiunilor din sistem) este echilibrat de avantajele obţinute: atât precizie şi explicaţii superioare, cât şi o reprezentare mai naturală, mai nuanţată şi mai uşor de interpretat. În plus, eficienţa are de câştigat din faptul că modelăm efectiv mai puţine variabile în sistemul de diagnoză (spre deosebire de sistemele de inferenţă fuzzy).

Contribuţia capitolului stă în faptul că propune sistemul DiaMed ca pe o alternativă de modelare, care, atunci când dispunem de informaţiile corespunzătoare asupra sistemului, poate fi o modalitate eficientă de abordare a sistemelor complexe, cu multe variabile. Am arătat, prin prezentarea algoritmului lui Koscielny, că ideea de a restructura dinamic un spaţiu de căutare complex a mai fost folosită în automatică, şi am argumentat teoretic faptul că DiaMed prezintă anumite avantaje (enunţate mai sus), faţă de algoritmul respectiv.

Atât avantajele cât şi dezavantajele rămân deschise studiului practic prin implementarea pe sisteme cât mai variate şi prin compararea cu alte modalităţi de lucru.

X. Concluzii şi direcţii de dezvoltare

"-În ce direcţie trebuie să merg acum?"
„-Depinde unde vrei să ajungi."

(Lewis Caroll: "Alice în Ţara Minunilor")

Lucrarea prezentă şi-a propus o analiză a problemei diagnozei în general şi a diagnozei medicale în special. Originalitatea sa stă în contribuţii ce ating atât partea de modelare şi raţionament pentru diagnoză, cât şi unele aspecte de sistematizare şi comparare a abordărilor de până în prezent.

Capitolul II ne-a introdus în specificul diagnozei medicale, cu relevarea principalelor surse de dificultate şi eroare şi oferind o imagine detaliată a procesului de diagnoză în sine. Acest specific reprezintă justificarea teoretică a necesităţii abordărilor complexe din capitolele următoare. Capitolele III şi IV au propus o sistematizare comparativă a tehnicilor de diagnoză de până în prezent, introducând sistemul original DiaMed în contextul general şi prezuentând, spre comparaţie, două sisteme clasice de diagnoză medicala înrudite cu el: CASNET şi CHECK (DiaMed fiind o continuare posibilă a ideilor din aceste sisteme). O contribuţie importantă a celor două capitole stă în faptul că acestea prezintă unele idei de tratare posibilă a diagnozei ce nu au fost încă exploatate şi pot constitui punctul de plecare al unor studii viitoare. Capitolele justifică, deasemenea, utilitatea hibridizării combinative în DiaMed.

Am văzut, apoi, în Capitolul V, cum putem modela diagnosticele medicale prin intermediul unor funcţii de decizie fuzzy (folosite drept funcţii de discriminare pentru nivelul de selecţiede ipoteze din DiaMed), şi care sunt avantajele acestei modelări, comparativ cu sistemele de inferenţă fuzzy. Deşi funcţiile de decizie în sine nu sunt o contribuţie personală, felul în care le-am adaptat pentru descrierea bolilor (defectelor) este o idee originală.

Bazându-se pe nonmonotonicitatea raţionamentului de diagnoză şi pe necesitatea unor relaţii asimetrice de atac (necesitate exemplificată în Capitolul II), Capitolul VI introduce sistemele de argumentare directă ca formalism modern de reprezentare, şi propune o definiţie originală a diagnosticului multiplu, apelând la semantica admisibilă (specific argumentativă). Capitolul prezintă şi bazele implementării eficiente a calculului acestei semantici folosind backtracking dinamic pentru algoritmi de satisfacere a constrângerilor dinamici (DCSP). O contribuţie a acestui capitol este dată de traducerea mecanismului dinamic de testare din diagnoză într-o problemă de rezolvare a constrângerilor dinamică.

Capitolul VIII adaptează un algoritm eficient pentru problemele dinamice de satisfacere a constrângerilor (DCSP) la diagnoza medicală, definindu-i un model de ştergere/adăugare de constrângeri bazat pe rezultatele testelor medicale, obţinute în cursul investigării unui anumit pacient, şi implementează efectiv acest algoritm în sistemul DiaMed. În plus, am folosit un model original de reprezentare a cunoştinţelor în medicină, model bazat tot pe argumente.

Capitolul IX arată că tehnologia de modelare din DiaMed nu este restricţionată la probleme de tip medical şi se poate adapta sistemelor dinamice, în măsura în care există un model adecvat pentru defectele acestora, aducând unele avantaje în cazul sistemelor complexe.

Ca o concluzie finală, credem că lucrarea noastră constituie o pledoarie pentru sistemele de argumentare directă şi pentru aplicabilitatea lor la problemele dificile ale lumii reale.

Am subliniat deja realizările şi avantajele sistemului original DiaMed propus de lucrarea noastră. Ca întotdeauna însă, se poate merge şi mai departe. Încă nu am răspuns la o întrebare importantă: cum anume este corect să fie măsurată puterea de discriminare a unui test cu răspuns multiplu. Deasemenea, am lăsat o parte importantă a testării la decizia utilizatorului (nu am automatizat decât parţial acest proces), şi ar mai fi cu siguranţă multe de adăugat pe marginea testării, care poate constitui, singură, subiectul unui tratat. Deasemenea, modul de reprezentare poate fi îmbunătăţit, iar modelul medical necesită completări serioase din partea unei echipe de specialişti, pentru a face posibilă o evaluare în funcţie de un set semnificativ de date reale. Ar mai merita studiat în continuare impactul tehnicilor inteligente asupra căutării regulilor aplicabile în inferenţa propoziţională în general.

Aşa cum argumentam în Capitolul II, diagnoza medicală este una dintre cele mai dificile probleme de raţionament uman. A automatiza diagnoza medicală înseamnă, practic, a automatiza gândirea umană. Ceea ce probabil că nu se va realiza niciodată pe deplin. Şi poate că, de fapt, nici nu este de dorit.

Bibliografie

[Ake78]S.B.Akers (1978): "Binary Decision Diagrams", *IEEE Trans. On Computers*, vol. C-27, no.6, :509-516;

[And89]S.K.Andersen & al. (1989): "HUGIN – A shell for building Bayesian belief universes for expert systems", *Proc. of IJCAI 89*, :1080-1085;

[BarGel94]Baral C., Gelfond M. (1994), "Logic Programming and Knowledge Representation", *Journal of Logic Programming* 19,20, :73-148;

[BDKT97]Bondarenko A., Dung P.M., Kowalski R.A.,Toni F. (1997), "An abstract , argumentation-theoretic approach to default reasoning", *Artificial Intelligence*, Vol.93, Issue 1-2, :63-101;

[BesDou04]Besnard P., Doutre S. (2004): "Checking the acceptability of a set of arguments", *Proceedings of the 10th International Workshop on Non-Monotonic Reasoning*, Canada, :59-64;

[BreDix97]Brewka, Dix, Konolige (1997) "Nonmonotonic Reasoning: an overview", *CLSI Lecture Notes 73*, Stanford, California;

[Bry86]R. Bryant (1986): "Graph-Based Algorithms for Boolean Function Manipulation", *IEEE Transactions on Computers*, C-35, 8: 677-691;

[Byl91]Bylander, D.Allemang, M.Tanner, J.R.Josephson (1991): "The Computational Complexity of Abduction",*Artificial Intelligence*, Vol.49, :25-60;

[Cal01] Calado J. & all (2001). "Soft Computing Approaches to Fault Diagnosis for Dynamic Systems", *European Journal of Control* **7**:248-286;

[Cart80]McCarthy, J.,(1980), "Circumscription - a Form of Non-Monotonic Reasoning", *Artificial Intelligence 13*, North-Holland, :27-39;

[CasFanMen03] Castellano G., Fanelli A., Mencar C. (2003): "A Fuzzy Clustering Approach for Mining Diagnostic Rules", *Proceedings of 2003 IEEE International Conference on Systems, Man and Cybernetics*, vol1, :2007-2012, Washington D.C., USA;

[Cass93]C. Cassandras (1993): "Discrete Event Systems: Modeling and Performance Analysis", Irwin Publ., Boston;

[CayDou02] Cayrol C., Doutre S., Lagasquie-Schiex M.C., Mengin J.: "Minimal defence: a refinement of the preferred semantics for argumentation frameworks", *NMR* 2002, :408-415;

[CayDubPra94]D.Cayrac, D.Dubois, H.Prade (1994): "Possibility theory in fault mode effects analyses - a satellite fault diagnosis application", *Proc. of the 3rd IEEE International Conference on Fuzzy Systems*, Orlando, FL, 1176-1181;

[Cla90]Clark P. (1990), "Nonmonotonic Reasoning, Argumentation and Machine Learning", *Technical Report TIMLG-38*, Turing Institute, Glasgow, UK;

[ClaKui98]D. Clancy, B. Kuipers (1998): "Qualitative simulation as a temporally-extended constraint satisfaction problem", *AAAI/IAAI, :240-247;*

[ConTor92]L.Console, P.Torasso (1992): "A Spectrum of Logical Definitions of Model-based Diagnosis", in W.Hamscher et al.(eds.) *Readings in Model-based Diagnosis*, Morgan Kaufman Publ., San Mateo,CA;

[deKleer86]de Kleer J.(1986), "An assumption-based truth maintenance system", *Artificial Intelligence 28*, :127-162;

[deKleer86b]deKleer J.(1986), "Back to backtracking: Controlling the ATMS", *Proceedings of the 5th National Conference on Artificial Intelligence*, :910-917;

[deKleer89]deKleer J. (1989), "A comparison of ATMS and CSP techniques", *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, Mi;

[deKleer89b]deKleer J. (1989), "Propositional Inference in CSP and ATMS techniques", SSL Paper P89-00023;

[deKleer90]deKleer J. (1990), "Exploiting locality in a TMS", *Proceedings of AAAI*-91, :264-271, Anaheim, CA.

[deKleerMackRei92]J.de Kleer, A.Mackworth, R.Reiter (1992): "Characterizing Diagnoses and Systems", *Artificial Intelligence 56*: 197-222 ;

[DerDoy80]McDermott,D., Doyle,J.(1980), "Non-Monotonic Logic I", *Artificial Intelligence 13,* North-Holland, :41-72;

[Der82]McDermott,D.(1982), "Non-Monotonic Logic II: Nonmonotonic Modal Theories", *JACM 29*, :33-57;

[Dix55]Dix J. (1955), "A Classification Theory of Semantics of Normal Logic Programs: I.Strong Properties", *Fundamenta Informaticae* XXII(3), :227-255;

[DKT97] P.M.Dung, R.A.Kowalski, F.Toni: (1997) "Argumentation-theoretic proof procedures for default reasoning", *Technical Report*, Imperial College, London, UK;

[DixJ95] Dix J. (1995) "Semantics of Logic Programs: Their Intuitions and Formal Properties", in A. Fuhrmann and Hans Rott, editors, Logic, Action and Information-Essays on Logic in Artificial Intelligence, DeGruyter :241-327;

[Doutre02]Doutre S.(2002), "Autour de la semantique preferee des systemes d'argumentation", These en informatique, Universite Toulouse III-Paul Sabatier;

[Doy79]Doyle J.(1979), "A truth maintenance system", *Artificial Intelligence* 12, :231-272;

[DubPra85]Dubois D., Prade H.(1985) : *"Theorie des possibilites: applications a la representation des connaissances en informatique"*, Barcelon NY Paris Masson;

[DudHarSto00]R.Duda , P.Hart , D.Stork (2000): "Pattern Classification", John Wiley Interscience;

[DumiBuiu00] Dumitrache I., Buiu C., (2000): *„Algoritmi Genetici – Principii Fundamentale şi Aplicaţii în Automatică",* Ed Mediamira, Cluj-Napoca;

[DumMihRos98]I. Dumitrache, I.R.Mihu, K.Rosu(1998): "Hybrid Decision Making System for Medical Diagnosis", *Proceedings of the Third International Conference on Neural Networks and Expert Systems in Medicine and Healthcare*, World Scientific Publishing;

[Dung91]Dung P.M. (1991), "Negation as hypothesis: an abductive foundation for logic programming", *Proceedings 8th International Conference on Logic Programming*, MIT Press, Paris, :3-17;

[Dun95]Dung.P.M. (1995) "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n-persons games", *Artificial Intelligence* 77:321-357,

[Dup94]D.T.Dupre (1994): "Characterizing and Mechanizing Abductive Reasoning", Ph.D. Thesis, Department of Computer Science, University of Torino;

[EshKo88]Eshghi K., Kowalski R.(1988), „Abduction through deduction", *Technical Report Department of Computing*, Imperial College, London;

[EshKo89]Eshghi K., Kowalski R.(1989), „Abduction compared with negation by failure", *Proceedings 6th International Conference on Logic Programming*, MIT Press, Lisbon, :234-255;

[Fran96]P.M.Frank (1996): "Analytical and Qualitative Model-based Fault Diagnosis ", *European Journal of Control*, 2 (1), :6-28;

[Ful95] Fuller R., (1995). *"Neural Fuzzy Systems"*, Abo Akademi's tryckeri, Abo, ESF Series A: 443;

[GebKru]J.Gebhardt, R.Kruse (1996): "POSSINFER: A software tool for possibilistic inference", in D. Dubois, H. Prade, P. Yager (Eds.) , *Fuzzy Set Methods in Information Engineering: A Guided Tour of Applications*, Wiley, NY pp. 407-418;

[GelLi88]Gelfond M., Lifschitz V.(1988), „The Stable model semantics for logic programs", *Proceedings 5th International Conference and Symposium on Logic Programming*., Washington, Seattle, :1070-1080;

[Ger95]R. Germundsson (1995):"Symbolic Systems- Theory, Computation and Applications", PhD. thesis, Linkoping University, Sweden;

[GiaRil94]J.Giarratano, G. Riley (1994): "Expert Systems: Principles & Programming", PWS Pub.Co, Boston;

[Gin96]Ginsberg M., Crawford J., Etherington D., *"Dynamic Backtracking"*, University of Oregon, *Technical Report* 8/1/1996;

[GinMcAll94]Ginsberg M., McAllester D.(1994), "GSAT and dynamic backtracking", *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning,* :226-237;

[GovMah00]Governatori G., Maher M.J., Antoniou G., Billington D. (2000), "Argumentation Semantics for Defeasible Logics", in Mizoguchi, R. and Slaney J., editors, *PRICAI 2000: Topics in Artificial Intelligence*, Vol. 1886 of LNAI, Berlin Springer Verlag, :27-37;

[Gui77]S.Guiaşu (1977): *"Information Theory with Applications"*, McGraw-Hill, New York;

[Hay98]S.Haykin (1998): *"Neural Networks: A Comprehensive Approach"*, Prentice Hall;

[Heck91]A. Heckermann (1991): "Probabilistic Similarity Networks: PATHFINDER", MIT Press, Cambridge MA;

[HuaDar96]C. Huang, A.Darwiche (1996): "Inference in Belief Networks: A Procedural Guide", *International Journal of Approximate Reasoning*, vol 15, nr 3; :225-263;

[HuaHen96]K. Huang, M.Henrion (1996): "Efficient Search-Based Inference for Noisy-OR Belief Networks: TopEpsilon", *12th Conference on Uncertainty in Artificial Intelligence, Portland,* OR, 325-331;

[Karr91] Karr C., (1991): "Design of an adaptive fuzzy logic controller using a genetic algorithm", in *Proceedings of th 4th International Conference on Genetic Algorithms,* Belew and Brooker eds.;

[KunSte99] Kuncheva L., Steimann F. (1999): „Fuzzy Diagnosis", *Artificial Intelligence in Medicine,* 16(2):121-128;

[Ise97] Isermann R., (1997). "Supervision, Fault Detection and Fault Diagnosis Methods-an Introduction", *Control Engineering Practice*, vol. **5**, no. 5 : 639-652;

[JafLas87]Jaffar J., Lassez J-L.,(1987) "Constraint Logic Programming", Proc. 14th ACM Symposium on Principles of Programming Languages, Munich;

[Jan92]R.Jang (1992): "Neuro-Fuzzy Modeling: Architecture, Analysis and Applications", PhD. thesis , University of California at Berkeley;

[KaMan90]Kakas A., Mancarella P.(1990), „Generalized Stable Models: a Semantics for Abduction", *Proceedings 9th European Conference on AI, Stockholm*, :385-391;

[KaManDun94]Kakas A., Mancarella P., Dung P.M.(1994), „The acceptability semantics for logic programs", *Proceedings 11th International Conference on Logic Programminmg*, MIT Press, Santa Margherita Ligure, Italy, :504-519;

[KKoT98]Kakas A., Kowalski R., Toni F., (1998), "The role of abduction in logic programming", *Handbook of logic in Artificial Intelligence and Logic Programming 5*, Oxford University Press, :235-324;

[Kon88]Konolige, K.,(1988), "Defeasible argumentation in reasoning about events" , in Z.W.Ras and L.Saitta, editors, *Methods for Intelligent Systems,*3, Elsevier NY, pp 380-390;

[Kon92]K.Konolige (1992): "Abductive Theories in Artificial Intelligence" in Brewka, G. (ed) *Principles of Knowledge Representation*, CSLI Publications, Stanford University;

[KoSedZac99] Koscielny, J. M., Sedziak, D., and Zackroczymsky, K. (1999). "Fuzzy-logic fault isolation in large-scale systems", *International Journal of Applied Mathematics and Computer Science* **9** (3): 637-652.

[Kui86]B. Kuipers (1986): "Qualitative Simulation", *Artificial Intelligence 29*, pp. 289-338, Elsevier Science Publishers, North-Holland;

[Lar99]M. Larsson (1999): "Behavioral and Structural Model-Based Approaches to Discrete Diagnosis", PhD. thesis, Linkoping University, Sweden;

[LeeTam]Lee J., Tam V., (1995): "A framework for integrating artificial neural networks and logic programming", *International Journal of Artificial Intelligence Tools* 4, :3-32;

[Lon01]W. Long (2001): "Medical Informatics: reasoning methods", *Artificial Intelligence in Medicine*, 2371-2387;

[LonNai92]W. Long, S. Naimi (1992): "Development of a knowledge base for diagnostic reasoning in cardiology", *Computers in Biomedical Research 25* (1992), 292-311;

[LunSchro01]J.Lunze, J. Schroder (2001): "Fault Diagnosis of Stochastic Automata Networks", *Bridge Workshop '01*;

[McAll80]McAllester D.(1980), "An outlook on truth maintenance", *Artificial Intelligence Laboratory*, AIM-551, MIT, Cambridge, MA;

[McIlr98]S.McIlraith (1998): "Logic-based Abductive Inference", Knowledge Systems Laboratory, *Technical Report KSL-98-19*;

[Mich96]Michalewicz Z. (1996), "Genetic Algorithms + Data Structures= Evolutionary Programs", Springer, Berlin;

[MichFog04]Michalewicz Z., Fogel D.(2004), "How to solve it: modern heuristics", Springer, Berlin;

[MigI01]Miguel I., *"Dynamic Flexible Constraint Satisfaction and its Application to AI Planning",* PhD Thesis, University of Edinburgh, 2001;

[Mit00]S. Mitra (2000): "Neuro-Fuzzy Rule Generation: Survey in Soft Computing Framework", *IEEE Transactions on Neural Networks*, vol. 11-3;

[Mun03]Munteanu, S.(2003), "Medical diagnosis modeled in terms of fuzzy decision", *Proc. Of 14th International Conference on Control Systems and Computer Science*, Bucharest;

[Mun04]Munteanu S. (2004), "Argumentation-based Conflict Resolution for Discriminating Diagnostic Hypotheses", *The 12th International Symposium on Modeling, Simulation and System's Identification,* Galati, :218-225;

[Mun05]Munteanu, S.(2005), "A Hybrid Model for Diagnosing Multiple Disorders", *International Journal of Hybrid Intelligent Systems*, Vol 2. No.1 IOSPress, :35-55;

[MunDum06a]Munteanu S., Dumitrache I. (2006) "A DCSP-based nonmonotonic framework for medical diagnosis", trimisă la *Artificial Intelligence in Medicine;*

[MunDum06b]Munteanu S., Dumitrache I. (2006) "The Intelligence of a Hybrid System: DiaMed", *Technical Report*, Polytechnica University, Bucharest- acceptată în *Control Engineering and Applied Informatics*;

[MunDum06c]Munteanu S., Dumitrache I. (2006) "Applying DiaMed Hybrid Algorithm to the Diagnosis of Dynamic Systems", *Technical Report*, Polytechnica University, Bucharest –acceptată în *The Scientific Bulletin of Politehnica University*, Bucharest;

[Mun06] (2006) "The selection of diagnostic hypotheses through fuzzy decision in the DiaMed hybrid system", trimisă la "Fuzzy Systems and Artificial Intelligence";

[NeaPal01]D. Neagu, V. Palade (2001): *"Sisteme Inteligente Hibride"*, Ed. MatrixRom, Bucureşti;

[Nyb99]M.Nyberg (1999): "Model Based Fault Diagnosis", PhD. thesis, Linkoping University, Sweden;

[PalNea01]Palade V., Neagu C.(2001): *"Sisteme inteligente hibride"*, MatrixRom, Bucureşti;

[Pan99]A.Panati (1999): "Non Binary CSP and Heuristics for Modeling and Diagnosing Dynamic Systems", *Proc. 6th Congress of the Italian Association for Artificial Intelligence*;

[PanDup00]A.Panati, D.T. Dupre (2000)a: "State-based vs simulation based diagnosis of dynamic systems" , *Proc. ECAI*;

[PanDupFor00]A. Panati, D.T. Dupre, M.Formagnana (2000)b: "Causal Simulation and Diagnosis of Dynamic Systems", *Proc. DX, 11th International Workshop on Principles of Diagnosis*;

[Par94]Paredis J.(1994), "Co-evolutionary constraint satisfaction", *Proceedings of the 3rd Conference on Parallel Problem Solving from Nature",* : 46-55;

[Patt94]R.Patton (1994): "Robust Model Based Fault Diagnosis: the State of the Art", *IFAC SAFEPROCESS '94*, Helsinki, Finland;

[Patt97]R.Patton (1997): "Fault Tolerant Control: the 1997 situation (survey)", *Proc. of SAFEPROCESS '97*, pp 1033-1055, IFAC;

[PattChe97]R.J. Patton , Chen J. (1997) "Observer-based Fault Detection and Isolation: Robustness and Applications", *Control Eng. Practice*, vol. 5, no.5, pp 671-682;

[Pea01a]J.Pearl (2001)a: "Causal Inference in the Health Sciences : A Conceptual Introduction", *TR R-282*, Cognitive Systems Lab. UCLA;

[Pea01b]J. Pearl (2001)b: "Bayesianism and causality or why I am only a half bayesian",in D. Corfield, J. Williamson (eds.), *Foundations of Bayesianism*, Kluwer Applied Logic Series;

[Ped04]W.Pedrycz (2004): “Logic-Oriented Fuzzy Neural Networks”, *IJHIS* vol 1.(1):3-11;

[PerApAl93]Pereira, L.M., Aparicio, J.N., Alferes, J.J.(1993), "Non-monotonic Reasoning with Logic Programming", *Journal of Logic Programming*, Special Issue on Nonmonotonic Reasoning 17 (2,3&4);

[Po88]D.Poole (1988): “Representing Knowledge for Logic-based Diagnosis” *Proc. International Conference on Fifth Generation Computer Systems*, Tokyo, :1282-1290;

[Poll92]J.Pollock (1992) : “How to Reason Defeasibly”, *Artificial Intelligence* 57, pp1-42;

[Poll02]Pollock, J. (2002), "Defeasible Reasoning with Variable Degrees of Justification", *Artificial Intelligence* 133, :233-282;

[PopIon99]Popescu E., Ionescu R., (1999) “Rheumatology”, Technical Press, Bucharest;

[PraDag96]M.Pradhan, P. Dagum (1996): “Optimal Monte Carlo Estimation of Belief Network Inference”, 12th Conference on Uncertainty in Artificial Intelligence, Portland, OR, :446-453;

[PraVre]H.Prakken,G.Vreeswijk:"Logics for Defeasible Argumentation" (2002), Gabbay D., Guenther F., eds. *“Handbook of philosophical logic”*, vol. 4, Kluwer Academic Publishers, :218-319;

[Rei80]Reiter, R. (1980), "A Logic for Default Reasoning", *Artificial Intelligence 13*, North-Holland, :81-132;

[ReiKle87]Reiter R., de Kleer J. (1987): “Foundations of Assumption-based TMS”, *National Conference on AI*, Seattle, WA, :183-189;

[Res88]Restian A.(1988), “Diagnosticul medical”, Ed. Dacia, Cluj;

[Rim83]Rimniceanu R.(1983), “Aspecte si probleme ale medicinii contemporane”, Ed. Medicala, Bucuresti;

[RusNor02]Russell S., Norvig P.(2002), “Artificial Intelligence: A Modern Approach”, Prentice Hall;

[SamSen96]M. Sampath, R. Sengupta (1996): “Failure Diagnosis using Discrete Event Models”, IEEE Transactions on Control Systems Technology 4 (2) :105-124;

[San99]Sanchez M.M. (1999), “A Neurosymbolic Approach to the Classification of Scarce and Complex Data”, PhD Thesis, University of Sussex;

[SatIwa91]Satoh K., Iwayama N. (1991), "Computing Abduction by using the TMS", *Proceedings of 8th International Conference on Logic Programming*, Paris, MIT Press.

[Sha00]Shankar R.D & al.(2000), "Implementing Clinical Practice Guidelines while Taking Account of Evidence", *AMIA Annual Symposium*, Los Angeles, CA, :303-304;

[ShaMus99]Shankar R.D, Musen M.(1999) *,"*Justification of Automated Decision-Making: Medical Explanations as Medical Arguments*"*, *Proc AMIA Symposium*, Washington DC, :395-399*;*

[Sou83]G.Soula, B.Vialettes, J.L.San Marco (1983): "PROTIS, a Fuzzy Deduction-Rule System: Application to the Treatment of Diabetes", in MEDINFO 83, Eds. Van Bemmel, M.G. Ball, NY, North Holland, :533-536;

[SouKay02]Sousa, J., Kaymak, U. (2002) :*"Fuzzy Decision Making in Modeling and Control",* World Scientific Pub Co;

[Sta87]State L.(1987): "Metode statistice de recunoaştere a formelor", Tipografia Universității Bucureşti;

[StaSus77]Stallman R., Sussman G., (1977) "Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis", Artificial Intelligence, 9(2):135-196;

[TaSu85] Takagi T., Sugeno M. (1985) "Fuzzy Identification of Systems and Its Applications to Modeling and Control", *IEEE Transactions on Systems, Man and Cibernetics, vol SMC-15, no.1,* January-February 1985;

[TeoCheGal89] H.N.Teodorescu, M.Chelaru, D.Galea & al.(1989) *"Fuzzy Systems and Applications"*, I.P.Iaşi;

[TeoCheKan01] H.N.Teodorescu, M.Chelaru, A. Kandel & al. (2001) „Fuzzy Methods in tremor assessment, prediction and rehabilitation", *Artificial Intelligence in Medicine*, 21(1-3), :107-130;

[TorCon89]P.Torasso, L.Console (1989): "A multilevel architecture for diagnostic problem-solving", *Computational Intelligence*,1 :101-112.

[Tou58]Toulmin, S.E. (1958), *"The Uses of Argument",* University Press, Cambridge, UK;

[VerSchi]Verfaillie G., Schiex T. (1994), "Dynamic Backtracking for Dynamic Constraint Satisfaction Problems", *Proceedings of the ECAI '94 Workshop on Constraint Satisfaction Issues Raised by Practical Applications*, Amsterdam, the Netherlands;

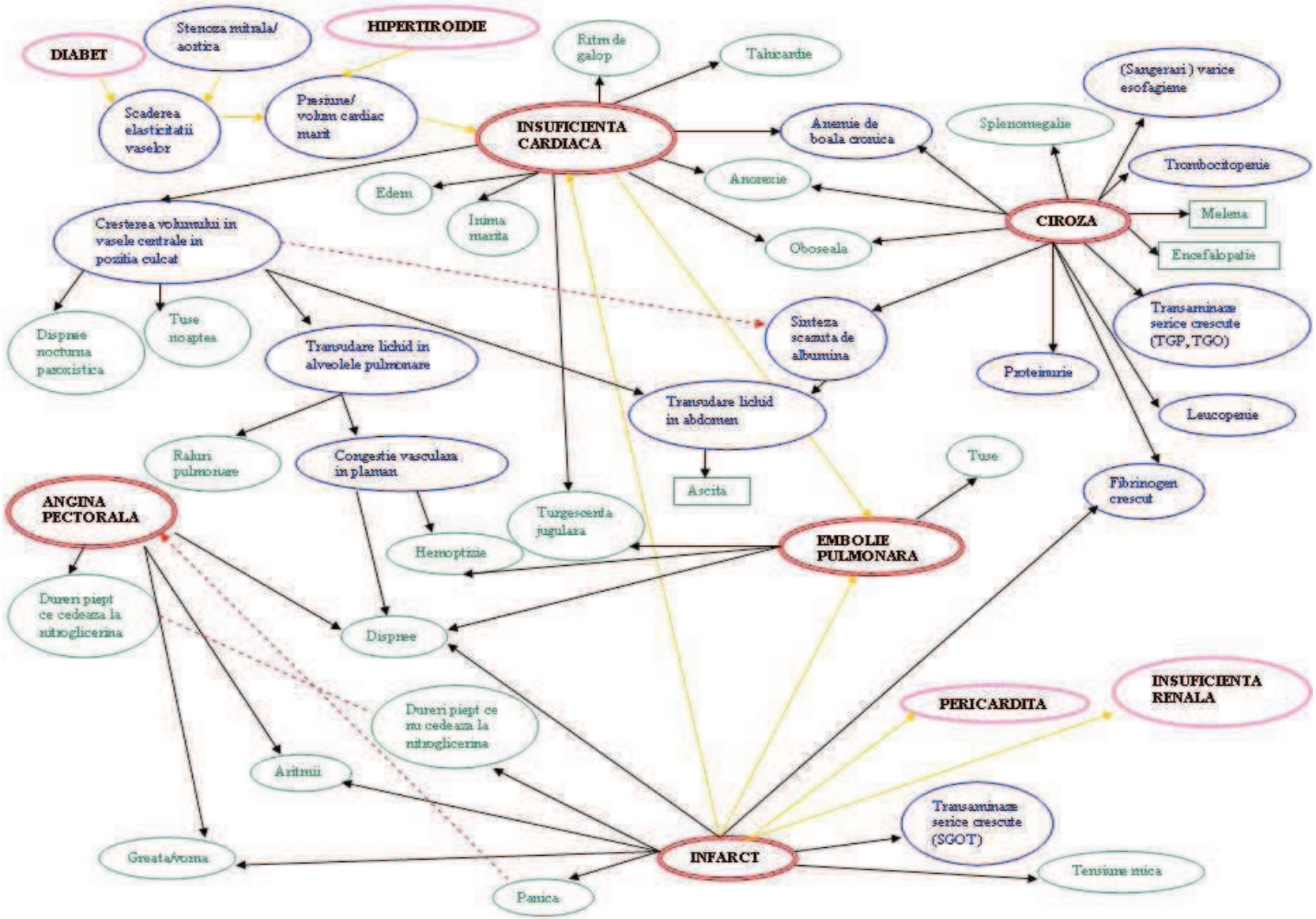
[Vre92]Vreeswijk, G. (1992), "Reasoning with Defeasible Arguments:Examples and Applications", *Logics in Artificial Intelligence,* :189-211;

[Wal75]Waltz D. (1975), "Understanding line drawings of scenes with shadows", in *"The Psychology of Computer Vision"*, :19-91. McGrawHill;

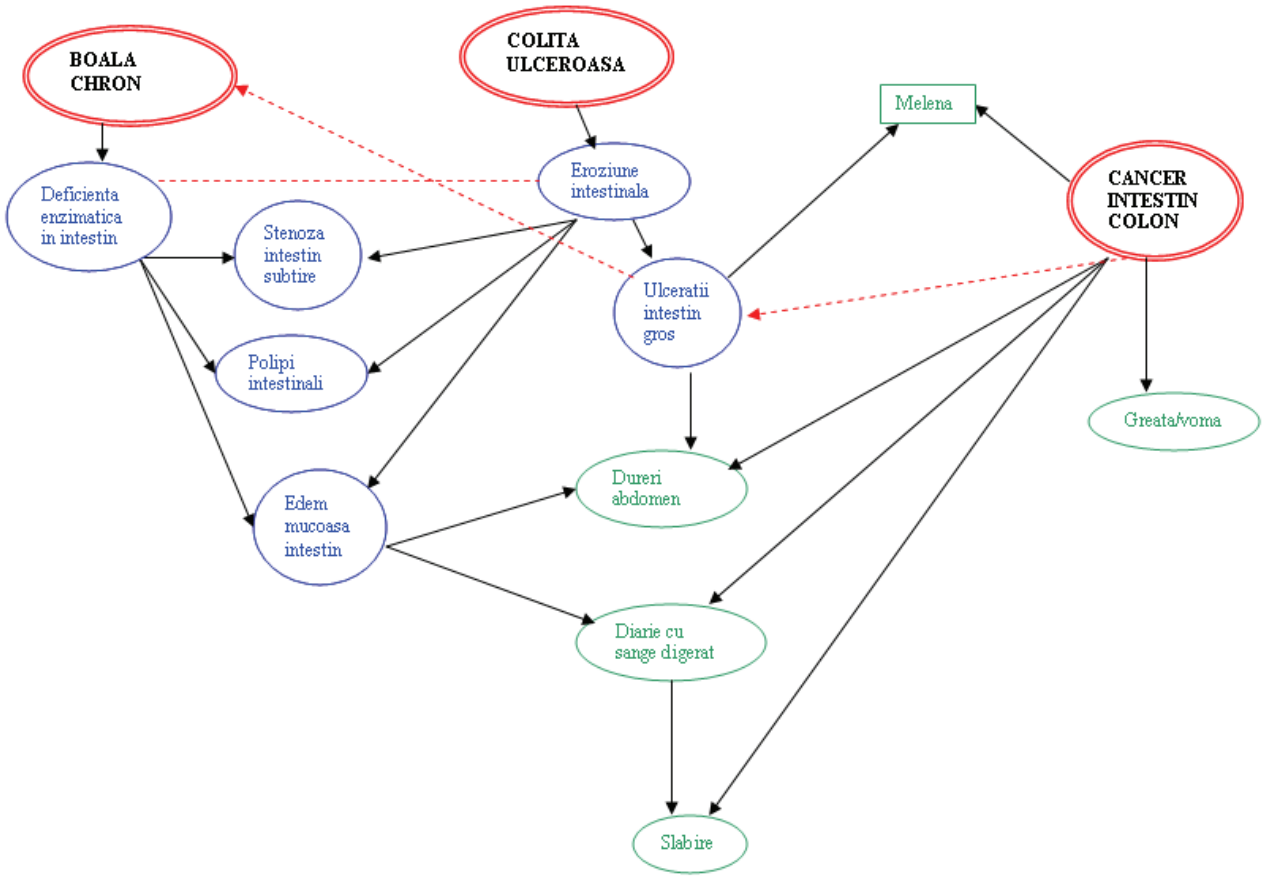
[Weis74] Weiss S. (1974), *"A System for Model-Based Computer Aided Diagnosis and Therapy"*, PhD Thesis, Department of Computer Science, Rutgers University, CBM-TR-27;

[Zad73] Zadeh L., (1973) "Outline of a new approach to the analysis of complex systems and decision processes", *IEEE Transactions on Systems, Man and Cibernetics, 3, :28-44*.

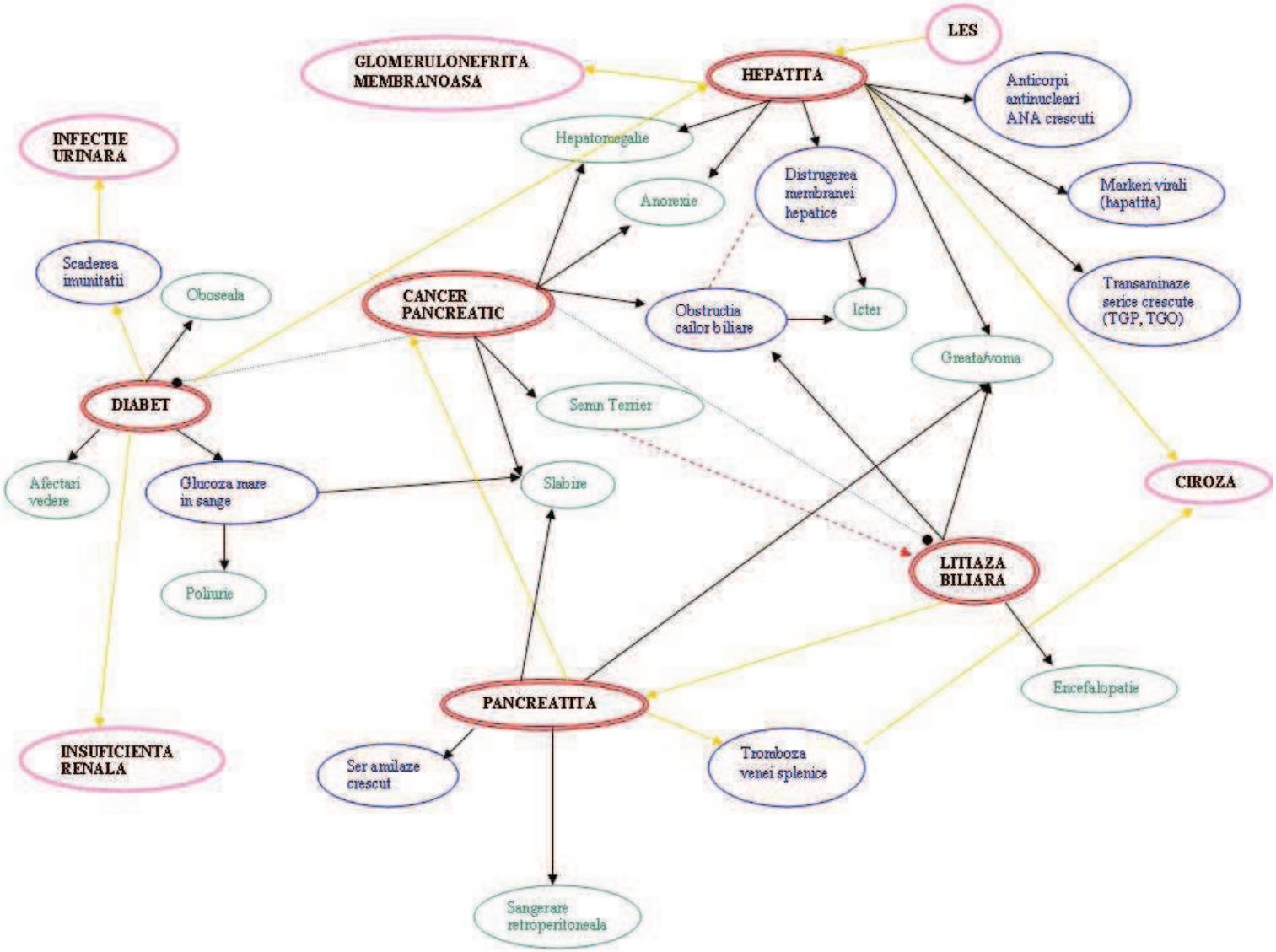
Anexa1. Modelul medical


DIABET
Stenoza mitrala/ aortica
HIPERTIROIDIE
Scaderea elasticitatii vaselor
Presiune/ volum cardiac marit
Ritm de galop
Tahicardie
INSUFICIENTA CARDIACA
Anemie de boala cronica
Splenomegalie
(Sangerari) varice esofagiene
Trombocitopenie
Anorexie
Edem
Inima marita
Cresterea volumului in vasele centrale in pozitia culcat
Oboseala
CIROZA
Melena
Encefalopatie
Dispnee nocturna paroxistica
Tuse noaptea
Transudare lichid in alveolele pulmonare
Sinteza scazuta de albumina
Transaminaze serice crescute (TGP, TGO)
Proteinurie
Transudare lichid in abdomen
Leucopenie
Raluri pulmonare
Congestie vasculara in plaman
Ascita
Tuse
Fibrinogen crescut
ANGINA PECTORALA
Turgescenta jugulara
Hemoptizie
EMBOLIE PULMONARA
Dureri piept ce cedeaza la nitroglicerina
Dispnee
Dureri piept ce nu cedeaza la nitroglicerina
PERICARDITA
INSUFICIENTA RENALA
Aritmii
Transaminaze serice crescute (SGOT)
Greata/voma
INFARCT
Tensiune mica
Panica

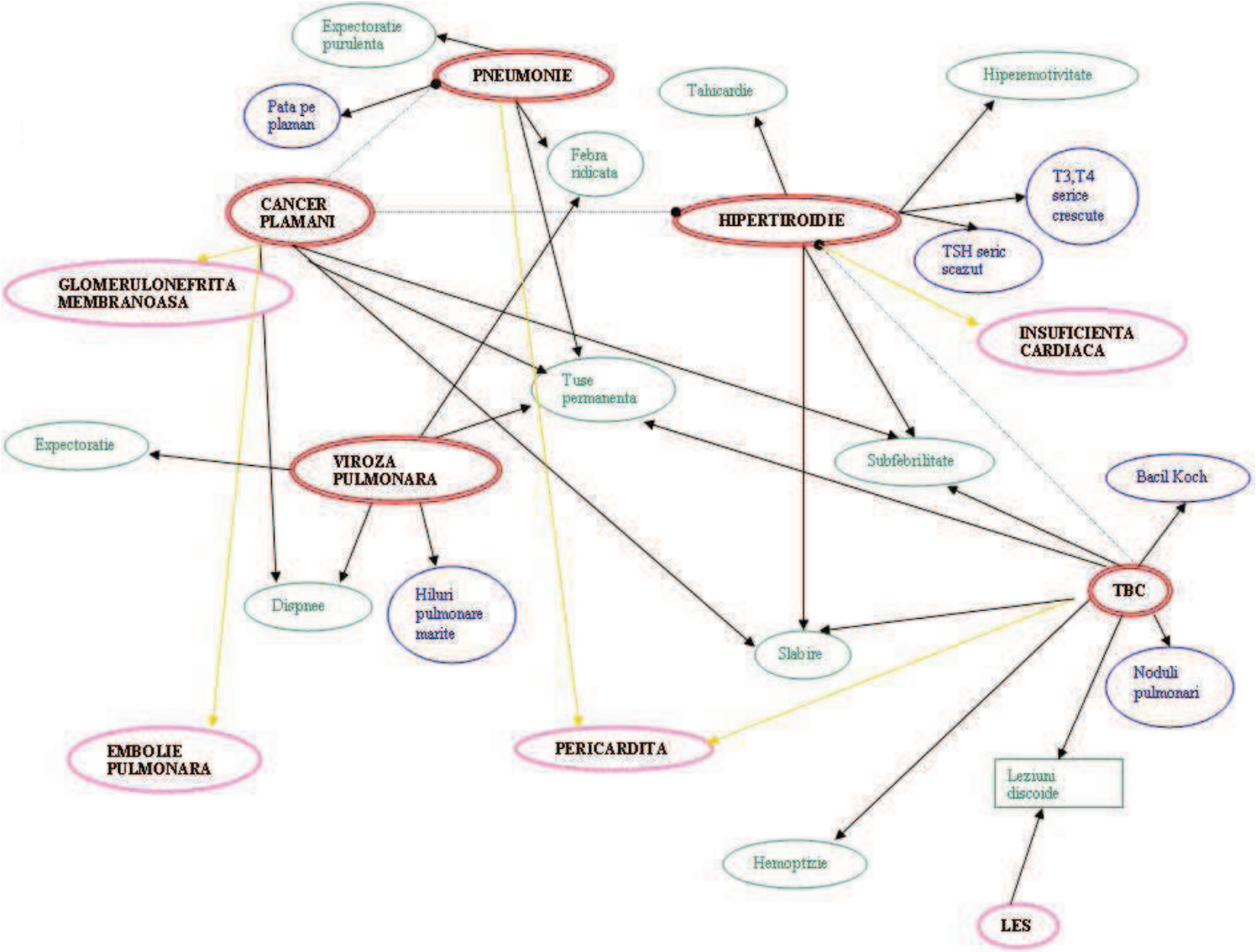
PLANŞA I


BOALA CHRON
COLITA ULCEROASA
Melena
CANCER INTESTIN COLON
Deficienta enzimatica in intestin
Eroziune intestinala
Stenoza intestin subtire
Ulceratii intestin gros
Polipi intestinali
Greata/voma
Dureri abdomen
Edem mucoasa intestin
Diarie cu sange digerat
Slabire

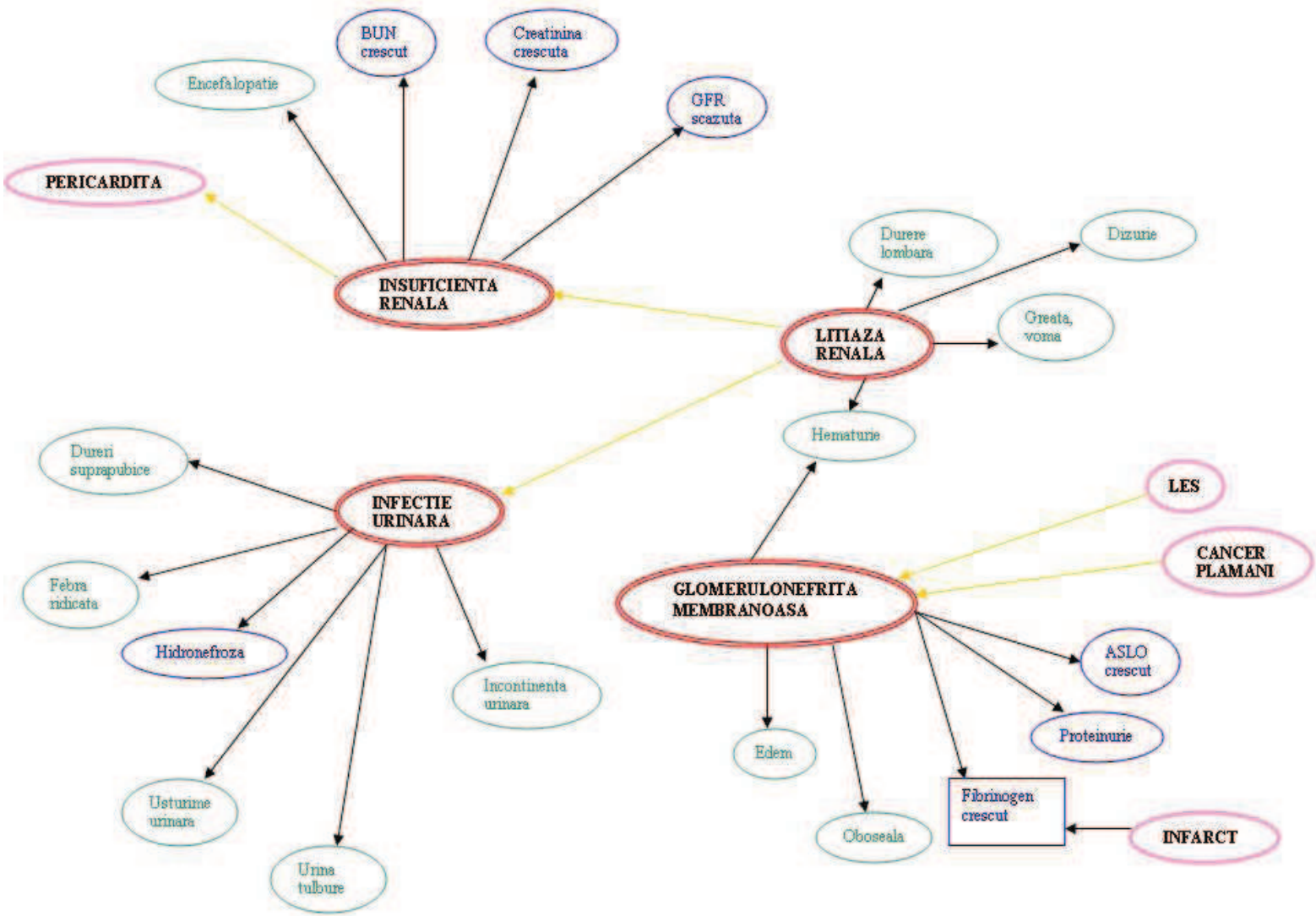
PLANŞA II


LES
GLOMERULONEFRITA MEMBRANOASA
HEPATITA
Anticorpi antinucleari ANA crescuti
INFECTIE URINARA
Hepatomegalie
Distrugerea membranei hepatice
Markeri virali (hapatita)
Anorexie
Scaderea imunitatii
Oboseala
CANCER PANCREATIC
Transaminaze serice crescute (TGP, TGO)
Obstructia cailor biliare
Icter
Greata/voma
DIABET
Semn Terrier
Afectari vedere
Glucoza mare in sange
Slabire
CIROZA
LITIAZA BILIARA
Poliurie
Encefalopatie
PANCREATITA
INSUFICIENTA RENALA
Ser amilaze crescut
Tromboza venei splenice
Sangerare retroperitoneala

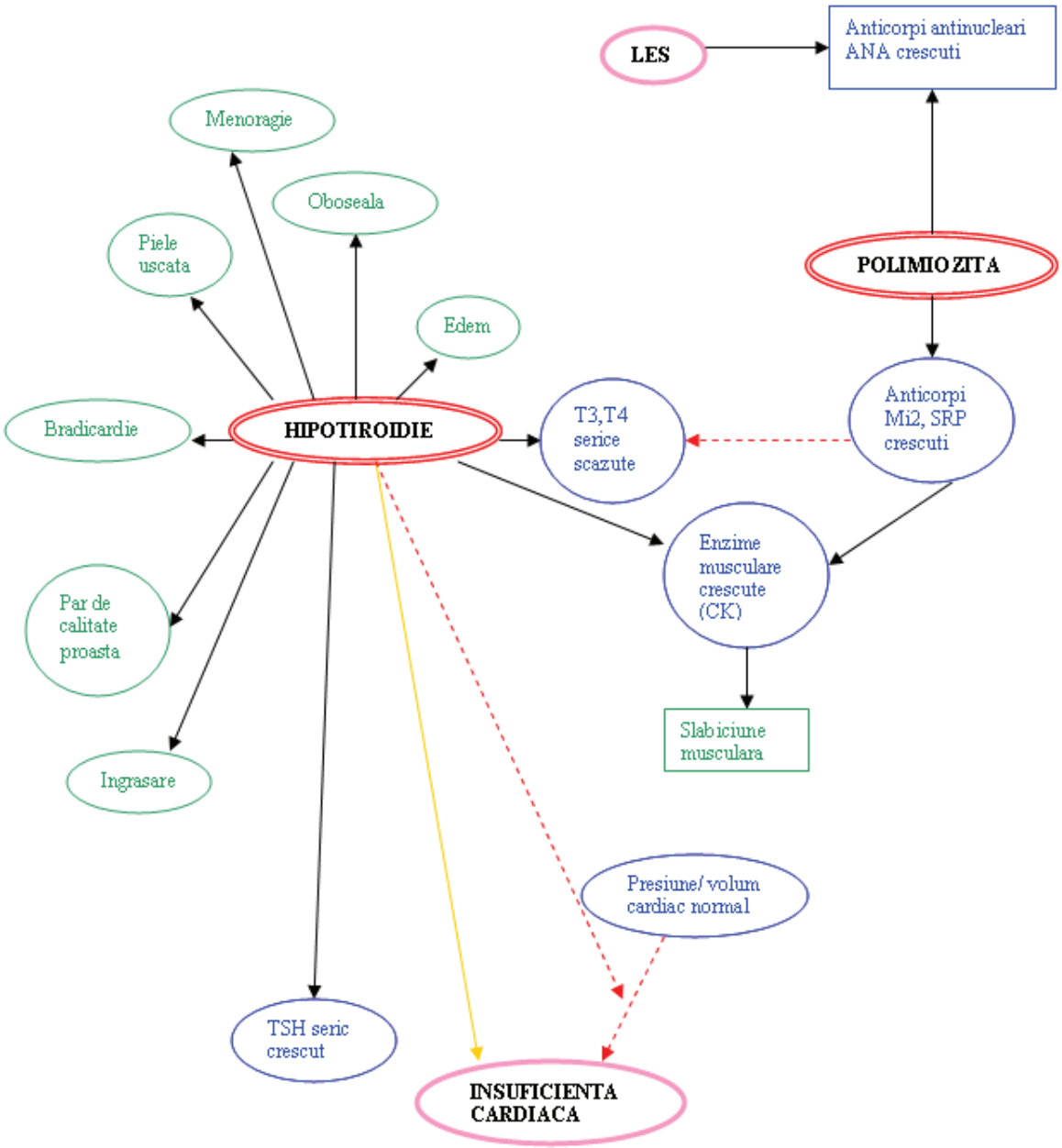
PLANŞA III


Expectoratie purulenta
PNEUMONIE
Pata pe plaman
Febra ridicata
CANCER PLAMANI
GLOMERULONEFRITA MEMBRANOASA
Tahicardie
Hiperemotivitate
T3,T4 serice crescute
HIPERTIROIDIE
TSH seric scazut
INSUFICIENTA CARDIACA
Tuse permanenta
Expectoratie
VIROZA PULMONARA
Subfebrilitate
Bacil Koch
Dispnee
Hiluri pulmonare marite
TBC
Slabire
Noduli pulmonari
EMBOLIE PULMONARA
PERICARDITA
Leziuni discoide
Hemoptizie
LES

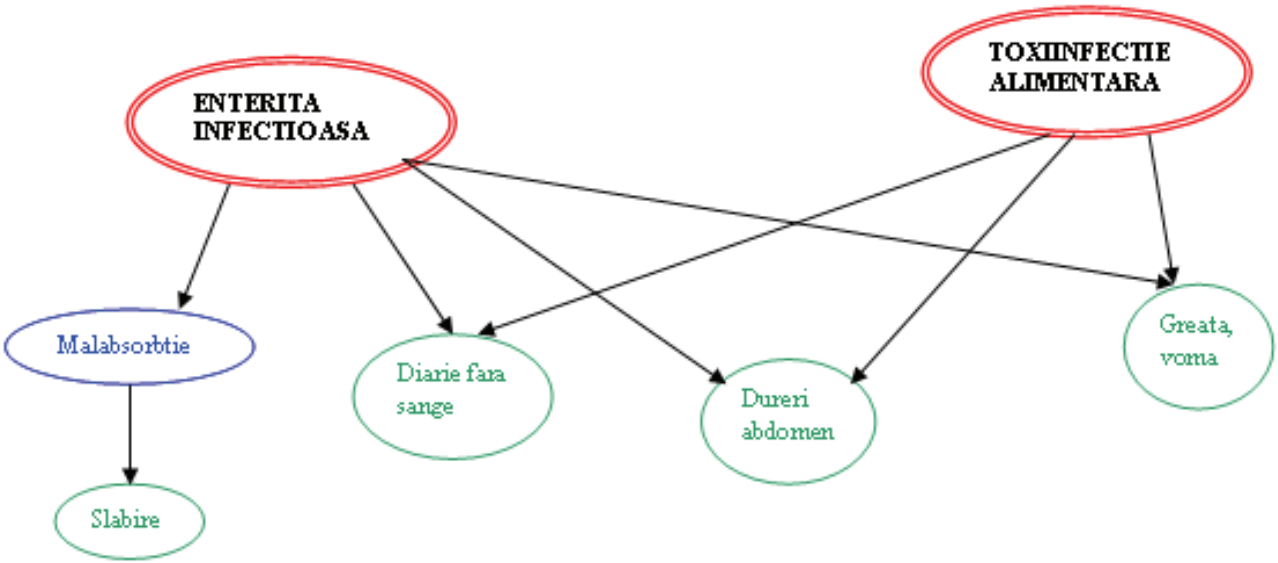
PLANŞA IV


BUN crescut
Creatinina crescuta
Encefalopatie
GFR scazuta
PERICARDITA
INSUFICIENTA RENALA
Durere lombara
Dizurie
LITIAZA RENALA
Greata, voma
Hematurie
Dureri suprapubice
INFECTIE URINARA
Febra ridicata
Hidronefroza
Incontinenta urinara
Usturime urinara
Urina tulbure
LES
CANCER PLAMANI
GLOMERULONEFRITA MEMBRANOASA
ASLO crescut
Proteinurie
Edem
Oboseala
Fibrinogen crescut
INFARCT

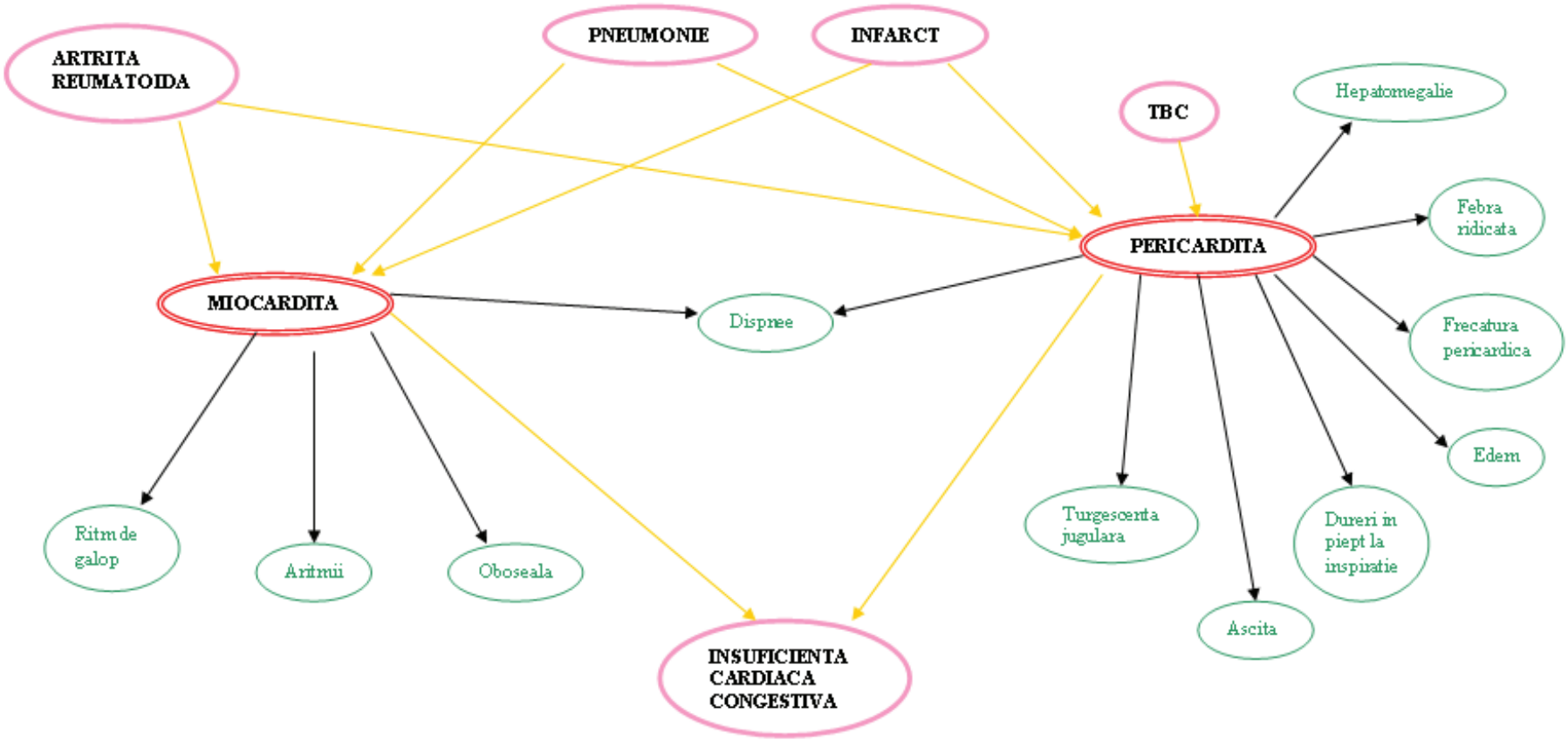
PLANŞA V


LES
Anticorpi antinucleari ANA crescuti
Menoragie
Oboseala
Piele uscata
Edem
POLIMIOZITA
Anticorpi Mi2, SRP crescuti
Bradicardie
HIPOTIROIDIE
T3,T4 serice scazute
Enzime musculare crescute (CK)
Par de calitate proasta
Slabiciune musculara
Ingrasare
Presiune/ volum cardiac normal
TSH seric crescut
INSUFICIENTA CARDIACA

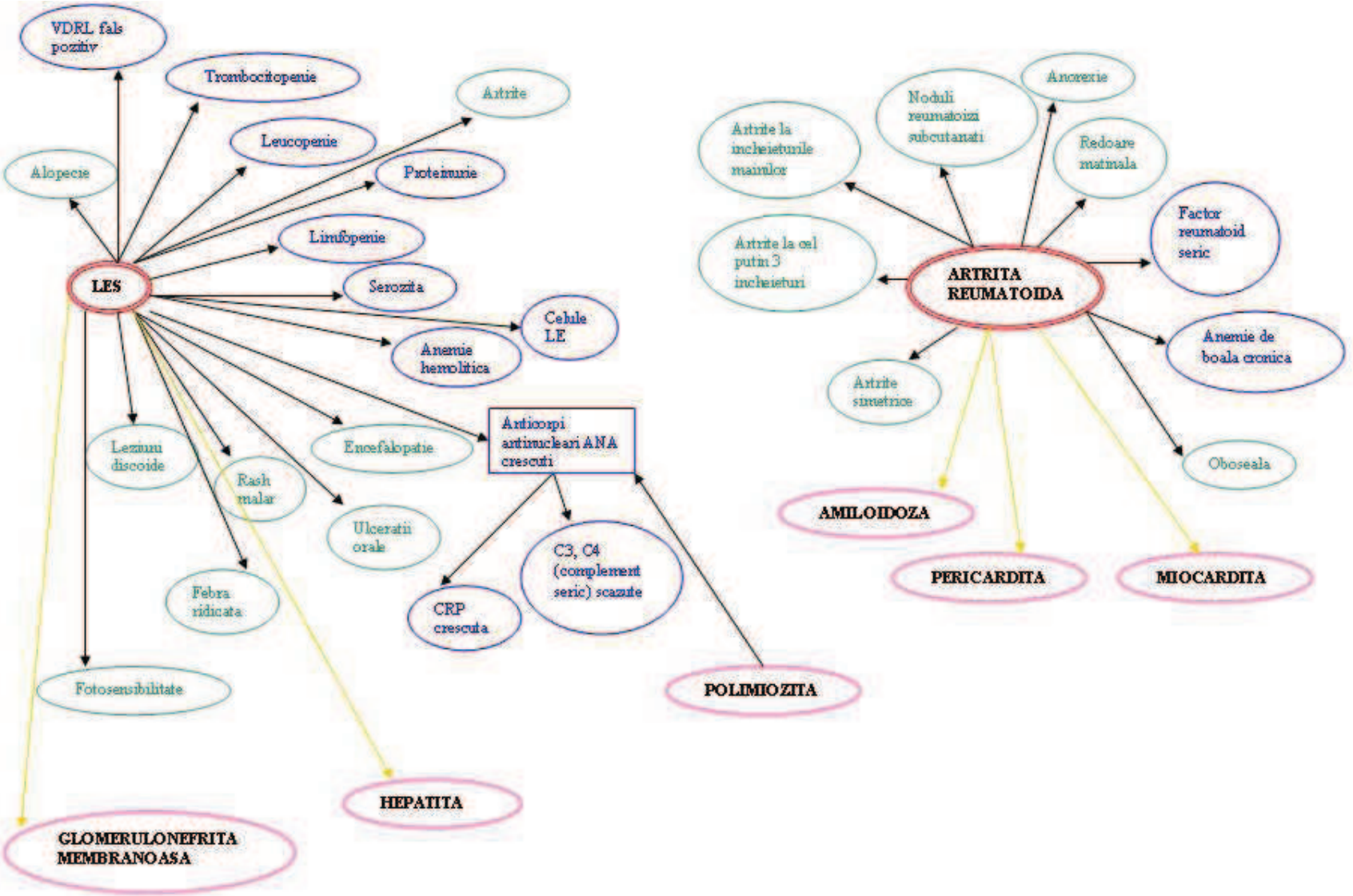
PLANŞA VI


ENTERITA INFECTIOASA
TOXIINFECTIE ALIMENTARA
Malabsorbtie
Diarie fara sange
Dureri abdomen
Greata, voma
Slabire

PLANŞA VII


ARTRITA REUMATOIDA
PNEUMONIE
INFARCT
TBC
Hepatomegalie
Febra ridicata
PERICARDITA
MIOCARDITA
Dispnee
Frecatura pericardica
Edem
Ritm de galop
Aritmii
Oboseala
Turgescenta jugulara
Dureri in piept la inspiratie
Ascita
INSUFICIENTA CARDIACA CONGESTIVA

PLANŞA VIII


VDRL fals pozitiv
Trombocitopenie
Artrite
Leucopenie
Alopecie
Proteinurie
Limfopenie
LES
Serozita
Celule LE
Anemie hemolitica
Anticorpi antinucleari ANA crescuti
Lezium discoide
Encefalopatie
Rash malar
Ulceratii orale
C3, C4 (complement seric) scazute
Febra ridicata
CRP crescuta
Fotosensibilitate
POLIMIOZITA
HEPATITA
GLOMERULONEFRITA MEMBRANOASA
Anorexie
Noduli reumatoizi subcutanati
Artrite la incheieturile mainilor
Redoare matinala
Factor reumatoid seric
Artrite la cel putin 3 incheieturi
ARTRITA REUMATOIDA
Anemie de boala cronica
Artrite simetrice
Oboseala
AMILOIDOZA
PERICARDITA
MIOCARDITA

PLANŞA IX

Anexa 2. Lista completă de noduri şi codificările lor în program

	NUME_NOD	DEN_MED
	boli	
1	BI1	ANGINA_PECTORALA
2	BREUM1	ARTRITA_REUMATOIDA
3	BGI1	BOALA_CHRON
4	BGI2	CANCER_COLON/INTESTIN
5	BGI3	CANCER_PANCREAS
6	BP1	CANCER_PLAMANI
7	BH1	CIROZA
8	BGI4	COLITA_ULCEROASA
9	BE1	DIABET
10	BP2	EMBOLIE_PULMONARA
11	BGI5	ENTERITA_INFECTIOASA
12	BREN1	GLOMERULONEFRITA_MEMBRANOASA
13	BH2	HEPATITA
14	BE2	HIPERTIROIDIE
15	BE3	HIPOTIROIDIE
16	BI2	INFARCT_MIOCARDIC
17	BREN2	INFECTIE_URINARA
18	BI3	INSUFICIENTA_CARDIACA_CONGESTIVA
19	BREN3	INSUFICIENTA_RENALA
20	BREUM2	LES
21	BGI6	LITIAZA_BILIARA
22	BREN4	LITIAZA_RENALA
23	BI4	MIOCARDITA
24	BGI7	PANCREATITA
25	BI5	PERICARDITA
26	BP3	PNEUMONIE
27	BREUM3	POLIMIOZITA
28	BP4	TBC
29	BGI8	TOXIINFECTIE_ALIMENTARA
30	BP5	VIROZA_PULMONARA
	simpt superficiale	
1	SS1	AFECTARI_VEDERE
2	SS2	ALOPECIE
3	SS3	ANOREXIE
4	SS4	ARITMII
5	SS5	ARTRITE
6	SS6	ARTRITE_LA_CEL_PUTIN_3_INCHEIETURI
7	SS7	ARTRITE_LA_INCHEIETURILE_MAINILOR
8	SS8	ARTRITE_SIMETRICE
9	SS9	ASCITA
10	SS10	BRADICARDIE
11	SS11	DIARIE_CU_SANGE_DIGERAT
12	SS12	DIARIE_FARA_SANGE
13	SS13	DISPNEE
14	SS14	DISPNEE_NOCTURNA_PAROXISTICA
15	SS15	DIZURIE

16	SS16	DURERI_ABDOMEN
17	SS17	DURERE_LOMBARA
18	SS18	DURERI_IN_PIEPT_LA_INSPIRATIE
19	SS19	DURERI_PIEPT_CE_CEDEAZA_LA_NITROGLICERINA
20	SS20	DURERI_PIEPT_CE_NU_CEDEAZA_LA_NITROGLICERINA
21	SS21	DURERI_SUPRAPUBICE
22	SS22	EDEM
23	SS23	ENCEFALOPATIE
24	SS24	EXPECTORATIE
25	SS25	EXPECTORATIE_PURULENTA
26	SS26	FEBRA_RIDICATA
27	SS27	FOTOSENSIBILITATE
28	SS28	FRECATURA_PERICARDICA
29	SS29	GREATA/VOMA
30	SS30	HEMATURIE
31	SS31	HEMOPTIZIE
32	SS32	HEPATOMEGALIE
33	SS33	HIPEREMOTIVITATE
34	SS34	ICTER
35	SS35	INIMA_MARITA
36	SS36	INCONTINENTA_URINARA
37	SS37	INGRASARE
38	SS38	LEZIUNI_DISCOIDE
39	SS39	MELENA
40	SS40	MENORAGIE
41	SS41	NODULI_REUMATOIZI_SUBCUTANATI
42	SS42	OBOSEALA
43	SS43	PANICA
44	SS44	PAR_DE_CALITATE_PROASTA
45	SS45	PIELE_USCATA
46	SS46	POLIURIE
47	SS47	RALURI_PULMONARE
48	SS48	RASH_MALAR
49	SS49	REDOARE_MATINALA
50	SS50	RITM_DE_GALOP
51	SS51	SANGERARE_RETROPERITONEALA
52	SS52	SEMN_TERRIER
53	SS53	SLABICIUNE_MUSCULARA
54	SS54	SLABIRE
55	SS55	SPLENOMEGALIE
56	SS56	SUBFEBRILITATE
57	SS57	TAHICARDIE
58	SS58	TENSIUNE_MICA
59	SS59	TURGESCENTA_JUGULARA
60	SS60	TUSE_NOAPTEA
61	SS61	TUSE_PERMANENTA
62	SS62	ULCERATII_ORALE
63	SS63	USTURIME_URINARA
64	SS64	URINA_TULBURE

Simptome intermediare

1 SI1 ANEMIE_DE_BOALA_CRONICA
2 SI2 ANEMIE_HEMOLITICA
3 SI3 ANTICORPI_ANTINUCLEARI(ANA)_CRESCUTI
4 SI4 ANTICORPI_Mi-2,SRP_CRESCUTI
5 SI5 ASLO_CRESCUT
6 SI6 BUN_CRESCUT
7 SI7 CELULE_LE
8 SI8 C3,C4_SCAZUTE(COMPLEMENT_SERIC)
9 SI9 CONGESTIE_VASCULARA_IN_PLAMAN
10 SI10 CREATININA_CRESCUTA
11 SI11 CRESTEREA_VOLUMULUI_IN_VASELE_CENTRALE_IN_POZITIA_CULCAT
12 SI12 CRP_CRESCUTA
13 SI13 DEFICIENTA_ENZIMATICA_IN_INTESTIN
14 SI14 DISTRUGEREA_MEMBRANEI_HEPATICE
15 SI15 EDEM_MUCOASA_INTESTIN
16 SI16 ENZIME_MUSCULARE_CRESCUTE(CK,ALDOLAZA,MIOGLOBINA_SERICA)
17 SI17 FACTOR_REUMATOID_SERIC
18 SI18 FIBRINOGEN_CRESCUT
19 SI19 GFR_SCAZUTA
20 SI20 GLUCOZA_MARE_IN_SANGE
21 SI21 HIDRONEFROZA
22 SI22 HILURI_PULMONARE_MARITE
23 SI23 LEUCOPENIE
24 SI24 LIMFOPENIE
25 SI25 MALABSORBTIE
26 SI26 MARKERI_VIRALI(HEPATITA)
27 SI27 NODULI_PULMONARI
28 SI28 OBSTRUCTIA_CAILOR_BILIARE
29 SI29 PATA_PE_PLAMAN
30 SI30 POLIPI_INTESTINALI
31 SI31 PRESIUNE/VOLUM_CARDIAC_MARIT
32 SI32 PRESIUNE/VOLUM_CARDIAC_NORMAL
33 SI33 PROTEINURIE
34 SI34 (SANGERARI)_VARICE_ESOFAGIENE
35 SI35 SCADEREA_ELASTICITATII_VASELOR
36 SI36 SCADEREA_IMUNITATII
37 SI37 SER_AMILAZE_CRESCUT
38 SI38 SEROZITA
39 SI39 SINTEZA_SCAZUTA_DE_ALBUMINA
40 SI40 STENOZA_ MITRALA/AORTICA
41 SI41 STENOZA_INTESTIN_SUBTIRE
42 SI42 T3,T4_SERICE_SCAZUTE

43 SI43 T3,T4_SERICE_CRESCUTE
44 SI44 TRANSAMINAZE_SERICE_CRESCUTE(SGOT)
45 SI45 TRANSAMINAZE_SERICE_CRESCUTE(TGP,TGO)
46 SI46 TRANSUDARE_LICHID_IN_ABDOMEN
47 SI47 TRANSUDARE_LICHID_IN_ALVEOLELE_PULMONARE
48 SI48 TROMBOCITOPENIE
49 SI49 TROMBOZA_VENEI_SPLENICE
50 SI50 TSH_SERIC_CRESCUT
51 SI51 TSH_SERIC_SCAZUT
52 SI52 ULCERATII_INTESTIN_GROS
53 SI53 VDRL_FALS_POZITIV

Anexa 3. Ponderile simptomelor în cadrul bolilor

1 **ANGINA_PECTORALA**

simptom	pondere
ARITMII	0.7
DISPNEE	0.8
DURERI_PIEPT_CE_CEDEAZA_LA_NITROGLICERINA	0.9
GREATA/VOMA	0.5

2 **ARTRITA_REUMATOIDA**

ANEMIE_DE_BOALA_CRONICA	0.5
ANOREXIE	0.5
ARTRITE_LA_CEL_PUTIN_3_INCHEIETURI	0.9
ARTRITE_LA_INCHEIETURILE_MAINILOR	0.8
ARTRITE_SIMETRICE	0.8
FACTOR_REUMATOID_SERIC	0.8
NODULI_REUMATOIZI_SUBCUTANATI	0.7
OBOSEALA	0.5
REDOARE_MATINALA	0.7

3 **BOALA_CHRON**

DEFICIENTA_ENZIMATICA_IN_INTESTIN	1
DIARIE_CU_SANGE_DIGERAT	0.6
DURERI_ABDOMEN	0.6
EDEM_MUCOASA_INTESTIN	0.9
POLIPI_INTESTINALI	0.9
SLABIRE	0.5
STENOZA_INTESTIN_SUBTIRE	0.9

4 **CANCER_COLON/INTESTIN**

DIARIE_CU_SANGE_DIGERAT	0.7
DURERI_ABDOMEN	0.6
GREATA/VOMA	0.6
MELENA	0.7
SLABIRE	0.7

5 **CANCER_PANCREAS**

ANOREXIE	0.8
HEPATOMEGALIE	0.8
ICTER	0.8
OBSTRUCTIA_CAILOR_BILIARE	0.7
SEMN_TERRIER	1
SLABIRE	0.5

6 **CANCER_PLAMANI**

DISPNEE	0.7
SLABIRE	0.5
SUBFEBRILITATE	0.7
TUSE_PERMANENTA	0.8

7 **CIROZA**

ANEMIE_DE_BOALA_CRONICA	0.6
ANOREXIE	0.6
ASCITA	1
ENCEFALOPATIE	0.7
FIBRINOGEN_CRESCUT	0.7
LEUCOPENIE	0.7
MELENA	0.7
OBOSEALA	0.5
PROTEINURIE	0.8
SANGERARI_VARICEALE(ESOFAG)	1
SINTEZA_SCAZUTA_DE_ALBUMINA	0.8
SPLENOMEGALIE	0.9
TRANSAMINAZE_SERICE_CRESCUTE(TGP,TGO)	1
TRANSUDARE_LICHID_IN_ABDOMEN	1
TROMBOCITOPENIE	1

8 **COLITA_ULCEROASA**

DIARIE_CU_SANGE_DIGERAT	0.7
DURERI_ABDOMEN	0.5
EDEM_MUCOASA_INTESTIN	1
POLIPI_INTESTINALI	1
SLABIRE	0.5
STENOZA_INTESTIN_SUBTIRE	1
ULCERATII_INTESTIN_GROS	1

9 **DIABET**

AFECTARI_VEDERE	0.6
GLUCOZA_MARE_IN_SANGE	1
OBOSEALA	0.5
POLIURIE	0.8
SCADEREA_IMUNITATII	0.7
SLABIRE	0.9

10 **EMBOLIE_PULMONARA**

DISPNEE	0.5
HEMOPTIZIE	0.7
TURGESCENTA_JUGULARA	0.7
TUSE_PERMANENTA	0.8

11 **ENTERITA_INFECTIOASA**

DIARIE_FARA_SANGE	0.8
DURERI_ABDOMEN	0.5
GREATA/VOMA	0.5
MALABSORBTIE	0.7
SLABIRE	0.5

12 **GLOMERULONEFRITA_MEMBRANOASA**

ASLO_CRESCUT	0.7
EDEM	0.9
FIBRINOGEN_CRESCUT	0.8
HEMATURIE	1
OBOSEALA	0.7
PROTEINURIE	1

13 **HEPATITA**

ANOREXIE	0.8
ANTICORPI_ANTINUCLEARI(ANA)_CRESCUTI	0.9
DISTRUGEREA_MEMBRANEI_HEPATICE	1
GREATA/VOMA	0.9
HEPATOMEGALIE	1
ICTER	1
MARKERI_VIRALI(HEPATITA)	1
TRANSAMINAZE_SERICE_CRESCUTE(TGP,TGO)	1

14 **HIPERTIROIDIE**

HIPEREMOTIVITATE	0.8
SLABICIUNE_MUSCULARA	0.6
SLABIRE	0.6
SUBFEBRILITATE	0.7
T3,T4_SERICE_CRESCUTE	1
TAHICARDIE	0.7
TSH_SERIC_SCAZUT	1

15 **HIPOTIROIDIE**

BRADICARDIE	0.6
EDEM	0.7
ENZIME_MUSCULARE_CRESCUTE(CK,ALDOLAZA,MIOGLOBINA_SERICA)	0.8
INGRASARE	0.8
MENORAGIE	0.8
OBOSEALA	0.8
PAR_DE_CALITATE_PROASTA	0.6
PIELE_USCATA	0.6
SLABICIUNE_MUSCULARA	0.6
TAHICARDIE	0.6
T3,T4_SERICE_SCAZUTE	1
TSH_SERIC_CRESCUT	1

16 **INFARCT_MIOCARDIC**

ARITMII	0.8
DISPNEE	0.9
DURERI_PIEPT_CE_NU_CEDEAZA_LA_NITROGLICERINA	1
FIBRINOGEN_CRESCUT	0.6
GREATA/VOMA	0.5
PANICA	1
TENSIUNE_MICA	0.7
TRANSAMINAZE_SERICE_CRESCUTE(SGOT)	0.8

17 **INFECTIE_URINARA**

DURERI_SUPRAPUBICE	0.8
FEBRA_RIDICATA	0.8
HIDRONEFROZA	0.9
INCONTINENTA_URINARA	0.9
PIELONEFRITA	0.8
USTURIME_URINARA	1
URINA_TULBURE	1

18 **INSUFICIENTA_CARDIACA_CONGESTIVA**

ANEMIE_DE_BOALA_CRONICA	0.5
ANOREXIE	0.5
ASCITA	0.7
CONGESTIE_VASCULARA_IN_PLAMAN	0.7
CRESTEREA_VOLUMULUI_IN_VASELE_CENTRALE_IN_POZITIA_CULCAT	0.6
DISPNEE	0.9
DISPNEE_NOCTURNA_PAROXISTICA	1
EDEM	0.7

HEMOPTIZIE	0.5
INIMA_MARITA	1
OBOSEALA	0.5
RALURI_PULMONARE	1
RITM_DE_GALOP	1
TAHICARDIE	0.7
TRANSUDARE_LICHID_IN_ALVEOLELE_PULMONARE	0.7
TURGESCENTA_JUGULARA	0.9
TUSE_NOAPTEA	0.8

19 **INSUFICIENTA_RENALA**

BUN_CRESCUT	1
CREATININA_CRESCUTA	1
ENCEFALOPATIE	0.6
GFR_SCAZUTA	1

20 **LES**

ALOPECIE	0.5
ANEMIE_HEMOLITICA	0.7
ANTICORPI_ANTINUCLEARI(ANA)_CRESCUTI	0.8
ARTRITE	0.5
CELULE_LE	1
C3,C4_SCAZUTE(COMPLEMENT_SERIC)	1
CRP_CRESCUTA	1
ENCEFALOPATIE	0.5
FEBRA_RIDICATA	0.5
FOTOSENSIBILITATE	0.6
LEUCOPENIE	0.6
LEZIUNI_DISCOIDE	1
LIMFOPENIE	0.6
PROTEINURIE	0.6
RASH_MALAR	0.9
SEROZITA	0.7
TROMBOCITOPENIE	0.6
ULCERATII_ORALE	0.5
VDRL_FALS_POZITIV	0.8

21 **LITIAZA_BILIARA**

ENCEFALOPATIE	0.5
GREATA/VOMA	0.7
ICTER	0.8
OBSTRUCTIA_CAILOR_BILIARE	1

22 **LITIAZA_RENALA**

DIZURIE	0.8
DURERE_LOMBARA	0.9
GREATA/VOMA	0.5
HEMATURIE	0.9

23 **MIOCARDITA**

ARITMII	0.7
DISPNEE	0.8
OBOSEALA	0.7
RITM_DE_GALOP	0.9

24 **PANCREATITA**

DURERI_ABDOMEN	0.5
GREATA/VOMA	0.5
SANGERARE_RETROPERITONEALA	1
SER_AMILAZE_CRESCUT	1
SLABIRE	0.5

25 **PERICARDITA**

ASCITA	0.6
DISPNEE	0.8
DURERI_IN_PIEPT_LA_INSPIRATIE	0.8
EDEM	0.7
FEBRA_RIDICATA	0.7
FRECATURA_PERICARDICA	1
HEPATOMEGALIE	0.6
TURGESCENTA_JUGULARA	0.9

26 **PNEUMONIE**

EXPECTORATIE_PURULENTA	1
FEBRA_RIDICATA	0.8
PATA_PE_PLAMAN	1
TUSE_PERMANENTA	0.5

27 **POLIMIOZITA**

ANTICORPI_ANTINUCLEARI(ANA)_CRESCUTI	0.8
ANTICORPI_Mi-2,SRP_CRESCUTI	1
ENZIME_MUSCULARE_CRESCUTE(CK,ALDOLAZA,MIOGLOBINA_SERICA)	1

	SLABICIUNE_MUSCULARA	0.5

28 **TBC**

	BACIL KOCH	1
	HEMOPTIZIE	0.7
	LEZIUNI_DISCOIDE	0.6
	NODULI_PULMONARI	1
	SLABIRE	0.5
	SUBFEBRILITATE	0.7
	TUSE_PERMANENTA	0.5

29 **TOXIINFECTIE_ALIMENTARA**

	DIARIE_FARA_SANGE	0.5
	DURERI_ABDOMEN	0.5
	GREATA/VOMA	0.5

30 **VIROZA_PULMONARA**

	DISPNEE	0.5
	EXPECTORATIE	0.5
	FEBRA_RIDICATA	0.5
	HILURI_PULMONARE_MARITE	1
	TUSE_PERMANENTA	0.5

Anexa 4. Constrângerile de tip atac

ID	NUMEC	NOD1	NOD2	CONTEXT	TIP
1	a1	BI3	BH1	BH1	a
2	a1	BI3	BH1	BI3	a
3	a2	BI1	BI2	BI1	a
4	a2	BI1	BI2	BI2	a
5	a3	BI2	BI1	BI1	a
6	a3	BI2	BI1	BI2	a
7	a4	BGI1	BGI4	BGI1	a
8	a4	BGI1	BGI4	BGI4	a
9	a5	BGI2	BGI4	BGI2	a
10	a5	BGI2	BGI4	BGI4	a
11	a6	BGI3	BH2	BGI3	a
12	a6	BGI3	BH2	BH2	a
13	a7	BGI6	BH2	BGI6	a
14	a7	BGI6	BH2	BH2	a
15	a8	BGI3	BGI6	BGI6	a
16	a8	BGI3	BGI6	BGI3	a
17	a9	SI32	BI3	BI3	a
18	a10	BE3	SI32	BE3	a
19	a11	BREUM3	BE3	BREUM3	a
20	a11	BREUM3	BE3	BE3	a

Anexa 5. Lista nodurilor şi calitatea lor față de fiecare context

ID_NOD	NUME_NOD	CONTEXT	TIP	PONDERE
1	SS1	BE1	s	1
2	SS2	BREUM2	s	0.5
3	SS3	BREUM1	s	0.5
4	SS3	BH2	s	0.8
5	SS3	BGI3	s	0.8
6	SS3	BH1	s	0.6
7	SS3	BI3	s	0.5
8	SS4	BI1	s	0.7
9	SS4	BI2	n	0.8
10	SS4	BI4	s	0.7
11	SS5	BREUM2	n	0.5
12	SS6	BREUM1	s	0.9
13	SS7	BREUM1	s	0.8
14	SS8	BREUM1	s	0.8
15	SS9	BI5	s	0.6
16	SS9	BI3	s	0.7
17	SS9	BH1	n	1
18	SS10	BE3	s	0.6
19	SS11	BGI1	s	0.6
20	SS11	BGI2	s	0.7
21	SS11	BGI4	s	0.7
22	SS12	BGI5	n	0.8
23	SS12	BGI8	n	0.5
24	SS13	BI1	s	0.8
25	SS13	BI2	n	0.9
26	SS13	BI3	n	0.9
27	SS13	BI4	s	0.8
28	SS13	BI5	s	0.8
29	SS13	BP1	s	0.7
30	SS13	BP2	n	0.5
31	SS13	BP5	s	0.5
32	SS14	BI3	n	1
33	SS15	BREN4	s	0.8
34	SS16	BGI7	s	0.5
35	SS16	BGI1	s	0.6
36	SS16	BGI4	s	0.5
37	SS16	BGI2	s	0.6
38	SS16	BGI8	n	0.5
39	SS16	BGI5	n	0.5
40	SS17	BREN4	n	0.9
41	SS18	BI5	n	0.8
42	SS19	BI1	n	0.9
43	SS20	BI2	n	1
44	SS21	BREN2	s	0.8
45	SS22	BREN1	s	0.9
46	SS22	BE3	s	0.7
47	SS22	BI5	s	0.7
48	SS22	BI3	s	0.7

49	SS23	BGI6	s	0.5
50	SS23	BREN3	s	0.6
51	SS23	BREUM2	s	0.5
52	SS23	BH1	s	0.7
53	SS24	BP5	s	0.5
54	SS25	BP3	s	1
55	SS26	BP3	n	0.8
56	SS26	BP5	n	0.5
57	SS26	BREN2	n	0.8
58	SS26	BREUM2	s	0.5
59	SS26	BI5	s	0.7
60	SS27	BREUM2	s	0.6
61	SS28	BI5	n	1
62	SS29	BI1	s	0.5
63	SS29	BH2	s	0.9
64	SS29	BGI6	s	0.7
65	SS29	BGI7	s	0.5
66	SS29	BI2	s	0.5
67	SS29	BGI2	s	0.6
68	SS29	BGI5	n	0.5
69	SS29	BGI8	n	0.5
70	SS29	BREN4	s	0.5
71	SS30	BREN4	s	0.9
72	SS30	BREN1	s	1
73	SS31	BP4	s	0.7
74	SS31	BI3	s	0.5
75	SS31	BP2	n	0.7
76	SS32	BH2	n	1
77	SS32	BGI3	s	0.8
78	SS32	BI5	s	0.6
79	SS33	BE2	s	0.8
80	SS34	BH2	n	1
81	SS34	BGI3	s	0.8
82	SS34	BGI6	n	0.8
83	SS35	BI3	n	1
84	SS36	BREN2	s	0.9
85	SS37	BE3	s	0.8
86	SS38	BP4	s	0.6
87	SS38	BREUM2	n	1
88	SS39	BGI2	s	0.7
89	SS39	BH1	s	0.7
90	SS40	BE3	s	0.8
91	SS41	BREUM1	s	0.7
92	SS42	BE1	s	0.8
93	SS42	BREUM1	s	0.5
94	SS42	BREN1	s	0.7
95	SS42	BE3	s	0.8
96	SS42	BI4	s	0.7
97	SS42	BH1	s	0.5
98	SS42	BI3	s	0.5

99	SS43	BI2	n	1
100	SS44	BE3	s	0.6
101	SS45	BE3	s	0.6
102	SS46	BE1	n	0.7
103	SS47	BI3	n	1
104	SS48	BREUM2	n	0.9
105	SS49	BREUM1	s	0.7
106	SS50	BI4	n	0.9
107	SS50	BI3	n	1
108	SS51	BGI7	s	1
109	SS52	BGI3	n	1
110	SS53	BE3	s	0.6
111	SS53	BREUM3	n	0.5
112	SS54	BG1	s	0.5
113	SS54	BGI2	s	0.7
114	SS54	BGI4	s	0.5
115	SS54	BGI7	s	0.5
116	SS54	BGI3	s	0.5
117	SS54	BE1	n	0.9
118	SS54	BE2	s	0.6
119	SS54	BP4	n	0.5
120	SS54	BP1	s	0.5
121	SS54	BGI5	s	0.5
122	SS55	BH1	n	0.9
123	SS56	BE2	s	0.7
124	SS56	BP1	s	0.7
125	SS56	BP4	n	0.7
126	SS57	BE2	s	0.7
127	SS57	BI3	s	0.7
128	SS58	BI2	s	0.7
129	SS59	BI5	s	0.9
130	SS59	BI3	n	0.9
131	SS59	BP2	s	0.7
132	SS60	BI3	s	0.8
133	SS61	BP4	n	0.5
134	SS61	BP5	n	0.5
135	SS61	BP1	n	0.8
136	SS61	BP3	n	0.5
137	SS61	BP2	n	0.8
138	SS62	BREUM2	s	0.5
139	SS63	BREN2	n	1
140	SS64	BREN2	n	1
141	SI1	BI3	s	0.5
142	SI1	BH1	s	0.6
143	SI1	BREUM1	s	0.5
144	SI2	BREUM2	s	0.7
145	SI3	BH2	n	0.9
146	SI3	BREUM2	n	0.8
147	SI4	BREUM3	n	1
148	SI5	BREN1	n	0.7

149	SI6	BREN3	n	1
150	SI7	BREUM2	n	1
151	SI8	BREUM2	n	1
152	SI9	BI3	n	0.7
153	SI10	BREN3	n	1
154	SI11	BI3	n	0.6
155	SI12	BREUM2	n	1
156	SI13	BGI1	n	1
157	SI14	BH2	n	1
158	SI15	BGI1	n	0.9
159	SI15	BGI4	n	1
160	SI16	BREUM3	n	1
161	SI16	BE3	n	0.8
162	SI17	BREUM1	s	0.8
163	SI18	BH1	n	0.7
164	SI18	BI2	n	0.6
165	SI18	BREN1	n	0.8
166	SI19	BREN3	n	1
167	SI20	BE1	n	0.5
168	SI21	BREN2	s	0.9
169	SI22	BP5	n	1
170	SI23	BH1	s	0.7
171	SI23	BREUM2	s	0.6
172	SI24	BREUM2	s	0.6
173	SI25	BGI5	s	0.7
174	SI26	BH2	s	1
175	SI27	BP4	n	1
176	SI28	BGI3	s	0.7
177	SI28	BGI6	n	1
178	SI29	BP3	n	1
179	SI30	BGI1	s	0.9
180	SI30	BGI4	s	1
181	SI31	BE2	n	1
182	SI31	BI3	n	1
183	SI31	SI40	n	1
184	SI33	BH1	s	0.8
185	SI33	BREN1	n	1
186	SI33	BREUM2	s	0.6
187	SI34	BH1	s	1
188	SI35	BE1	n	1
189	SI35	SI40	n	1
190	SI36	BE1	s	0.7
191	SI37	BGI7	n	1
192	SI38	BREUM2	s	0.7
193	SI39	BH1	n	0.8
194	SI41	BGI1	n	0.9
195	SI41	BGI4	n	1
196	SI42	BE3	n	1
197	SI43	BE2	n	1
198	SI44	BI2	n	0.8

199	SI45	BH1	n	1
200	SI45	BH2	n	1
201	SI46	BH1	n	1
202	SI46	BI3	n	0.7
203	SI47	BI3	n	0.7
204	SI48	BH1	n	1
205	SI48	BREUM2	n	0.6
206	SI49	BGI7	n	1
207	SI50	BE3	n	1
208	SI51	BE2	n	1
209	SI52	BGI4	n	1
210	SI53	BREUM2	s	0.8

Anexa 6. Codul sursă (selecții)

//PanouAlgoritm.java

```
package gui;

import algoritm.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.util.*;
import javax.swing.event.*;

public class PanouAlgoritm extends javax.swing.JPanel {
    Monitor m; dbt b;
    public java.util.List necesare;
    ListaMea lista1, lista2, lista3;
    JListData dat1,dat2;
    JListData dat3;
    java.util.List li1,li2;
    public java.util.List date_confirmate, date_infirmate;

    public PanouAlgoritm(ListaMea l1, ListaMea l2, ListaMea l3, JListData d1, JListData
d2, JListData d3)
         throws IOException {
        lista1=l1; lista2=l2; lista3=l3;
        dat1=d1;dat2=d2;dat3=d3;
        date_confirmate=new ArrayList(); date_infirmate=new ArrayList();
        m=new Monitor(this);b=new dbt(this);

        initComponents();
    }

    private void initComponents() {
        jPanel1 = new javax.swing.JPanel();
        jPanel2 = new javax.swing.JPanel();

        jScrollPane1 = new javax.swing.JScrollPane();
        jScrollPane2 = new javax.swing.JScrollPane();
        jScrollPane3 = new javax.swing.JScrollPane();
        jButtonOK = new javax.swing.JButton();
        jButtonPrima = new javax.swing.JButton();
        jButtonReluare = new javax.swing.JButton();
        jButtonTesteaza = new javax.swing.JButton();
        jPanel3 = new javax.swing.JPanel();
        jScrollPane4 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jScrollPane5 = new javax.swing.JScrollPane();
```

```
        jTextArea2 = new javax.swing.JTextArea();
        jScrollPane6 = new javax.swing.JScrollPane();
        jTextArea3 = new javax.swing.JTextArea();
        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel3 = new javax.swing.JLabel();

        jPanel1.setBackground(new java.awt.Color(204, 255, 102));
        jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder(null,
"ALGORITM",              javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION,  new  java.awt.Font("Bookman
Old Style", 1, 12)));
        jPanel2.setBackground(new java.awt.Color(153, 255, 153));
        jPanel2.setBorder(new javax.swing.border.MatteBorder(null));
        jScrollPane1.setBackground(new java.awt.Color(255, 255, 204));
        jScrollPane1.setBorder(javax.swing.BorderFactory.createTitledBorder(null,
"Simptome                                                                  superficiale:",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION,  new  java.awt.Font("Bookman
Old Style", 0, 11)));
        jScrollPane1.getViewport().add(lista1);

        jScrollPane2.setBackground(new java.awt.Color(255, 255, 204));
        jScrollPane2.setBorder(javax.swing.BorderFactory.createTitledBorder(null,
"Simptome                                  de                                  profunzime:",
javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION,  new  java.awt.Font("Bookman
Old Style", 0, 11)));
        jScrollPane2.getViewport().add(lista2);

        jScrollPane3.setBackground(new java.awt.Color(255, 255, 204));
        jScrollPane3.setBorder(javax.swing.BorderFactory.createTitledBorder(null,
"Manifestari necesare:", javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION,  new  java.awt.Font("Bookman
Old Style", 0, 11)));
        jScrollPane3.getViewport().add(lista3);

        jButtonOK.setFont(new java.awt.Font("Bookman Old Style", 1, 11));
        jButtonOK.setText("OK");
        jButtonOK.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonOKActionPerformed(evt);
            }
        });

        jButtonPrima.setFont(new java.awt.Font("Bookman Old Style", 1, 11));
```

```
        jButtonPrima.setText("Prima rulare");
        jButtonPrima.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonPrimaActionPerformed(evt);
            }
        });

        jButtonReluare.setFont(new java.awt.Font("Bookman Old Style", 1, 11));
        jButtonReluare.setText("Reluare");
        jButtonReluare.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonReluareActionPerformed(evt);
            }
        });

        jButtonTesteaza.setFont(new java.awt.Font("Bookman Old Style", 1, 11));
        jButtonTesteaza.setText("Testeaza");
        jButtonTesteaza.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButtonTesteazaActionPerformed(evt);
            }
        });

        org.jdesktop.layout.GroupLayout jPanel2Layout = new
org.jdesktop.layout.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel2Layout.createSequentialGroup()

.add(jPanel2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(jPanel2Layout.createSequentialGroup()
                        .add(19, 19, 19)
                        .add(jScrollPane1,  org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
288, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED))
                    .add(org.jdesktop.layout.GroupLayout.TRAILING,
jPanel2Layout.createSequentialGroup()
                        .addContainerGap(158, Short.MAX_VALUE)
                        .add(jButtonPrima)
                        .add(48, 48, 48)))

.add(jPanel2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(jPanel2Layout.createSequentialGroup()
                        .add(8, 8, 8)
```

```
                .add(jScrollPane2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
275, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED, 17,
Short.MAX_VALUE)

.add(jPanel2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(org.jdesktop.layout.GroupLayout.TRAILING,
jPanel2Layout.createSequentialGroup()
                        .add(jButtonTesteaza)
                        .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED))
                    .add(org.jdesktop.layout.GroupLayout.TRAILING,
jPanel2Layout.createSequentialGroup()
                        .add(jButtonOK)
                        .add(22, 22, 22)))
                .add(jScrollPane3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
233, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
              .add(jPanel2Layout.createSequentialGroup()
                .add(31, 31, 31)
                .add(jButtonReluare)))
            .addContainerGap())
      );
      jPanel2Layout.setVerticalGroup(
         jPanel2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
         .add(jPanel2Layout.createSequentialGroup()
            .add(36, 36, 36)

.add(jPanel2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
              .add(org.jdesktop.layout.GroupLayout.TRAILING, jScrollPane3,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 200, Short.MAX_VALUE)
              .add(jScrollPane2, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 200,
Short.MAX_VALUE)
              .add(jScrollPane1, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, 200,
Short.MAX_VALUE))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jPanel2Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
              .add(jButtonPrima)
              .add(jButtonReluare))
            .addContainerGap())
         .add(org.jdesktop.layout.GroupLayout.TRAILING,
jPanel2Layout.createSequentialGroup()
            .addContainerGap(108, Short.MAX_VALUE)
            .add(jButtonOK)
            .add(37, 37, 37)
            .add(jButtonTesteaza)
            .add(87, 87, 87))
```

```
        );

        jPanel3.setBackground(new java.awt.Color(153, 255, 153));
        jPanel3.setBorder(javax.swing.BorderFactory.createLineBorder(new
java.awt.Color(0, 0, 0)));
        jTextArea1.setBackground(new java.awt.Color(255, 255, 204));
        jTextArea1.setColumns(20);
        jTextArea1.setFont(new java.awt.Font("Bookman Old Style", 1, 10));
        jTextArea1.setRows(5);
        jScrollPane4.setViewportView(jTextArea1);

        jTextArea2.setBackground(new java.awt.Color(255, 255, 204));
        jTextArea2.setColumns(20);
        jTextArea2.setFont(new java.awt.Font("Bookman Old Style", 1, 10));
        jTextArea2.setRows(5);
        jScrollPane5.setViewportView(jTextArea2);

        jTextArea3.setBackground(new java.awt.Color(255, 255, 204));
        jTextArea3.setColumns(20);
        jTextArea3.setFont(new java.awt.Font("Bookman Old Style", 1, 10));
        jTextArea3.setRows(5);
        jScrollPane6.setViewportView(jTextArea3);

        jLabel1.setFont(new java.awt.Font("Bookman Old Style", 1, 11));
        jLabel1.setText("Solutii posibile:");

        jLabel2.setFont(new java.awt.Font("Bookman Old Style", 1, 11));
        jLabel2.setText("Ipoteze infirmate:");

        jLabel3.setFont(new java.awt.Font("Bookman Old Style", 1, 11));
        jLabel3.setText("Solutii partiale admisibile:");

        org.jdesktop.layout.GroupLayout jPanel3Layout = new
org.jdesktop.layout.GroupLayout(jPanel3);
        jPanel3.setLayout(jPanel3Layout);
        jPanel3Layout.setHorizontalGroup(
            jPanel3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel3Layout.createSequentialGroup()
                .addContainerGap()

.add(jPanel3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                    .add(jLabel1)
                    .add(jScrollPane4,      org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
315, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE))
                .add(20, 20, 20)
```

```
.add(jPanel3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                .add(jScrollPane5,      org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
174, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .add(jLabel2))
            .add(17, 17, 17)

.add(jPanel3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                .add(jScrollPane6,  org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,  397,
Short.MAX_VALUE)
                .add(jLabel3))
            .addContainerGap())
        );
        jPanel3Layout.setVerticalGroup(
            jPanel3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel3Layout.createSequentialGroup()
                .add(28, 28, 28)

.add(jPanel3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.BASELINE)
                .add(jLabel1)
                .add(jLabel3)
                .add(jLabel2))
            .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED)

.add(jPanel3Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
                .add(jScrollPane4,  org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,  108,
Short.MAX_VALUE)
                .add(jScrollPane5,  org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,  108,
Short.MAX_VALUE)
                .add(jScrollPane6,  org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,  108,
Short.MAX_VALUE))
            .addContainerGap())
        );

        org.jdesktop.layout.GroupLayout      jPanel1Layout      =      new
org.jdesktop.layout.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(org.jdesktop.layout.GroupLayout.TRAILING,
jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.add(jPanel1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.TRAILING)
```

```
                .add(org.jdesktop.layout.GroupLayout.LEADING, jPanel3,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .add(org.jdesktop.layout.GroupLayout.LEADING, jPanel2,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
            .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel1Layout.createSequentialGroup()
                .addContainerGap()
                .add(jPanel2, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(org.jdesktop.layout.LayoutStyle.RELATED,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .add(jPanel3, org.jdesktop.layout.GroupLayout.PREFERRED_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.PREFERRED_SIZE)
                .addContainerGap())
        );

        org.jdesktop.layout.GroupLayout layout = new
org.jdesktop.layout.GroupLayout(this);
        this.setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel1, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(org.jdesktop.layout.GroupLayout.LEADING)
            .add(jPanel1, org.jdesktop.layout.GroupLayout.DEFAULT_SIZE,
org.jdesktop.layout.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
    }// </editor-fold>

    private void jButtonReluareActionPerformed(java.awt.event.ActionEvent evt) {
try{
m.preiaDate();Iterator it;
    java.util.List AC=new ArrayList(), RC=new ArrayList();
    java.util.List ConsActnou=new ArrayList();
    java.util.List V2nou=new ArrayList();

                b.Ipoteze_noi=b.disp.creaza_Context(date_confirmate, b.Ipoteze_infirmate);
```

```
                AC=b.disp.creaza_ConstrActive(b.Ipoteze_noi);
                b.Ipoteze_noi.removeAll(b.Ipoteze_considerate);
                b.Ipoteze_considerate.addAll(b.Ipoteze_noi);

                ConsActnou.addAll(AC);
                ConsActnou.addAll((ArrayList)((b.SolutiiConsAct).get(0)));

                 if(!b.Ipoteze_infirmate.isEmpty())
                {it=b.Ipoteze_infirmate.iterator();
                while(it.hasNext())
                {String context=it.next().toString();
                 RC.addAll(b.disp.creazaRC(context, ConsActnou));}}

                 V2nou.addAll(b.Ipoteze_noi);
    java.util.List                                                    Dom_Curente=new
ArrayList();Dom_Curente=b.disp.creaza_VarActive();
    ConsActnou.removeAll(RC); ConsActnou=removeDuplicates2(ConsActnou);

    for(int i=0; i<b.SolutiiAsignare.size();i++)// in solutii: asignari
    {java.util.List aux=new ArrayList();
     aux.addAll((ArrayList)((b.SolutiiAsignare).get(i)));
         boolean c=true;
         it=aux.iterator();
         while(it.hasNext())
         {VarVal vv=(VarVal)it.next();
         if((vv.val.equals("in"))&&(b.Ipoteze_infirmate.contains(vv.var))) c=false;}

      if(c==true)
       b.rezolva(new
ArrayList(),V2nou,Dom_Curente,(ArrayList)((b.SolutiiAsignare).get(i)),
              new ArrayList(), ConsActnou);}
      ((ArrayList)(b.SolutiiAsignare)).clear();
}
catch(IOException e){};
   }

   private void jButtonPrimaActionPerformed(java.awt.event.ActionEvent evt) {
try{
  java.util.List VarActive=new ArrayList(), ConsAct=new ArrayList();m.preiaDate();
        VarActive.addAll(b.disp.creaza_Context(date_confirmate,b.Ipoteze_infirmate));
        ConsAct=b.disp.creaza_ConstrActive(VarActive);
        b.Ipoteze_considerate.addAll(VarActive);
    b.rezolva(new ArrayList(),VarActive, b.disp.creaza_VarActive(),new ArrayList(),new
ArrayList(),ConsAct);
}
```

```
catch(IOException e){};

    }

    private void jButtonTesteazaActionPerformed(java.awt.event.ActionEvent evt) {
      dat3.removeAll();
      Iterator  it=necesare.iterator();
        while (it.hasNext()){
        String s=it.next().toString().trim();
        if (!date_confirmate.contains(b.tradu(s)))
           dat3.addElement(b.tradu(s));}
    }

    private void jButtonOKActionPerformed(java.awt.event.ActionEvent evt) {
int i;

     Vector vect=new Vector(100);
//trb sa dea valori listei de date confirmate - cu denumirea medicala simptome
        String[] s1=lista1.getSelectedItems();
        String[] s2=lista2.getSelectedItems();

        for(i=0;i<s1.length;i++)
        {vect.addElement(s1[i].trim());
         date_confirmate.add(s1[i].trim());}
               for(i=0;i<s2.length;i++)
         {vect.addElement(s2[i].trim());
         date_confirmate.add(s2[i].trim());}
         date_confirmate=removeDuplicates(date_confirmate);

        li1=new ArrayList();li2=new ArrayList();
        String[] s3=lista3.getSelectedItems();
        for(i=0;i<s3.length ;i++)
        {li1.add(b.traduinv(s3[i]));
           li2.add(s3[i]);}


        date_infirmate.addAll(li1);
        date_infirmate=removeDuplicates(date_infirmate);
        b.elimina_infirmate(li2);
        m.puneDate(date_confirmate,date_infirmate);
    }


   public java.util.List removeDuplicates2(java.util.List L)
  {java.util.List rez=new ArrayList();
   Iterator it=L.iterator();
```

```
 while (it.hasNext())
 {Constraint o=(Constraint)it.next();
    if (!o.isContained(rez)) rez.add(o);}
return rez;}//end removeDuplicates2


public java.util.List removeDuplicates(java.util.List L)
{java.util.List rez=new ArrayList();
 Iterator it=L.iterator();
 while (it.hasNext())
 {Object o=(Object)it.next();
    if (!rez.contains(o)) rez.add(o);}
return rez;}//end removeDuplicates


  // Variables declaration - do not modify
  private javax.swing.JButton jButtonOK;
  private javax.swing.JButton jButtonPrima;
  private javax.swing.JButton jButtonReluare;
  private javax.swing.JButton jButtonTesteaza;
  private javax.swing.JLabel jLabel1;
  private javax.swing.JLabel jLabel2;
  private javax.swing.JLabel jLabel3;
  private javax.swing.JPanel jPanel1;
  private javax.swing.JPanel jPanel2;
  private javax.swing.JPanel jPanel3;
  public javax.swing.JScrollPane jScrollPane1;
  private javax.swing.JScrollPane jScrollPane2;
  private javax.swing.JScrollPane jScrollPane3;
  private javax.swing.JScrollPane jScrollPane4;
  private javax.swing.JScrollPane jScrollPane5;
  private javax.swing.JScrollPane jScrollPane6;
  public javax.swing.JTextArea jTextArea1;
  public javax.swing.JTextArea jTextArea2;
  public javax.swing.JTextArea jTextArea3;
  // End of variables declaration

}
```

//dbt.java

```
package algoritm;

import bd.*;
import gui.*;
import java.util.*;
import java.io.*;
import javax.swing.*;

//CONTEXT= O LISTA CU IPOTEZELE CURENTE DE LA SELECTIE.- DIRIJEAZA
SELECTIA DE
//VARIABILE CURENTE SI CONSTRANGERILE


class Explicatie
{public VarVal vv; public List asignare;
  //public Explicatie (VarVal v, List a) {vv=v; asignare=a;}
 public Explicatie (String Var, String Val, List a,List Asignare)
 {vv=new VarVal(Var,Val);
  asignare=new ArrayList();
  Iterator it=Asignare.iterator();

  while (it.hasNext())
  {VarVal vv=(VarVal)it.next();
  if (a.contains(vv.var)) asignare.add(vv);}}
}//end Explicatie

public class dbt {
    public  boolean sol;

public dispecer disp;
public List Ipoteze_considerate, Ipoteze_noi, Ipoteze_infirmate;
List fisiere;String numefis;
Factory fact;
BufferedReader in;
PanouAlgoritm med;
public List SolutiiV1, SolutiiV2, SolutiiDC, SolutiiAsignare, SolutiiExplicElim,
SolutiiConsAct;

public boolean acoperire_fara_x(List asign, String x)
{Iterator t=asign.iterator();String v, val;
 List Simptome=new ArrayList();
while (t.hasNext())
{VarVal vvv=(VarVal)t.next();
 v=vvv.var; val=vvv.val;
 if((val.equals("in"))&&(!v.equals(x)))Simptome.addAll(disp.simpt(v));}
```

```
if (Simptome.containsAll(med.date_confirmate)) return true;
return false;}

public boolean acoperire_minimala(List asign)
{Iterator t=asign.iterator();String v, val;
 boolean r=true;
 if(!acoperire_fara_x(asign,"")) r=false;

while (t.hasNext())
{VarVal vvv=(VarVal)t.next();
 v=vvv.var; val=vvv.val;
 if((val.equals("in"))&&(acoperire_fara_x(asign,v))==true)
     r= false;}

 return r;}

//V=lista de variabile asignate; Asignarec= lista de VarVal= var asignate cu valorile lor
public boolean admisibil(List V, List Asignarec,List ConsAct)
 {List Asignare=new ArrayList();Asignare.addAll(Asignarec);
  Iterator t,ai;
   int i1=-1,i2=-1;
String attack=null,v,val;

Iterator t1=Asignare.iterator();
while (t1.hasNext())
{attack=null;
 VarVal vvv=(VarVal)t1.next();
 v=vvv.var; val=vvv.val;

   if(val.equals("out"))
 if(!V.isEmpty())
 {t=ConsAct.iterator();
   while(t.hasNext())
  {  Constraint c=(Constraint)t.next();
     String var1=c.var1,var2=c.var2,val2="";

     if ((var1.equals(v))&&V.contains(var2)&&(c.type=='a'))
      {//valoarea lui var2 in Asignare->val2

          ai=Asignare.iterator();
          while(ai.hasNext())
          {VarVal vv=(VarVal)ai.next();
             String s=vv.var; String valoare=vv.val;
           if (s.equals(var2)) val2=valoare;}
```

```
     if((val2.equals("in")))
    //nodul v ataca nodul "in" var2
      attack=v;
    }}

   if(attack!=null)  //trb gasit ceva "in" ce sa atace attack=v
  {boolean gasit=false;
   t=ConsAct.iterator();
   while(t.hasNext())
     {Constraint c=(Constraint)t.next();
      String var1=c.var1,var2=c.var2,val1="";

    if ((var2.equals(v))&&V.contains(var1)&&(c.type=='a'))
     {//valoarea lui var2 in Asignare->val2

         ai=Asignare.iterator();
         while(ai.hasNext())
         {VarVal vv=(VarVal)ai.next();
            String s=vv.var; String valoare=vv.val;
          if (s.equals(var1)) val1=valoare;}

         //a doua conditie: sa nu se atace pe sine
     if((val1.equals("in"))&&(!var1.equals(attack))) gasit=true;}}

   if (!gasit) return false;}
  }}
 return true;}//end admisibil

 public List removeDuplicates(List L)
  {List rez=new ArrayList();
   Iterator it=L.iterator();
   while (it.hasNext())
   {Object o=it.next();
      if (!rez.contains(o)) rez.add(o);}
  return rez;}//end removeDuplicates

  public String traduinv(String sir)
 {if (disp.traduinv1(sir)!=null) sir=disp.traduinv1(sir);
   else sir=disp.traduinv2(sir);
  return sir;}

//initializare structuri in fnct de var curente
    public dbt(PanouAlgoritm me) throws IOException
    { med=me;sol=false;
       SolutiiV1=new ArrayList();SolutiiV2=new ArrayList();
       SolutiiDC=new ArrayList(); SolutiiAsignare=new ArrayList();
```

```
        SolutiiExplicElim=new ArrayList(); SolutiiConsAct=new ArrayList();

        Ipoteze_considerate=new ArrayList();
        Ipoteze_infirmate=new ArrayList();
        disp=new dispecer();
        disp.curatenie();disp.creaza_Tabele();}//end dbt

  public String tradu(String sir)
    {String s=disp.tradu(sir);
    return s;}

  public void elimina_infirmate(List nec_inf)
    {List infirmate =disp.elimina_infirmate(nec_inf);
     Iterator it=infirmate.iterator();
     while (it.hasNext())
     med.jTextArea2.append(disp.tradu_boli((it.next()).toString())+"\n");
     Ipoteze_infirmate.addAll(infirmate);}

      public void rezolva(List V1, List V2, List Dom_Curente, List Asignare, List
ExplicElim, List ConsAct)
    throws IOException
    {Iterator it; List VA=new ArrayList();
     VA.addAll(V1); VA.addAll(V2);

      List necesare=disp.necesare(VA);
    dbt_variabile(V1, V2, Dom_Curente,Asignare, ExplicElim,ConsAct, necesare);}

     //verifica val curenta a lui v fata de valorile var din V si intoarce prima
    //constrangere incalcata sau succes=null
    //V= lista de stringuri
 public Constraint backward_checking(List V, String v, String val,List Asignare, List
ConsAct)
 throws IOException
 {Iterator t,ai;
  List X=new ArrayList();
  //tr sa faca backward checking pt V=V1+ cele asignate deja(din Asignare)
  t=Asignare.iterator();

  while(t.hasNext())
  {VarVal vv1=(VarVal)t.next();
  X.add(vv1.var);}
  X.addAll(V); X=removeDuplicates(X);

  t=ConsAct.iterator();
  int i1=-1,i2=-1;
```

```
if(!X.isEmpty())
 while(t.hasNext())
 {  Constraint c=(Constraint)t.next();
    String var1=c.var1,var2=c.var2,val2="";

    if ((var1.equals(v))&&X.contains(var2))
     {//valoarea lui var2 in Asignare->val2
        ai=Asignare.iterator();
        while(ai.hasNext())
        {VarVal vv=(VarVal)ai.next();
           String s=vv.var; String valoare=vv.val;
         if (s.equals(var2)) val2=valoare;}
      if (c.isFalse(val, val2)) return c;}

     if ((var2.equals(v))&&X.contains(var1))
     { //valoarea lui var1 in Asignare->val2
       ai=Asignare.iterator();
        while(ai.hasNext())
        {VarVal vv=(VarVal)ai.next();
           String s=vv.var; String valoare=vv.val;
           if (s.equals(var1)) val2=valoare;}
     if (c.isFalse(val2, val)) return c;}
     }

 return null;}

 public List sterge_explic_elimin(String v, List V, List DC,List ExplicElimc)throws
IOException
 {//pt toate var din V sterge explicatiile eliminatorii care contin v si intoarce
 // valorile corespunzatoare din domeniile lor
    //V e o lista de stringuri=nume de variabile
     List ExplicElim=new ArrayList();ExplicElim.addAll(ExplicElimc);

    List l1=new ArrayList(); Set Var=new HashSet();

    for(int i=0; i<V.size(); i++) Var.add(V.get(i));
    boolean c1,c2;String nume_var;

    List ExElim=new ArrayList();Iterator it=ExplicElim.iterator();
    while(it.hasNext())
    {Explicatie e=(Explicatie)it.next();
     nume_var=e.vv.var;
     List l=e.asignare;

     Iterator it2=l.iterator();
     c1=Var.contains(nume_var); // numevar sa fie din m V
```

```
    c2=false;//sa aiba in explic elim pe v
            while(it2.hasNext())
            {VarVal v1=(VarVal)it2.next();
             if(v1.var==v){c2=true;l1.add(v1.val);}}

    if (!c1||!c2) ExElim.add(e);}//end while

 ExplicElim=ExElim;
 return ExplicElim;
}//end sterge explic elimin


//verifica daca val pt v e admisibila si o asigneaza
//V1-VAR ASIGNATE, V2-NEASIGNATE
public boolean dbt_valoare(List V1,String v, String val,List Dom_Curente,List
Asignare,List ExplicElim, List ConsAct)
throws IOException
{Constraint c=backward_checking(V1,v,val,Asignare, ConsAct);

if(c==null){ return true;}//succes
else {//System.out.println("exista constr incalcate!");
   //  System.in.read();
    List V3=c.variabile();//m de var din c;
     //V3-v
     V3.remove(v);

     //tr val var din V3 din Asignare
     //V3= m. de  var din constrangerea c incalcata de valoarea val pentru var

     ExplicElim.add(new Explicatie(v,val,V3,Asignare));
     return false;} //failure

 }//end dbtvaloare

//V1=var asignate, V2= var neasignate-liste de stringuri
 //liste de VarSet
public void dbt_variabile(List V1, List V2, List Dom_Curente, List Asignarec, List
ExplicElim, List ConsActc, List necesare)
throws IOException
{String v; Set d;
 List ConsAct, Asignare;
 Iterator it; selectie sel=new selectie(med.date_confirmate);

   if (V2.isEmpty())
    {if(!admisibil(V1,Asignarec, ConsActc))
       {float nr=0;
```

```
 System.out.print("Inadmisibil: ");

       //dc esueaza aici trebuie sa reia alg de la 0
       //: mesaj : mai testati: acoperiri incomplete
          it=Asignarec.iterator();
          while(it.hasNext())
          {VarVal vv=(VarVal)it.next();
           if(vv.val.equals("in")) {System.out.print(vv.var+" ");
            if(!vv.var.equals("SI32")) nr+=sel.suma_ponderi_simptome(vv.var);}
          }//System.out.println();

           Asignare=new ArrayList(); Asignare.addAll(Asignarec);
           ConsAct=new ArrayList(); ConsAct.addAll(ConsActc);

          if (nr>2.5)
          {System.out.print("nr"+nr);System.out.println();
           SolutiiAsignare.add(Asignare);
           SolutiiConsAct.add(ConsAct);}
          med.necesare=necesare;
       }

            if((admisibil(V1,Asignarec,
ConsActc))&&(acoperire_fara_x(Asignarec,"")==false))
            {System.out.print("Admisibil neacoperitor: ");
             StringBuffer sb=new StringBuffer();
             float nr=0;
            it=Asignarec.iterator();
            while(it.hasNext())
            {VarVal vv=(VarVal)it.next();//System.out.print(vv.var+vv.val+" ");}
             if(vv.val.equals("in"))
              {System.out.print(vv.var+" ");
              sb.append(disp.tradu_boli(vv.var)+"  ");
              if(!vv.var.equals("SI32")) nr+=sel.suma_ponderi_simptome(vv.var);
              }

              }//
              sb.append("\n");
             System.out.print("nr"+nr);System.out.println();
              if (nr>3) med.jTextArea3.append(sb.toString());
       }//end if

       if((admisibil(V1,Asignarec, ConsActc))&&(acoperire_minimala(Asignarec)==true))
       {System.out.print("Solutie: "); sol=true;
        StringBuffer sb=new StringBuffer();
        //dc esueaza aici trebuie sa reia alg de la 0
        //: mesaj : mai testati: acoperiri incomplete
```

```
        it=Asignarec.iterator();
        while(it.hasNext())
        {VarVal vv=(VarVal)it.next();//System.out.print(vv.var+vv.val+" ");}
         System.out.print(vv.var+" ");
         if(vv.val.equals("in")) sb.append(disp.tradu_boli(vv.var)+"  ");

        }  sb.append("\n");//System.out.println();
        med.jTextArea1.append(sb.toString());
     }}//end if

   else {v=(String)V2.get(0);//v= o variabila din V2
        d=new HashSet();//domeniul curent al lui v
        it=Dom_Curente.iterator();
        while (it.hasNext())
         {VarSet vs=(VarSet)it.next();
          if (vs.var.equals(v)) d=vs.multime;}

  Object  tab []=d.toArray(); boolean gasit=false;

  for(int i=0;i<tab.length;i++)
  {String val=(String)tab[i];VarVal vv=new VarVal(v,val);
      if (dbt_valoare(V1,v,val,Dom_Curente,Asignarec,ExplicElim, ConsActc)) {

                          gasit=true; Asignarec.add(vv);
                          V1.add(v);V2.remove(v);
                          dbt_variabile(V1,          V2,          Dom_Curente,          Asignarec,
ExplicElim,ConsActc,necesare);

                          it=ExplicElim.iterator();List Ee=new ArrayList();
                          while (it.hasNext())
                          {Explicatie e= (Explicatie)it.next();
                           if(!e.vv.equals(vv))Ee.add(e);}

                          ExplicElim=Ee;
                          V1.remove(v);V2.add(v); Asignarec.remove(vv);}}//end for

      if  (gasit==false)  dbt_bt_variabila(V1,  V2,  Dom_Curente,v,Asignarec,ExplicElim,
ConsActc, necesare);}//end else
}//end dbt_variabile

 public    void  dbt_bt_variabila(List  V1,  List  V2,  List  Dom_Curente,String  v,List
Asignare, List ExplicElim, List ConsAct, List necesare)
 throws IOException
 {List V3=new ArrayList(); Iterator it;

  //V3=m conflicte ale lui v= uniunea explicatiilor eliminatorii a valorilor
```

```
  //eliminate ale lui v
    it=ExplicElim.iterator();
    while(it.hasNext())
    {Explicatie ex=(Explicatie)it.next();
     if(ex.vv.var==v)
     {Iterator i=ex.asignare.iterator();
        while(i.hasNext())
        {VarVal vv=(VarVal)i.next();
        V3.add(vv.var);}}}//end while

    if (V3.isEmpty()) {JOptionPane.showMessageDialog(null,"N-am solutie, V3
empty","",
                              JOptionPane.PLAIN_MESSAGE);}

    else{String v2="";//ultima var ce apartine V3 din V1
    it=V1.iterator();
              while (it.hasNext())
             {String sir=(String)it.next();
              if (V3.contains(sir)) v2=sir;}
    //val2=valoarea curenta a lui v2
    VarVal vv2=null;String val2="";
    it=Asignare.iterator();
     while (it.hasNext())
             {vv2=(VarVal)it.next();
              if (vv2.var==v2) val2=vv2.val;}


    //V4=multimea de variabile de dupa v2 din V1 (lista de VarVal)
    List V4=new ArrayList();boolean gasit=false;
    it=V1.iterator();
              while (it.hasNext())
             {String sir=(String)it.next();
              if(gasit)V4.add(sir);
              if (sir==v2)  gasit=true;}

    //deasigneaza v2
   Asignare.remove(vv2);

        V3.remove(v2);
        ExplicElim.add(new Explicatie(v2, val2,V3,Asignare));
        //aici se modifica Dom _Curente:
        V4.addAll(V2);
        ExplicElim=sterge_explic_elimin(v2, V4, Dom_Curente,ExplicElim);
        //deasignare V2
       V2.add(v2);V1.remove(v2);}
```

```
 }//end dbt_bt_variabila
}//end class dbt1
```

//selectie.java

```
package algoritm;

import java.util.*;
import java.io.*;

class Simptom {
public String nume; float pondere;float grad_de_realizare;
public Simptom(String n, float p){nume=n;pondere=p;grad_de_realizare=0;}
}

public class selectie {
   List Manifestari;
   List Context, Infirmate;
//aici trebuie creat Contextul

   public selectie(List date)
   {Manifestari=date;}

   public  selectie (List date, List infirmate) {
   Manifestari=date;Infirmate=infirmate;
   try{creaza_Context();}
   catch (IOException e) {}
 }

   public void creaza_Context()throws IOException{
      Context=new ArrayList();

      double prag=0.2;
      if((angina_pectorala()>prag)&&(!Infirmate.contains("BI1")))
      {Context.add("BI1");
      System.out.println("angina pectorala "+angina_pectorala());
      }

      if((artrita_reumatoida()>prag)&&(!Infirmate.contains("BREUM1")))
      {Context.add("BREUM1");
     System.out.println("artrita reumatoida "+artrita_reumatoida());
      }

      if((boala_Chron()>prag)&&(!Infirmate.contains("BGI1")))
      {Context.add("BGI1");
      System.out.println("boala Chron "+boala_Chron());
      }

      if((cancer_colon_intestin()>prag)&&(!Infirmate.contains("BGI2")))
      {Context.add("BGI2");
```

```
System.out.println("cancer colon "+cancer_colon_intestin());
}

if((cancer_pancreas()>prag)&&(!Infirmate.contains("BGI3")))
{Context.add("BGI3");
System.out.println("cancer pancreas "+cancer_pancreas());
}

if((cancer_plamani()>prag)&&(!Infirmate.contains("BP1")))
{Context.add("BP1");
System.out.println("cancer plamani "+cancer_plamani());
}

if((ciroza()>prag) &&(!Infirmate.contains("BH1")))
{Context.add("BH1");
System.out.println("ciroza "+ciroza());
}

if((colita_ulceroasa()>prag) &&(!Infirmate.contains("BGI4")))
{Context.add("BGI4");
System.out.println("colita ulceroasa "+colita_ulceroasa());
}

if((diabet()>prag) &&(!Infirmate.contains("BE1")))
{Context.add("BE1");
 System.out.println("diabet "+diabet());
}

if((embolie_pulmonara()>prag)&&(!Infirmate.contains("BP2")))
{Context.add("BP2");
System.out.println("embolie pulmonara "+embolie_pulmonara());
}

if((enterita_infectioasa()>prag) &&(!Infirmate.contains("BGI5")))
{Context.add("BGI5");
 System.out.println("enterita infectioasa "+enterita_infectioasa());
}

if((glomerulonefrita_membranoasa()>prag)&&(!Infirmate.contains("BREN1")))
{Context.add("BREN1");
 System.out.println("glomerulonefrita                membranoasa
"+glomerulonefrita_membranoasa());
}

if((hepatita()>prag)&&(!Infirmate.contains("BH2")))
{Context.add("BH2");
```

```
 System.out.println("hepatita "+hepatita());
 }

 if((hipertiroidie()>prag)&&(!Infirmate.contains("BE2")))
 {Context.add("BE2");
 System.out.println("hipertiroidie "+hipertiroidie());
 }

 if((hipotiroidie()>prag) &&(!Infirmate.contains("BE3")))
 {Context.add("BE3");
System.out.println("hipotiroidie "+hipotiroidie());
 }

 if((infarct_miocardic()>prag) &&(!Infirmate.contains("BI2")))
 {Context.add("BI2");
System.out.println("infarct miocardic "+infarct_miocardic());
 }

 if((infectie_urinara()>prag)&&(!Infirmate.contains("BREN2")))
 {Context.add("BREN2");
System.out.println("infectie urinara "+infectie_urinara());
 }

 if((insuficienta_cardiaca_congestiva()>prag)&&(!Infirmate.contains("BI3")))
 {Context.add("BI3");
System.out.println("insuficienta cardiaca "+insuficienta_cardiaca_congestiva());
 }

 if((insuficienta_renala()>prag)&&(!Infirmate.contains("BREN3")))
 {Context.add("BREN3");
 System.out.println("insuficienta renala "+insuficienta_renala());
 }

 if((les()>prag)&&(!Infirmate.contains("BREUM2")))
 {Context.add("BREUM2");
 System.out.println("LES "+les());
 }

 if((litiaza_biliara()>prag)&&(!Infirmate.contains("BGI6")))
 {Context.add("BGI6");
System.out.println("litiaza biliara "+litiaza_biliara());
 }

 if((litiaza_renala()>prag)&&(!Infirmate.contains("BREN4")))
 {Context.add("BREN4");
 System.out.println("litiaza renala "+litiaza_renala());
```

```
}

if((miocardita()>prag)&&(!Infirmate.contains("BI4")))
{Context.add("BI4");
System.out.println("miocardita "+miocardita());
}

if((pancreatita()>prag)&&(!Infirmate.contains("BGI7")))
{Context.add("BGI7");
System.out.println("pancreatita "+pancreatita());
}

if((pericardita()>prag)&&(!Infirmate.contains("BI5")))
{Context.add("BI5");
System.out.println("pericardita "+pericardita());
}

if((pneumonie()>prag) &&(!Infirmate.contains("BP3")))
{Context.add("BP3");
System.out.println("pneumonie "+pneumonie());
}

if((polimiozita()>prag) &&(!Infirmate.contains("BREUM3")))
{Context.add("BREUM3");
System.out.println("polimiozita "+polimiozita());
}

if((tbc()>prag)&&(!Infirmate.contains("BP4")))
{Context.add("BP4");
System.out.println("tbc "+tbc());
}

if((toxiinfectie_alimentara()>prag)&&(!Infirmate.contains("BGI8")))
{Context.add("BGI8");
System.out.println("toxiinfectie alimentara "+toxiinfectie_alimentara());
}

if((viroza_pulmonara()>prag)&&(!Infirmate.contains("BP5")))
{Context.add("BP5");
 System.out.println("viroza "+viroza_pulmonara());
}

//IN Context AM CE S-A SELECTAT PE BAZA Manifestari=date

//adauga aici la context si cele externe!
```

```
      BufferedReader in =new BufferedReader(new
FileReader("e://programe/java/DiaMed/src/modelmedical/externe.txt"));
       String s,s2=new String(),s1=new String();

       while((s=in.readLine())!=null)
       {int n=s.lastIndexOf(" ");
        s1=s.substring(0,n).trim(); s2=s.substring(n,s.length()).trim();
        if(Context.contains(s2)) Context.add(s1);}
    };


    //ipoteze selectate pe baza datelor-> AC
    public List returneaza_Context() {return Context;}

    //max
    public float S_norma(float a, float b)
    {if (a>=b) return a;
     else return b;}

    //min
    public float T_norma(float a, float b)
    {if (a<=b) return a;
     else return b;}

    public float S_norma1(float a, float b)
    {return a+b-a*b;}

    public float T_norma1(float a, float b)
    {return a*b;}

    public float S_norma2(float a, float b)
    {if ((a+b)<1) return (a+b);
     else return 1;}

    public float T_norma2(float a, float b)
    {if ((a+b-1)>0) return (a+b-1);
     else return 0;}

    public List selectie_fisier (String nume)throws IOException
    {List Rez=new ArrayList();
      String numefis=nume;

      BufferedReader in =new BufferedReader(new FileReader(numefis));
       String s,s2=new String(),s1=new String();

       while((s=in.readLine())!=null)
```

```
        {int n=s.lastIndexOf(" ");
         s1=s.substring(0,n); s2=s.substring(n,s.length());
         Rez.add(new Simptom(s1,Float.parseFloat(s2)));}

    return Rez;}

    public float integrala_fuzzy(List l)
    {//l=lista de Simptome
        float S=0;float pondere_totala=0,gAi;
        int i,j;
       Iterator it=l.iterator();

       while(it.hasNext())
       {Simptom s=(Simptom)it.next();
        if(Manifestari.contains(s.nume)) s.grad_de_realizare=1;
        pondere_totala+=s.pondere;}

       //sortare l crescator dupa grad de realizare...

       for(i=0;i<l.size()-1;i++)
       for(j=i+1;j<l.size();j++)
          {float n1=((Simptom)l.get(i)).grad_de_realizare;
           float n2=((Simptom)l.get(j)).grad_de_realizare;
           if (n1>n2)  Collections.swap(l, i, j);}

       gAi=pondere_totala;
       for(i=0;i<l.size();i++)
       {
S=S_norma(S,T_norma(((Simptom)l.get(i)).grad_de_realizare,gAi/pondere_totala));
         gAi-=((Simptom)l.get(i)).pondere;
       }

    return S;}

    public float angina_pectorala()throws IOException{
        //Manifestari
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/angin
a_pectorala.txt");
        return integrala_fuzzy(simptome_boala);}

    public float artrita_reumatoida()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/artrita
_reumatoida.txt");
        return integrala_fuzzy(simptome_boala);}
```

```
    public float boala_Chron()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/boala
_Chron.txt");
        return integrala_fuzzy(simptome_boala);}

    public float cancer_colon_intestin()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/cance
r_colon_intestin.txt");
        return integrala_fuzzy(simptome_boala);}

    public float cancer_pancreas()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/cance
r_pancreas.txt");
        return integrala_fuzzy(simptome_boala);}

    public float cancer_plamani()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/cance
r_plamani.txt");
        return integrala_fuzzy(simptome_boala);}

    public float ciroza()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/ciroz
a.txt");
        return integrala_fuzzy(simptome_boala);}

    public float colita_ulceroasa()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/colita
_ulceroasa.txt");
        return integrala_fuzzy(simptome_boala);}

    public float diabet()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/diabe
t.txt");
        return integrala_fuzzy(simptome_boala);}

    public float embolie_pulmonara()throws IOException{
```

```
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/embo
lie_pulmonara.txt");
        return integrala_fuzzy(simptome_boala);}

    public float enterita_infectioasa()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/enteri
ta_infectioasa.txt");
        return integrala_fuzzy(simptome_boala);}

    public float glomerulonefrita_membranoasa()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/glom
erulonefrita_membranoasa.txt");
        return integrala_fuzzy(simptome_boala);}

    public float hepatita()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/hepat
ita.txt");
        return integrala_fuzzy(simptome_boala);}

    public float hipertiroidie()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/hipert
iroidie.txt");
        return integrala_fuzzy(simptome_boala);}

    public float hipotiroidie()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/hipoti
roidie.txt");
        return integrala_fuzzy(simptome_boala);}

    public float infarct_miocardic()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/infarc
t_miocardic.txt");
        return integrala_fuzzy(simptome_boala);}

    public float infectie_urinara()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/infect
ie_urinara.txt");
        return integrala_fuzzy(simptome_boala);}
```

```
    public float insuficienta_cardiaca_congestiva()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/insufi
cienta_cardiaca_congestiva.txt");
        return integrala_fuzzy(simptome_boala);}

    public float insuficienta_renala()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/insufi
cienta_renala.txt");
        return integrala_fuzzy(simptome_boala);}

    public float les()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/les.txt
");
        return integrala_fuzzy(simptome_boala);}

     public float litiaza_biliara()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/litiaz
a_biliara.txt");
        return integrala_fuzzy(simptome_boala);}

    public float litiaza_renala()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/litiaz
a_renala.txt");
        return integrala_fuzzy(simptome_boala);}

    public float miocardita()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/mioc
ardita.txt");
        return integrala_fuzzy(simptome_boala);}

    public float pancreatita()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/pancr
eatita.txt");
        return integrala_fuzzy(simptome_boala);}

    public float pericardita()throws IOException{
```

```
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/pericardita.txt");
        return integrala_fuzzy(simptome_boala);}

    public float pneumonie()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/pneumonie.txt");
        return integrala_fuzzy(simptome_boala);}

    public float polimiozita()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/polimiozita.txt");
        return integrala_fuzzy(simptome_boala);}

    public float tbc()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/tbc.txt");
        return integrala_fuzzy(simptome_boala);}

    public float toxiinfectie_alimentara()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/toxiinfectie_alimentara.txt");
        return integrala_fuzzy(simptome_boala);}

    public float viroza_pulmonara()throws IOException{
        List
simptome_boala=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/viroza_pulmonara.txt");
        return integrala_fuzzy(simptome_boala);}


    public float suma_ponderi_simptome(String nume_boala)throws IOException
    {//l=lista de Simptome
        float S=0; float total=0;
        int i,j; List l=null;

        if(nume_boala.equals("BI1"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/angina_pectorala.txt");
        if(nume_boala.equals("BREUM1"))
```

```
l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/artrita_reumatoida.txt
");
        if(nume_boala.equals("BGI1"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/boala_Chron.txt");
        if(nume_boala.equals("BGI2"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/cancer_colon_intesti
n.txt");
        if(nume_boala.equals("BGI3"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/cancer_pancreas.txt")
;
        if(nume_boala.equals("BP1"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/cancer_plamani.txt");
        if(nume_boala.equals("BH1"))
         l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/ciroza.txt");
        if(nume_boala.equals("BGI4"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/colita_ulceroasa.txt")
;
        if(nume_boala.equals("BE1"))
         l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/diabet.txt");
        if(nume_boala.equals("BP2"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/embolie_pulmonara.t
xt");
        if(nume_boala.equals("BGI5"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/enterita_infectioasa.t
xt");
        if(nume_boala.equals("BREN1"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/glomerulonefrita_me
mbranoasa.txt");
        if(nume_boala.equals("BH2"))
         l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/hepatita.txt");
        if(nume_boala.equals("BE2"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/hipertiroidie.txt");
        if(nume_boala.equals("BE3"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/hipotiroidie.txt");
        if(nume_boala.equals("BI2"))
```

```
l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/infarct_miocardic.txt
");
        if(nume_boala.equals("BREN2"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/infectie_urinara.txt");
        if(nume_boala.equals("BI3"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/insuficienta_cardiaca
_congestiva.txt");
        if(nume_boala.equals("BREN3"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/insuficienta_renala.tx
t");
        if(nume_boala.equals("BREUM2"))
         l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/les.txt");
        if(nume_boala.equals("BGI6"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/litiaza_biliara.txt");
        if(nume_boala.equals("BREN4"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/litiaza_renala.txt");
        if(nume_boala.equals("BI4"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/miocardita.txt");
        if(nume_boala.equals("BGI7"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/pancreatita.txt");
        if(nume_boala.equals("BI5"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/pericardita.txt");
        if(nume_boala.equals("BP3"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/pneumonie.txt");
        if(nume_boala.equals("BREUM3"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/polimiozita.txt");
        if(nume_boala.equals("BP4"))
         l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/tbc.txt");
        if(nume_boala.equals("BGI8"))

l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/toxiinfectie_alimenta
ra.txt");
        if(nume_boala.equals("BP5"))
```

```
l=selectie_fisier("e://programe/java/DiaMed/src/modelmedical/boli/viroza_pulmonara.txt
");

      Iterator it=l.iterator();
      while(it.hasNext())
      {Simptom s=(Simptom)it.next();
       if(Manifestari.contains(s.nume)) total+=s.pondere;}
  return total;}


}
```

//dispecer.java

```
package bd;

import algoritm.*;
import java.sql.*;
import java.util.*;
import java.io.*;

public class dispecer {
     Statement cmdSQL=null;Connection conn=null;
     String mesaj=null;List Context;

    public dispecer() {conectareBD();}

  public List removeDuplicates(List L)
  {List rez=new ArrayList();
   Iterator it=L.iterator();
   while (it.hasNext())
   {Object o=it.next();
       if (!rez.contains(o)) rez.add(o);}
  return rez;}


    public void curatenie()
    {try{
     cmdSQL.execute("drop table NODURI");cmdSQL.execute("drop table
CONSTRANGERI");
     cmdSQL.execute("drop table DICTIONARS");cmdSQL.execute("drop table
DICTIONARI");
     cmdSQL.execute("drop table DICTIONARBOLI");}
     catch(SQLException e) {System.out.println("Exceptie la stergere tabele."+e);}
    }

    public String getMesaj()
    {return mesaj;}

    void conectareBD()
    {try {Class.forName("com.mysql.jdbc.Driver").newInstance();
     conn=DriverManager.getConnection("jdbc:mysql:///diagnoza","","");
     cmdSQL=conn.createStatement();
     }
    catch(Exception e) {System.out.println("Eroare la conectarea la BD diagnoza:"+e);
                    System.exit(1);}
    }//end conectare BD
```

```
   //creaza tabele din fisiere NODURI, RELATII, DICTIONAR
   public void creaza_Tabele(){

   try{
   cmdSQL.execute("create table NODURI(ID int, NUME_NOD varchar(20),
CONTEXT varchar(40), TIP char, PONDERE float)");
   cmdSQL.execute("create table CONSTRANGERI(ID int, NUMEC varchar(10),
NOD1 varchar(20), NOD2 varchar(20),"+
       "CONTEXT varchar(30), TIP char)");//numec -> determina unic si contextul

   cmdSQL.execute("create table DICTIONARS(ID int, NUME_NOD
varchar(20),DEN_MED varchar(80))");
   cmdSQL.execute("create table DICTIONARBOLI(ID int, NUME_NOD
varchar(20),DEN_MED varchar(80))");
   cmdSQL.execute("create table DICTIONARI(ID int, NUME_NOD
varchar(20),DEN_MED varchar(80))");
   BufferedReader in;
   String numefis="e://programe/java/DiaMed/src/modelmedical/noduri.txt";
   in=new BufferedReader(new FileReader(numefis)) ;
        String s=new String();int id; String nume_nod,context;char tip;double pondere;
        StringTokenizer st;
        while((s=in.readLine())!=null)
          {st=new StringTokenizer(s," ");
           id=Integer.parseInt(st.nextToken());
           nume_nod=st.nextToken();context=st.nextToken();
           tip=st.nextToken().charAt(0);
           pondere=Double.parseDouble(st.nextToken());

           try{
           cmdSQL.executeUpdate("insert into NODURI
values("+id+",'"+nume_nod+"','"+context
             +"','"+tip+"',"+pondere+")");}
           catch (SQLException e){System.out.println("eroare la inserare"+e);}
           }//end while


  numefis="e://programe/java/DiaMed/src/modelmedical/constrangeri.txt";
   in=new BufferedReader(new FileReader(numefis)) ;
        s=new String(); String nod1,nod2,numec;
        while((s=in.readLine())!=null)
          {st=new StringTokenizer(s," ");
           id=Integer.parseInt(st.nextToken());
           numec=st.nextToken();
           nod1=st.nextToken();nod2=st.nextToken();context=st.nextToken();
           tip=st.nextToken().charAt(0);
```

```
          cmdSQL.executeUpdate("insert into CONSTRANGERI
values("+id+",'"+numec+"','"+nod1+"','"+nod2+"','"
                          +context+"','"+tip+"')");
          }//end while


  numefis="e://programe/java/DiaMed/src/modelmedical/dictionars.txt";
   in=new BufferedReader(new FileReader(numefis)) ;
        s=new String(); String den_med;
        while((s=in.readLine())!=null)
        {st=new StringTokenizer(s," ");
          id=Integer.parseInt(st.nextToken());
          nume_nod=st.nextToken();den_med=st.nextToken();

          cmdSQL.executeUpdate("insert into DICTIONARS
values("+id+",'"+nume_nod+"','"+den_med+"')");
          }//end while


   numefis="e://programe/java/DiaMed/src/modelmedical/dictionari.txt";
   in=new BufferedReader(new FileReader(numefis)) ;
        s=new String();
        while((s=in.readLine())!=null)
          {//System.out.println(s);
          st=new StringTokenizer(s," ");
          id=Integer.parseInt(st.nextToken());
          nume_nod=st.nextToken();den_med=st.nextToken();

          cmdSQL.executeUpdate("insert into DICTIONARI
values("+id+",'"+nume_nod+"','"+den_med+"')");
          }//end while


        numefis="e://programe/java/DiaMed/src/modelmedical/dictionarboli.txt";
   in=new BufferedReader(new FileReader(numefis)) ;
        s=new String();
        while((s=in.readLine())!=null)
        {st=new StringTokenizer(s," ");
          id=Integer.parseInt(st.nextToken());
          nume_nod=st.nextToken();den_med=st.nextToken();

          cmdSQL.executeUpdate("insert into DICTIONARBOLI
values("+id+",'"+nume_nod+"','"+den_med+"')");
          }//end while
```

```
  }//end try
  catch(Exception ex){System.out.println("Exceptie la crearea tabelelor: "+ex);}
}//end creaza_tabele



  public List creaza_Context(List date, List infirmate){
     //infirmate= ipoteze infirmate
     System.out.println("Context");
     List l=new ArrayList();
     selectie sel=new selectie(date, infirmate);
     Context=sel.returneaza_Context();
     //Context=ipotezele curente;
     //selectia tine cont de ponderea absoluta a simptomului
     Iterator it=Context.iterator();
     while(it.hasNext())
     {String s=it.next().toString();
      System.out.println(s);
      if (!infirmate.contains(s))l.add(s);}
// return ipoteze noi= selectate-infirmate; (selectate=Context)
     l=removeDuplicates(l);
    return l;}



  ////
  public List creaza_VarActive(){
  //aici creaza DC
     //algoritm de selectie Var_Active pe baza Contextului+ initializare dom_curente
     //=toate simptomele care apar cu campul context= o boala din Contextul actual

     List VA=new ArrayList(); Iterator it=Context.iterator();
     Set d=new HashSet(); d.add("in"); d.add("out");

     while(it.hasNext())
     {String s=it.next().toString();
      VA.add(new VarSet(s,d));}

         List rez=new ArrayList();

         for(int i=0; i<VA.size(); i++)
         {VarSet o=(VarSet)VA.get(i);
           if (!o.isContained(rez)) rez.add(o);}
         VA= rez;

  return VA;}
```

```
    public List creaza_ConstrActive(List VA){
    //algoritm de selectie Constr_Active pe baza Contextului
        //=toate constrangerile in care apar Var_active
        Factory fact=new Factory();
        Constraint c;
        String numec, var1, var2; char tip;
        List CA=new ArrayList(); Iterator it=VA.iterator();

        try
        {while(it.hasNext())
         {String s=(String)it.next();
          ResultSet rs=null;
          if(!s.equals(""))
          {rs=cmdSQL.executeQuery("select * from CONSTRANGERI where
(CONTEXT='"+s+"')");}
            while(rs.next())
              {numec=rs.getString(2);var1=rs.getString(3);var2=rs.getString(4);
               tip=rs.getString(6).charAt(0);
               c=fact.getCons(numec,tip,var1,var2);
                 if(!CA.contains(c)) CA.add(c);}
            }//end while
          }//end try
        catch(SQLException e){System.out.println("Exceptie la crearea Constr
Active:"+e);}

          List rez=new ArrayList();

            for(int i=0; i<CA.size(); i++)
            {Constraint o=(Constraint)CA.get(i);
               if (!o.isContained(rez)) rez.add(o);}
             CA=rez;

        return CA;
}//end creaza_ConstrActive

 //trebuie sterse Constr_active cu constr din Context si adaugate la RC
public List creazaRC(String contex, List Constr_Active)
{List consRC=new ArrayList();
 Iterator it=Constr_Active.iterator();
 try{
 while(it.hasNext())
  {Constraint c= (Constraint)it.next();
   ResultSet rs=cmdSQL.executeQuery("select CONTEXT from CONSTRANGERI
where '"+ c.name+"'=NUMEC");
```

```
   while(rs.next()){if(contex.equals(rs.getString(1)))
                consRC.add(c);}
 }}
 catch (SQLException e) {System.out.println("Eroare la stergerea de constrangeri");}
 consRC=removeDuplicates(consRC);
return consRC;
}//end creazaRC

//sterge Constr_Active //  contex=context infirmat
public List stergeCA(String contex, List Constr_Active)
{List L=Constr_Active;
 Iterator it=Constr_Active.iterator();
try{
 while(it.hasNext())
  {Constraint c= (Constraint)it.next();
   ResultSet rs=cmdSQL.executeQuery("select CONTEXT from CONSTRANGERI
where'"+ c.name+"'=NUMEC");
   while(rs.next()){if(contex.equals(rs.getString(1)))
                L.remove(c);}
 }}

catch (SQLException e) {System.out.println("Eroare la stergerea de constrangeri
active");}
return L;}

 public List necesare(List varact)
 {List nec=new ArrayList();
 Iterator it=varact.iterator();
 try{
 while(it.hasNext())
  {String numevar= (String)it.next();
   ResultSet rs=cmdSQL.executeQuery("select NUME_NOD from NODURI where ('"+
numevar+"'= CONTEXT) and (TIP='n')");
   while(rs.next()){nec.add(rs.getString(1));}
 }}
 catch (SQLException e) {System.out.println("Eroare la necesare");}
 nec=removeDuplicates(nec);
return nec;}

  public String tradu_boli(String sir)
    {String s=null;ResultSet rs=null;
     if(!sir.equals("SI32"))
      try
        {rs=cmdSQL.executeQuery("select DEN_MED from DICTIONARBOLI where '"+
sir+"'= NUME_NOD");
          rs.next(); s=rs.getString(1);}
```

```
     catch (SQLException e) {System.out.println("Eroare la traducere"+e);}
    else s="presiune/volum cardiac normal";
 return s.trim();
 }

public String tradu(String sir)
 {String s=null;ResultSet rs=null;
     try{
        if(sir.charAt(1)=='S')
        {rs=cmdSQL.executeQuery("select DEN_MED from DICTIONARS where '"+
sir+"'= NUME_NOD");
         rs.next(); s=rs.getString(1);}
   else {rs=cmdSQL.executeQuery("select DEN_MED from DICTIONARI where '"+
sir+"'= NUME_NOD");
        rs.next(); s=rs.getString(1);}
     }
 catch (SQLException e) {System.out.println("Eroare la traducere"+e);}
 return s.trim();}

  public String traduinv1(String sir)
 {String s=null;
    try{
    ResultSet rs=cmdSQL.executeQuery("select NUME_NOD from DICTIONARS
where '" + sir+"'= DEN_MED");
    while (rs.next()){ s=rs.getString(1);s.trim();}
    }
    catch (SQLException e) {System.out.println("Eroare la traducereinv1"+e);}
  return s;}

  public String traduinv2(String sir)
 {String s=null;
    try{
    ResultSet rs=cmdSQL.executeQuery("select NUME_NOD from DICTIONARI where
'" + sir+"'= DEN_MED");
    while (rs.next()){ s=rs.getString(1);s.trim();}
    }
    catch (SQLException e) {System.out.println("Eroare la traducereinv2"+e);}
  return s;}

  public List elimina_infirmate(List n_inf)
 {//intoarce lista de boli in care apar simpt din n_inf ca necesare
    System.out.println("elimina infirmate");
  List elim=new ArrayList();
 Iterator it=n_inf.iterator();
 try{
```

```
 while(it.hasNext())
  {String numevar= (String)it.next();
   //tradu in nume nod din den med
    System.out.println("contexte: infrmate");
   if (traduinv1(numevar)!=null) numevar=traduinv1(numevar);
   else numevar=traduinv2(numevar);
   ResultSet rs=cmdSQL.executeQuery("select CONTEXT from NODURI where ('"+
numevar+"'= NUME_NOD) and (TIP='n')");
   while(rs.next())elim.add(rs.getString(1));
 }}
 catch (SQLException e) {System.out.println("Eroare la necesare");}

 elim=removeDuplicates(elim);
return elim;}

  public List simpt(String boala)
 {List lista=new ArrayList(),lista2=new ArrayList();
  String s;

   try{
  ResultSet rs=cmdSQL.executeQuery("select NUME_NOD from NODURI where
('"+boala+"'= CONTEXT)");
   while(rs.next()) lista.add(rs.getString(1));  }
 catch (SQLException e) {System.out.println("Eroare la simpt"+e);}

  Iterator it=lista.iterator();
  while(it.hasNext())
  {s=it.next().toString();
   s=tradu(s);lista2.add(s);}

  lista=removeDuplicates(lista2);
 return lista;}
}
```