atea "Politehnica" din Timişoara
artamentul de Calculatoare

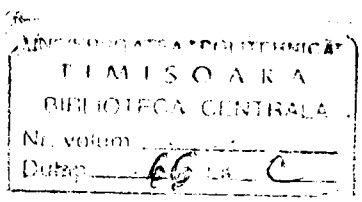
Nicolae Szirbik

CONTRIBUŢII LA METODELE DE INTEGRARE SIMBOLIC-CONEXIONISTE CU APLICAŢII ÎN SISTEMELE DE RAŢIONAMENT COMPOZIŢIONALE

conducător ştiinţific:
Prof.dr.ing. Ioan Jurca

-1998-

Mamei,



CUPRINS

Partea I
Motivaţia acestei cercetări, suportul teoretic existent, soluţiile propuse

Introducere

În timpul scurtei sale istorii, domeniul de cercetare numit inteligenţă artificială a fost arena unei dispute, uneori exagerate, între două arii de cercetare, doi *fréres ennemis*, pe care îi putem denota "simbolicism" şi "conexionism" [Hilario96]. Inteligenţa artificială îşi propune dezvoltarea unor modele care pot fi aplicate la construirea unor sisteme fizice capabile să manifeste însuşiri care pot fi considerate inteligente. Pentru că nu există o definiţie foarte clară a conceptului de inteligenţă, singurul procedeu care poate releva "inteligenţa" unui sistem artificial este testul Turing (1950), care implică o comparaţie cu o entitate inteligentă umană. Este important de observat că acest test se bazează pe analiza pur comportamentală a sistemului observat şi nu pe o analiză formală a mecanismelor interne care guvernează funcţionarea lui. Explicaţia disputei amintite poate fi şi o înţelegere diferită a conceptului de inteligenţă.

Există două şcoli în domeniul inteligenţei artificiale, care au păreri diferite în privinţa stabilirii "inteligenţei" unui sistem. Prima, numită şi "clasică", admite doar modele matematice care pot fi demonstrate că produc rezultate corecte în toate circumstanţele. Această şcoală, derivată din grupul de la Darthmouth (vezi istoria grupului prezentată în [Newell&Simon72]), formată din matematicieni cu o puternică pregătire formală, urmaşi ai lui Alan Turing şi John Von Neumann, îşi propunea nici mai mult nici mai puţin, decât realizarea unor maşini inteligente (în sensul curent al conceptului de inteligenţă) care să poată întrece omul în performaţele sale cognitive. Acesta era deja parţial întrecut de maşinile de calcul ale vremii respective, care puteau rezolva anumite probleme, dar doar pe cele care puteau reduce metoda aplicată la un algoritm format dintr-un set de operaţii de bază. Dificultăţile legate de complexitatea sistemelor formale necesare pentru reprezentarea unor probleme mai apropiate de "inteligenţă" şi probleme legate de intensitatea computaţională a algoritmilor exhaustivi de căutare folosiţi, au făcut ca această abordare să nu aibă rezultate de succes, chiar dacă dezvoltarea sistemelor fizice de calcul (hardware) a fost deosebită în perioada scursă din 1956 până astăzi. Sunt însă de apreciat realizările colaterale ale acestui tip

de cercetare cum ar fi: programarea funcţională, programarea logică, meta-euristicile, logicile non-monotonice, modelele clasice pentru sisteme expert, modelele pentru învăţarea automată (machine-learning). A doua şcoală este bazată pe ideea testului Turing şi se numeşte "comportamentalistă" (behaviorist) [McCulloch88]. Ea stabileşte că un sistem este inteligent doar din analiza exerioară a funcţionării sistemului. Dacă prin testare (cât de exhaustivă posibil) sistemul dă rezultate corecte, şi comportamentul său se apropie în anumite instanţe de cel inteligent uman, atunci sistemul poate fi acceptat, indiferent de mecanismele sale interioare. Se poate observa că lipsa demonstrării unei funcţionări invariabil corecte este slăbiciunea acestui tip de abordare. Cele două abordări sunt oarecum extreme, una concentrându-se pe întrebarea CUM se poate simula inteligenţa, iar a doua pe CE anume trebuie sistemul să poată să facă.

După părerea mea, insuccesul în realizarea unui model matematic al sistemelor inteligente este în primul rând legat de lipsa unei definiţii clare, matematice a inteligenţei. Există o mulţime de definiţii, dar niciuna nu este formală. Definiţia mea preferată este următoarea [Clark89]:

> Inteligenţa este tocmai acel *sumum* de însuşiri umane al căror conţinut nu este exprimat formal. Orice însuşire umană, care poate fi exprimată algoritmic este rutinieră. Dacă o însuşire este formalizată la un moment dat, ea părăseşte grupul însuşirilor care formează inteligenţa şi devine rutină.

Această definiţie statuează inteligenţa artificială ca fiind un concept elastic, care se modifică în timp, odată cu găsirea de către om, **prin metode creative, inteligente** a unor explicaţii pentru propriile sale însuşiri. Odată cu aceste explicaţii, omul poate realiza şi simularea acestor activităţi, care până în momentul respectiv au fost considerate inteligente. Şi această definiţie este discutabilă, pentru că am văzut cum maşina a învins omul într-un joc despre care era unanim acceptat că necesită "inteligenţă". Conform definiţiei, şahul nu ar mai reprezenta o activitate inteligentă, lucru care în mod cert nu este adevărat. Dacă vor putea fi construite modele şi maşini care să simuleze creativitatea, intuiţia, dialogul inter-uman, ar însemna tot conform definiţiei că inteligenţa ca şi concept ar fi un capitol epuizat, dat fiind faptul că toate manifestările ei sunt exprimabile formal. Dar trebuie să fim atenţi, definiţia are legătură cu conceptul de inteligenţă artificială. Până în perioada anilor 60, capacităţile de rezolvare a unor probleme simbolice erau considerate atribute de inteligenţă, dar odată cu găsirea metodelor matematice, acestea nu au mai fost considerate ca fiind activităţi "inteligente" **pentru calculator**. Pentru om însă, aceste capabilităţi vor rămâne întotdeauna un atribut al inteligenţei, aşa cum şi calităţile de şahist vor indica calificativul de "om inteligent". De remarcat este că metodele matematice folosite de programatorii maşinilor de jucat şah sunt bazate şi pe o serie de metode euristice, mai ales la estimarea valorii poziţiei, lucru care nu duce la găsirea celei mai avantajoase mutări, sau per ansamblul partidei, la cea mai scurtă victorie. Din acest motiv, este exagerat să afirmăm că jocul de şah este complet formalizat. Pur şi simplu, forţa brută a maşinii a devenit comparabilă cu forţa complet diferită ca natură a minţii umane,

despre care trebuie să amintim că ocupă un spaţiu de aproximativ 1500 cm cubi şi consumă în jur de 20 waţi. Chiar dacă maşinile vor avea în viitor dimensiuni si consumuri mai mici, mai rămâne de demonstrat formal, că algoritmii pe care acestea le folosesc, duc la cea mai bună soluţie. Interesant este faptul că discuţiile critice care se duc despre conceptul de inteligenţă artificială eşuează totdeauna în spaţiul filozofic, existând multe păreri diferite în ceea ce priveşte conţinutul acestui domeniu.

Revenind la cele două şcoli, se poate observa că şi modelele diferă. În timp ce şcoala clasică folosea în special metodele logicii matematice, ale sistemelor formale şi ale limbajelor formale, şcoala behavioristă folosea mai mult metode ad-hoc, rezultate din observaţii făcute asupra comportamentului uman şi a sistemului biologic (neuro-fiziologic) implicat în activităţile cognitive. Ajungând la nivelul anilor '60, s-a putut observa o cristalizare a metodelor folosite, şcoala clasică adoptând procesarea simbolică bazată pe formalismul matematic iar şcoala behavioristă folosind cu precădere un instrument matematic numit "reţea neuronală artificială" [Hecht90] [Hertz&al91] [Haykin94]. Din păcate, urmaşele cele două şcoli, devenite între timp arii distincte de cercetare denumite "simbolicism" şi "conexionism", au avut o evoluţie interesantă din punctul de vedere al concurenţei ştiinţifice. Merită a fi menţionată cartea "The Perceptrons" [Minsky&Papert69] care stabilea formal şi irefutabil limitările reţelei neuronale feedforward cu un singur strat. Această lucrare a oprit finanţarea cercetărilor din domeniu pentru aproape 20 de ani, deşi paradoxal, în acelaşi an Bryson şi Ho descopereau (întâmplător şi în alt domeniu) algoritmul care se numeşte astăzi "learning by error backpropagation", al cărui redescoperire în '85 (separat de către Parker, Le Cun, Rumelhart&al - plus Werbos in '74) a făcut ca reţelele neuronale să fie din nou o arie de cercetare cu finanţare bogată.

O altă "lovitură" din direcţia "simbolicistă" este lucrarea lui Fodor şi Pylyshin din 1988 [Fodor&Pylyshyn88] care argumentează că metodele conexioniste nu pot să fie un suport al unei teorii care să modeleze capacităţile cognitive, pentru că nu pot să ţină cont de complexitatea combinatorială a structurilor sintactice şi semantice ale reprezentărilor mentale. În cel mai bun caz, în opinia celor doi autori, conexionismul este doar o nouă metodă de implementare a acestor reprezentări şi a procesării lor, doar o alternativă aproximativă şi nesigură la metodele simbolice folosite de şcoala clasică. Contralovituri există şi din direcţia grupului de cercetători conexionişti, de remarcat fiind una cât se poate de "extremistă" a lui Pinker şi Prince, tot în 1988 [Pinker&Prince88], care susţine că teoriile simbolice nu sunt altceva decât aproximări grosolane şi foarte incomplete ale fenomenelor care au loc în mintea umană. Există şi o a treia părere, observabilă în ambele tabere, că cele două abordări sunt potrivite fiecare pentru un anumit set distinct de probleme şi că nu este necesar amestecul lor (direcţie numită eliminativism [Smolensky88]).

Între aceste poziţii, oarecum radicale, au apărut la sfârştiul anilor '80 [Reeke&Edelman88] [Touretzky&Hinton88] [Hendler89] [Goodman&al89], lucrări care se plasau la interfaţa dintre procesarea simbolică şi conexionistă. Originea lor este legată de încercările de a realiza anumite aspecte ale procesării simbolice cu metode

specific conexioniste şi de a realiza legătura între reţele neuronale cu rol de senzor sau rol motor cu un nivel de decizie superior, realizat prin metode simbolice (în special în domeniul conducerii roboţilor). Această nouă arie de cercetare a fost denumită *integrare neuro-simbolică* (neuro-symbolic integration) [Sun91]. Se pot identifica şi aici două direcţii de lucru: una *unificată* şi una *hibridă*. Prima încearcă să implementeze toate operaţiile dintr-un sistem inteligent prin reţele neuronale, pe când a doua încearcă să combine cele două paradigme, punând de exemplu în acelaşi sistem, reţele neuronale şi grafuri de decizie. Merită a fi remarcat că Marvin Minsky, realizatorul lui SNERT (1954), primul "calculator neuronal" şi adversarul ideilor conexioniste în perioada '65-'75, este unul din cei mai importanţi susţinători ai cercetării în domeniul sistemelor hibride [Minsky86] [Minsky91].

Această teză arată calea strabătută pentru identificarea unei metodologii de realizare a unor sisteme de raţionament automat hibride, simbolic-conexioniste.

11. Strategii şi metodologii

Integrarea simbolic-conexionistă este motivată de natura bivalentă a capacităţilor cognitive ale minţii umane. S-a observat că procesele mentale pot fi clasificate în diverse moduri, toate aceste procese luate în considerare având un rol important în obţinerea unui comportament inteligent. Un mod de clasificare este de a grupa procesele în două tipuri: *sintetice* şi *analitice*. Clasificarea nu se poate face în mod rigid, majoritatea proceselor având şi o componentă analitică şi una sintetică, într-o măsură mai mică sau mai mare. Procesele pot fi dispuse pe o axă limitată la capete, care să conţină la o extremitate procesele pur sinetice, iar la cealaltă extremitate pe cele pur analitice.

Procesele sintetice sunt de regulă asociate cu operaţii de nivel scăzut, care nu pot fi descompuse în operaţii mai simple, cu caracteristică de non-conştienţă (omul nu le execută în mod voit). Tipice sunt operaţiile de percepţie, cum sunt vederea şi recunoaşterea vorbirii (a cuvintelor). Ele sunt folosite de procese superioare, cum sunt identificarea unor obiecte din scena văzută, sau extragerea unui conţinut de idei din fluxul verbal ascultat. Ceea ce este cel mai caracteristic, cunoştinţele implicate în acest tip de operaţii sunt non-propoziţionale şi este aproape imposibil să le exprimăm formal, pentru că ele sunt mai degrabă ilustrative.

În contrast, procesele analitice sunt asociate cu operaţii conştiente, care pot fi descompuse în operaţii mai simple. Exemple: rezolvarea problemelor, luarea unor decizii în mod conştient, generarea unui text, dialogul (la un nivel propoziţional). Cunoştinţele implicate folosite au o exprimare propoziţională, legată de un limbaj, care poate fi limbajul natural, sau o serie de limbaje artificiale, mai mult sau mai puţin formale. [Newell90] oferă o clasificare graduală (pe scara complexităţii analitice) a acestor procese, folosind ca factor de discriminare, timpul necesar pentru efectuarea acestor operaţii. Ceea ce rezultă foarte clar este că majoritatea proceselor nu sunt pur

sintetice sau pur analitice. Un exemplu tipic este calculul mental: 4 înmulţit cu 3 este pur sintetic (pentru cine ştie pe de rost tabla lui Pitagora), pe când 34 înmulţit cu 13 este analitic în mare măsură (doar operaţiile obţinute prin secvenţierea şi descompunerea necesară sunt sintetice). Dar cum sunt 12 înmulţit cu 7 sau 5 înmulţit cu 15? Posibil ca undeva între cele două extreme.

Un caz interesant este învăţarea cântatului la un instrument. Un începător la pian trebuie să fie conştient de fiecare mişcare pe care o face, cunoştinţele sale sunt explicite, pe când un pianist mai experimentat este mai puţin conştient, concentrându-se mai mult asupra întregului şi fluenţei, pe când pentru profesionişti, cântatul în sine este un automatism, ei fiind mai mult conştienţi de nuanţele şi tipul de interpretare pe care o doresc. Se poate observa cum experienţa contribuie la deplasarea aceleiaşi operaţii executate pe axa imaginată, de la analitic la sintetic. Sunt însă şi operaţii care implică mai multe procese mentale în acelaşi timp, procese care pot fi plasate pe porţiuni foarte diferite ale axei. Operaţia mentală de a recunoaşte un membru al familiei după voce este pur sintetică şi fără nevoia de conştientizare. Operaţia mentală de a decide dacă iau umbrela colegului de birou implică o analiză conştientă a implicaţiilor şi consecinţelor posibile. Aceste două operaţii sunt la extremele axei, dar să ne imaginăm ce fel de procese sunt implicate în timpul în care cineva conduce o maşină în trafic intens (vezi figura).

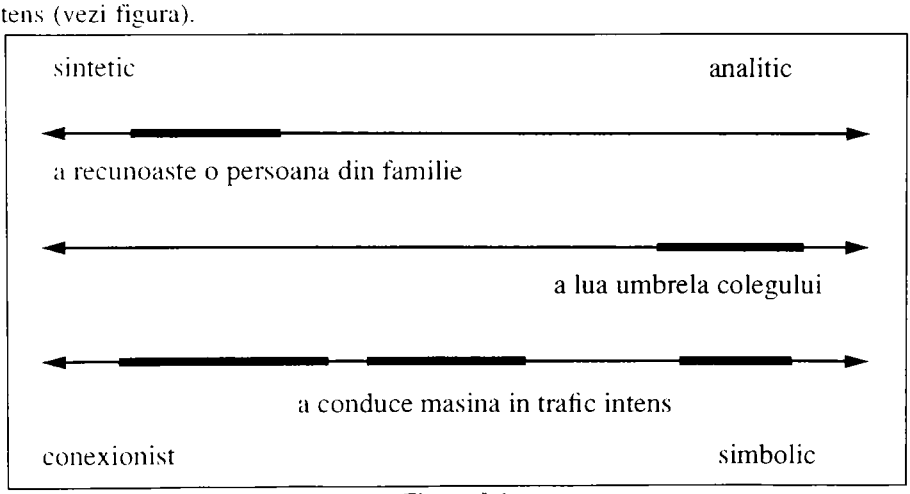


Figura I-1.

Este cât se poate de limpede că simularea proceselor mentale sintetice se poate face cu succes mai ale cu metode conexioniste, iar cele analitice cu metode simbolice. Se poate imagina o axă paralelă cu cea a proceselor mentale, dedicată metodelor de realizarea artificială folosite. La extremitatea sintetică să va oglindi extremitatea algoritmilor pur numerici ai reţelelor neuronale, iar la extremitatea analitică, va apare o analogie cu procesarea simbolică. Aici trebuie făcută observaţia crucială: **modelele oferite de domeniul inteligenţei artificiale sunt la cele două extremităţi ale axei**

metodelor folosite. Nu există la ora actuală nici un fel de modele bine aprofundate, de succes, care să se afle într-un punct al axei isomorf cu punctul unde procesele mentale sunt în aceiaşi măsură sintetice şi analitice (deşi astfel de procese există, mai mult, ele reprezentând majoritatea operaţiilor "inteligente" efectuate de om [Gutknecht&al91]).

Un exemplu al unui astfel de proces mental este stabilirea unui diagnostic medical de către un specialist. Identificarea unui model matematic corespunzător este necesară pentru transpunerea acestei operaţii într-un sistem expert. Se pot folosi mai multe metode, de exemplu inferenţa Bayesiană [Todd91] care are dezavantajul că cunoştinţele propoziţionale nu pot fi reprezentate, sau logica fuzzy, care are dezavantajul că sistemul nu poate să fie antrenat prin consultarea unei populaţii statistice de cazuri medicale (vezi sistemele fuzzy CaDiag şi FAULT [Leung&Lam89]). Folosirea modelelor computaţionale conexioniste se poate face pe două căi. Fie se gândeşte tot sistemul ca fiind conexionist, în acest caz reprezentarea şi procesarea cunoştinţelor propoziţionale facându-se prin metode conexioniste, fie se implementează o parte a sistemului prin metode simbolice, o altă parte prin metode conexioniste, accentul căzând în acest caz pe comunicarea dintre cele două părţi. Putem vorbi de o strategie unificată sau de una hibridă.

Strategia unificată se bazează pe ideea că în astfel de sisteme inteligente, nu este nevoie de structuri şi reprezentări simbolice precum şi de procesare simbolică explicită. Există o explicaţie pentru această presupunere, cei care o propun sunt de regulă neurobiologi, foarte îndepărtaţi de matematica simbolică şi ei caută de regulă modele care au o identificare cât mai apropiată de sistemele neuronale naturale (mai ales că ei studiază structuri primitive relativ la operaţiile analitice - orientarea şobolanilor de exemplu). Există bineînţeles şi argumentul că procesarea cea mai analitică posibil a simbolurilor la nivelul minţii umane se realizează tot prin reţele neuronale. O analiză mai detaliată a acestei probleme este prezentată în subcapitolul 4.3 al tezei.

Un caz special al strategiei unificate este **procesarea conexionistă a simbolurilor** (CSP - Connexionist Symbolic Procesing). Aceasta nu trebuie confundată cu **procesarea neuronală a simbolurilor** (NSP - Neuronal Symbolic Processing) [Edelman92], a cărui obiectiv este identificarea unor modele de neuroni cât mai apropiaţi de neuronii naturali şi care prin funcţionarea lor colectivă pot emula operaţii de tip analitic, cu caracter simbolic. CSP este o cercetare care încearcă realizarea unor operaţii de regulă simbolice (raţionamentul automat, planificarea, analiza sintactică) prin metode conexioniste unitare (un singur tip de reţea neronală, pe cât posibil mai simplă). Exemple de modele reuşite sunt: reprezentările şi clasificarea conceptelor [Feldman&Ballard82] [Shastri88], procesarea unor liste prin primitive gen LISP cu sistemul BOLTZCONS [Touretzky&Hinton88], extragerea unor proprietăţi prin reprezentarea ierarhică a cunoştinţelor [Sun91]. S-au obţinut unele rezultate şi in domeniul inferenţei logice [Hölldobler&Kurfess91] [Towell91], analiza limbajului natural [Bookman87] [Dyer91] şi implementarea conexionistă a sistemelor expert [Gallant88] [Dragomirescu&al93b].

I2. Viabilitatea strategiei hibride

Rostul implementatorului de sisteme inteligente (sisteme expert, sistem de conducere a roboţilor, sisteme de dialog maşină-om) este de a realiza aceste sisteme "acum şi aici". Pentru a fi capabil de acest lucru, este necesar ca el să aibă la dispoziţie modele şi metodologii rezultate din cercetarea în domeniul inteligenţei artificiale. Toate sistemele înşiruite mai sus necesită modele cu un grad de complexitate destul de ridicat, care face imposibilă realizarea lor prin utilizarea unei singure paradigme computaţionale. Pentru a cuprinde diversele operaţii care trebuie realizate, este necesară utilizarea simultană a mai multor modele, atât din spectrul simbolic, cât şi din cel conexionist. Complexitatea, dar mai ales diversitatea proceselor cognitive implicate chiar în activităţi aparent simple pentru om, a condus la ideea că un singur model unificat nu este suficient pentru atingerea unor scopuri imediate, lucru accentuat şi în [Minsky91]: "*To solve really hard problems, we'll have to use several different models and representations*". Există fireşte şi scopuri cu termen lung. Cel mai important este fireşte emularea prin maşină a principalelor capacităţilor cognitive umane. Probabil că nu va conta în acel moment dacă acest lucru a fost obţinut prin sisteme hibride. La fel de importantă este şi realizarea unor modele plauzibile din punct de vedere biologic, dar mai importantă este plauzibilitatea psihologică, legată mai mult de comportamentul sistemului şi mai puţin de detaliile de implementare. Contraargumentul invocat de direcţia de cercetare unificată este că succesul sistemelor de calcul existente este bazat pe arhitectura Von Neumann, care are un caracter puternic unificat. Dar pentru a realiza **acum** sisteme viabile, este necesar să se folosească din arsenalul existent, metodele cele mai avantajoase din punct de vedere computaţional. Folosirea unei strategii unificate duce de regulă la necesitatea unor resurse computaţionale (memorie, timp de calcul) exagerate. S-a observat că folosind o strategie hibridă, pentru aceiaşi problemă, necesarul de resurse este apreciabil mai mic [Miikulainen94].

Cercetarea în direcţia integrării neuro-simbolice (perioada 1986-1996) a încercat să ofere **modele** pentru realizarea unor sisteme economic viabile. Pentru că nici unul dintre acestea nu are la baza o teorie demonstrată a corectitudinii funcţionării, doar un indice de performanţă procentual, este puţin exagerată terminologia de model (deşi ea este folosită în majoritatea lucrărilor de specialitate). Eu consider însă că termenul de *metodologie* este mai potrivit. În momentul actual este imposibil de oferit un model corect funcţional, care să aibă în componenţa lui reţele neuronale, pentru că nu s-a demonstrat că aceste reţele au o funcţionare corectă (indiferent de natura informaţiei de intrare). Bineînţeles, în această discuţie, contează şi ce înţelegem prin *corectitudine*. Dacă o reţea recunoaşte toate tiparele cu care a fost antrenată şi în plus recunoaşte un set de probă alcătuit dintr-o mulţime mai mare de tipare decât baza de antrenare (fără a conţine însă tipare foarte diferite de cele învăţate), mulţi cercetători afirmă că reţeaua funcţionează corect, mai ales dacă poate rejecta tipare foarte diferite de cele antrenate. Este doar un punct de vedere, pentru că un analist mai riguros ar clasa ca funcţionare incorectă, orice rejectare sau interpretare greşită a oricărui tipar, indiferent de gradul său de similitudine cu cele învăţate. Din acest motiv, atunci când în teză este folosit

cuvântul model, cititorul riguros îl poate interpreta ca **metodologie**.

Metodologia propusă în teză este rezultatul unor idei personale, dar în acelaşi timp ea reflectă şi observaţiile făcute asupra unor sisteme propuse de alţii. Există o serie de metodologii propuse pentru sisteme hibride şi am încercat dealungul tezei să compar ideile şi soluţiile mele cu acestea. Pentru a-mi plasa metodologia în contextul altor propuneri, este necesară o sistematizare a acestora. Modelele hibride pot fi grupate în două mari familii: modele hibride translaţionale şi modele hibride funcţionale [Hilario86].

- **modelele translaţionale** sunt o clasă intermediară între modelele strategiei unificate şi modelele funcţionale. Ca şi modelele unificate (numite şi modele uniforme) cele translaţionale se bazează pe reţele neuronale pe post de procesoare de informaţie, dar datele de start şi rezultatele sunt structuri simbolice. De cele mai multe ori acestea sunt reguli propoziţionale [Fu&Fu90] [Towell91] (dacă este privit ca un generator de reguli - vezi discuţia despre acest model din capitolul 1) [Kunicky&al92]. În alte cazuri [Crucianu&Memmi92] [Sun92] [Giles93] [Miikulainen96] [Wermter&Weber97], limbajul natural (sau mai bine zis o componentă simplificată a lui) este cel pe baza căruia se construiesc structurile simbolice. Acestea nu sunt procesate ca atare, prin metode simbolice, ci sunt traduse fie în reţele neuronale, care îşi modifică parametrii sau chiar structura prin adăugarea unor noi componente prin antrenare, fie în vectori de activare neuronali, care sunt intrări pentru o reţea [Medsker94]. Rezultatul se obţine prin traducerea reţelei finale sau a ieşirii ei într-o structură simbolică. Merită remarcat faptul că sistemele construite pe baza acestui model, de exemplu CONSYDERR [Sun94], NP [Giles95] şi SCREEN [Wermter&Weber97] au performanţe foarte reduse, mai ales datorită limitărilor dimensionale ale reţelelor folosite. Unul din sistemele care au rezultate bune (în faza experimentală) este DISCERN [Miikulainen96], dar în acest caz, cu toate că toate procesările se fac prin reţele neuronale, există mai multe elemente de procesare, cu topologie şi funcţionalitate diferită. Primele mele propuneri de model [Szirbik&al95] [Szirbik95b] [Szirbik&Babii96] se pot înscrie în această clasă a modelelor translaţionale.

- **modelele hibride funcţionale** încorporează componente *complete* simbolice şi conexioniste. Acestea sunt caracterizate pe de o parte de reprezentările (structurile de date) folosite şi pe de altă parte de schemele de procesare. Pe lângă diferite tipuri de reţele neuronale putem avea: interpretoare de reguli, analizoare sintactice, automate de raţionament şi demonstratoare de teoreme. Aceste sisteme pot fi diferenţiate privind dealungul mai multor dimensiuni imaginare, cum sunt:

- clasa de aplicaţii pentru care se potriveşte modelul,
- gradul de generalitate,
- stabilirea componentei dominante (conexionistă sau simbolică), numită şi *mod de integrare* [Minsky91],
- gradul de integrare între componentele simbolice şi cele conexioniste.

Cele mai importante aspecte pentru mine au fost ultimele două. O taxonomie bazată pe gradul de integrare oferă două variante: prin cuplare puternică (tight coupling) şi cuplare slabă (loose coupling). În modelele cu cuplare slabă, interacţiunea între cele două tipuri de componente este strict localizată în spaţiu şi timp. Informaţia se transferă prin intermediul unui intermediar (agent, supervizor sau un "blackboard") iar interacţiunea este iniţiată întotdeauna explicit, fie pe baza unui protocol, fie datorită unui agent de control exterior. În sistemele cu cuplare puternică, datele nu mai sunt transferate de la componentele simbolice la cele conexioniste şi invers, ci sunt partajate prin structuri comune de date. Astfel, o modificare făcută de una din componente are repercursiuni imediate asupra celeilalte componente, fără să existe necesitatea iniţierii unei interacţiuni explicite. Unul din sistemele propuse este SEMMARK [Hendler89], unde unităţi ale unei reţele neuronale aveau legături directe cu acţiunile semantice asociate unei analize sintactice hibride. O alta propunere este "filtrul" neurosimbolic, care asigura legătura reprezentărilor conexioniste distribuite cu structurile simbolice din SYNTHESIS [Giacometti92]. Astfel de sisteme puternic cuplate (integrate) nu au avut rezultate deosebite, explicaţia fiind incompatibilitatea naturală a reprezentărilor simbolice şi conexioniste. Se poate însă observa o revenire a lor [Fahlman97] [Farhat97] [Shavlik97], deşi nu se poate vorbi încă de rezultate utilizabile în sistemele viabile economic (în plus, toate propunerile fiind legate mai ales de utilizarea reţelelor nesupervizate şi mai puţin revenirea la sistemele puternic cuplate).

Anii 92-96 au fost caracterizaţi mai ales prin dezvoltarea modelelor cu cuplare slabă. Rezultate încurajatoare au fost obţinute mai ales în domeniul medical ca FIBRIL [Jabri&al92], REASONER [Piramuthu&Shaw94] sau NST-EXPERT [Alonso&al95], dar şi în alte domenii, de menţionat fiind controlul roboţilor (pe larg, subiectul este tratat în [Leonard&Durrant94]). Unul din sistemele folosite efectiv, chiar pe scară largă este WASTEWATER [Krovvidy&Wee92]. Modelele mai recente, cum sunt DUAL [Kokinov96], NESSY3L [Orsier&Labbi96] sau GENUES (Generic Neuro-Expert System) [Khosla&Dillon96], propun noi arhitecturi, făcând o analiză prealabilă, puternic experimentală şi mai puţin teoretică a ideilor propuse. Cu toate că au existat şi încercări notabile de formalizare a integrării neurosimbolice [Smolensky86,88,90,92], complexitatea covârşitoare a problemei a făcut ca domeniul să fie unul eminamente experimental. Există critici serioase relativ la acest tip de abordare [Touretzky90], [Smolensky92], la un moment dat folosindu-se chiar epitetul de "alchimie" [Crevier93], dar există şi argumentări pro care pleacă de la imposibilitatea unei alternative [Honavar94], [Alexandre96].

Modul de integrare sau *schema ierarhică* se referă la modul în care componentele simbolice şi conexioniste sunt configurate relativ unele la altele şi relativ la întregul sistem. Patru scheme pot fi identificate:
- procesare secvenţială,
- sub-procesare,
- meta-procesare,
- co-procesare.

În cazul procesării secvenţiale (numită şi chainprocessing), una din cele două

componente are rolul principal (fie cea conexionistă fie cea simbolică) şi cealaltă se ocupă de operaţiile de pre/post procesare. Schema este folosită mai ales de sistemle expert medicale care au şi intrare analogică [Hayes&al92], [Alonso95] (ambele sisteme referite având componenta principală simbolică - un caz invers fiind WASTEWATER [Krovvidy&Wee92]). În cazul sub-procesării, una din componente este interioară celeilalte, care acţionează ca şi componentă principală. De obicei componenta simbolică este cea principală, ca în INNATE/QUALMS [Becraft&al91], unde un generator neuronal de situaţii candidate este apelat în mod repetat de procesorul simbolic. În domeniul roboticii, schema cu meta-procesare este cea mai folosită. Un nivel simbolic de meta-procesare care monitorizează mai multe componente neuronale este realizat în RSAA (Robotic Skill Acquisition Architecture) [Handelman&al89] şi COOHYS [Gutknecht&al91].

Cea mai interesantă şi mai plauzibilă psihologic şi biologic este co-procesarea. Componenta simbolică şi cea neuronală sunt parteneri egali, ambele pot avea interacţiuni directe cu exteriorul, ambele pot transmite şi recepţiona informaţie de la cealaltă componentă. Există două posibilităţi: fie cele două funcţionează fără supervizare (FIBRIL [Jabri&al91], fie exită un al treilea nivel de metaprocesare care monitorizează cooperarea celor două componente ca în NESSY3L [Orsier&Labbi96]. O variantă pe care am încercat-o eu a fost schema cu doar două nivele [Szirbik96] [Szirbik97], unde nivelul simbolic are şi rolul implicit de control.

La nivelul componentei conexioniste, este important de precizat ce fel de metode se folosesc pentru procesare. Putem avea reţele nesupervizate sau supervizate. Din punct de vedere al dinamicii reţelelor, putem avea reţele incrementale, care se antrenează în timpul utilizării sistemului, sau reţele statice, care sunt antrenate doar în timpul construirii lui. Se pot folosi reţele feedforward sau recurente, cu valori continue sau discrete, sau combinaţii ale acestora. În cazul sistemelor expert, se folosesc reţele feedforward supervizate (sau nesupervizate în cazul INNATE/QUALMS [Becraft&al91]) cu valori de intrare reale şi ieşiri binare. Pentru aplicaţiile în robotică, se folosesc mai mult reţele recurente dinamice.

Ca ultim aspect taxonomic, trebuie menţionate aria de aplicabilitate şi generalitate a modelului. Deşi primele sisteme expert hibride dezvoltate de mine au fost în domeniul medical, mai exact în domeniul bolilor infecţioase [Dragomirescu&al93a] [Dragomirescu&al93b] [Dragomirescu&al94], căutarea unui model nou, m-a făcut să aleg un domeniu cu un grad mai mare de generalitate, şi anume raţionamentul automat, fără a aplica imediat metodologia la o problemă specifică. Am încercat diverse modele de raţionament automat cu reţele feedforward [Szirbik&al93] [Szirbik93], aplicate apoi la sistemele expert medicale.

Modelul propus de mine în finalul acestei teze, poate fi clasificat ca fiind un **model hibrid funcţional, cu cuplare slabă între componentele simbolice şi cele conexioniste, cu co-procesare şi meta-procesare deopotrivă, care foloseşte reţele recurente supervizate** (iar într-o variantă îmbunatăţită foloseşte şi reţele multistrat).

Această combinaţie este oarecum asemănătoare cu cea propusă de Fahlman [Fahlman97], care are însă o cuplare mai puternică şi reţele recurente nesupervizate (rezultatele fiind însă neconcludente până în prezent).

I3. Structura tezei

Majoritatea rezultatelor prezentate în teză au fost publicate în volumele unor conferinţe sau în periodice de specialitate. Există şi o serie de lucrări publicate care sunt doar referite dar nu şi prezentate pe larg în teză (perioada 1991-94). Ele se referă la cercetarea din domeniul sistemelor expert pentru studierea procesului infecţios. Este prezentată însă metodologia folosită în aceste sisteme expert [Szirbik&al93] [Szirbik93] şi anume o variantă specifică a modelului Towell [Towell91]. Deasemenea, rezultatele din 1994 şi 1995 sunt prezentate la un grad mult mai mare de detaliu decât în lucrările publicate pe această temă [Szirbik&al95] [Szirbik95a] [Szirbik95b]. Metodologia finală propusă a fost prezentată în [Szirbik96] şi [Szirbik97], dar in teză se face în plus o prezentare exhaustivă a rezultatelor experimentale şi a interpretării lor.

Ca şi confirmare a viabilităţii metodologiei propuse şi a aplicabilităţii ei, invitaţia de a discuta o serie de rezultate şi idei la US Air Force Wright Laboratory (Dayton, Ohio) şi la Neuro-Lab al Carnegie-Mellon University (Pittsburgh, Pennsylvania) au condus la două seminarii cu colective de cercetători din aceste instituţii (aprilie 1997). Rezultatele prezentate în Statele Unite şi interpretările specialiştilor de acolo sunt prezentate în teză.

• Capitolul 1: Prezintă un model de raţionament implementat cu ajutorul reţelelor neuronale feedforward, propus de Towell in 1991. Se analizează performanţele, scăderile şi limitările acestui tip de procesare. Se introduc o serie de termeni care vor fi pe larg utilizaţi în teză.

• Capitolul 2: Prezintă cele două instrumente de lucru folosite, sistemul neurodinamic discret (instrument pur conexionist) şi grafurile conceptuale (instrument de reprezentare pur simbolică a cunoştinţelor). Se insistă în special pe unele particularităţi care au avut o mare importanţă relativ la experimentele făcute cu modelul propus.

• Capitolul 3: Prezintă modul în care am implementat transformarea structurilor simbolice în vectori de activare conexionişti. Sunt discutate şi o serie de consideraţii dimensionale.

• Capitolul 4: Prezintă prima arhitectură implementată şi testată, modelul monomodular hibrid translaţional. Este prezentată şi o posibilă aplicaţie pentru planificare robustă. Sunt pe larg prezentate rezultatele experimentale care au arătat puterea de generalizare a unui astfel de sistem. Problema bifurcaţiilor la antrenare este explicată în detaliu, precum şi o soluţie de evitare a ei. **Un fenomen descoperit**

experimental şi demonstrat formal, numit de mine principiul divergenţei este deasemenea prezentat.

• Capitolul 5: Prezintă modul în care au fost dezvoltate două sisteme de test, bazate pe un model distribuit, numit modelul (translaţional hibrid) multimodular omogen, precum şi limitările sale.

• Capitolul 6: Modelul final, denumit modelul multimodular hibrid eterogen (sau modelul pe două nivele) este prezentat ca şi metodologie de dezvoltare pentru sisteme cognitive. Sunt prezentate experimentele care demonstrează puterea de generalizare a componentei conexioniste şi capacitatea acesteia pentru anticiparea unor situaţii neantrenate sau implementate în sistem.

• Un ultim capitol, este dedicat concluziilor, contribuţiilor originale ale autorului tezei şi posibilelor linii de continuare ale acestei cercetări.

1. Suportul teoretic al unui model neurosimbolic unificat

Modelul neuronilor artificiali, propus de McCulloch şi Pitts în 1943, îmbunătăţit de Hebb şi rafinat de Rosenblatt în anii '50 (o prezentare detaliată a istoriei domeniului se poate găsi în [Hecht90]), este pe de o parte foarte depărtat de modelul neuronilor naturali (o excelentă prezentare "inginerească" a lor în [DeFelipe97]) şi este foarte diferit ca şi natură matematică de operaţiile folosite în logică şi sistemele formale. Făcând această constatare, este normal să ne întrebăm dacă este posibilă realizarea unor operaţii simbolice cu ajutorul reţelelor neuronale formate din neuroni artificiali. Că se pot realiza operaţii simbolice cu neuroni naturali, este mai mult decât evident, din moment ce omul poate să facă operaţii simbolice la nivel mental. Întrebarea este dacă reţele neuronale artificiale, formate dintr-un număr mult mai mic de neuroni decât cei disponibili în creierul uman pot procesa simboluri şi mai ales care sunt limitările impuse de modelul artificial. Există şi modele matematice pentru neuroni artificiali mai apropiate de schemele neurofiziologice ale neuronilor naturali (există foarte multe tipuri de celule nervoase, mai mult sau mai puţin specializate), dar acestea sunt încă în fază experimentală şi au dezavantaje: complexitate teoretică mare şi un necesar de resurse de implementare ridicat. Din acest motiv, trebuie să utilizăm vechiul model, foarte simplu şi uşor de implementat. Din punct de vedere teoretic, este destul de bine ştiut ce se poate face şi ce nu se poate face cu astfel de reţele [Hertz&al91], [Haykin94].

In continuare, se prezentă succint un model foarte simplu de procesare simbolică cu reţele neuronale (unificat), şi se introduc câteva noţiuni de bază, care sunt folosite pe larg în teză.

1.1. Modelul neuronului artificial, reţeaua feedforward

Neuronul artificial este un procesor foarte simplu de informaţie, care primeşte la intrare un set de valori, grupate de regulă într-un vector, şi generează o singură ieşire. Valorile pot fi întregi, reale sau grupuri de valori de tip matrice de diferite

tipuri. De regulă, întrările sunt numere reale. Întotdeauna intrările sunt însumate, obţinându-se *nivelul intern de activare* (notat în teză cu *h*). Aspectul esenţial care defineşte un neuron şi îl deosebeşte de ceilalţi cu care se află în reţea este *vectorul de ponderi w* (de aceiaşi lungime ca şi vectorul de intrare pentru neuronul respectiv). Din acest motiv, în loc de o simplă sumă a intrărilor se foloseşte ca nivel intern de activare, produsul scalar dintre cei doi vectori. Asupra acestei mărimi se aplică o funcţie de transfer (notată în teză cu *g*), iar valoarea acestei funcţii reprezintă ieşirea neuronului matematic, numită şi *nivel de activare* al neuronului. Dacă funcţia **g** este funcţia identitate sau o simpla multiplicare, atunci neuronul se numeşte *liniar*. Dacă notăm cu *x* vectorul de intrare şi cu *o* ieşirea, ecuaţia care descrie un asemenea neuron liniar este următoarea (γ fiind factorul de multiplicare):

$$o = \gamma \cdot \sum_{i=1}^{n} w_i \cdot x_i \qquad (1\text{-}1)$$

Acest tip de neuron nu are prea multe aplicaţii [Smolensky86]. Pentru neuroni *neliniari* se folosesc alte funcţii de transfer, de tip treaptă, continue sau discontinue. Funcţia pe care am folosit-o cu predilecţie în teză este tangenta hiperbolică, care are domeniul de definiţie în **R**, şi ia valori în mulţimea reală (-1,1). Ar fi mai simplu să folosim funcţii discontinue, de tip treaptă:

$$o = \begin{cases} 1 & daca \quad h \geq T \\ 0 & daca \quad h < T \end{cases} \quad \left(h = \sum w_i x_i\right) \qquad (1\text{-}2)$$

dar există considerente matematice care impun folosirea funcţilor continue (derivabilitatea). Funcţia treaptă (care are valori în {0,1}) este însă mai apropiată de ideea că un neuron are o funcţionare asemănătoare cu a unui bistabil, neuronul având doar două stări posibile: activ şi non-activ. Dar chiar folosind ca funcţie de transfer tangenta hiperbolică, se poate suplimentar transforma ieşirea dintr-o valoare reală într-o valoare binară:

$$o' = \begin{cases} 1 & daca \quad o \in [0.5,\ 1) \\ 0 & daca \quad o \in (-1,\ -0.5] \end{cases} \qquad (1\text{-}3)$$

şi se poate observa că apare posibilitatea unei a treia stări (incerte, atunci când $o \in (-0.5,\ 0.5)$). Există însă tehnici pentru a evita apariţia acestei stări nedorite.

O astfel de stare binară se poate asocia valorilor de TRUE şi FALSE din logică. Unui neuron i se poate asocia un simbol (cuvânt). Dacă avem mai mulţi astfel de neuroni cu simboluri asociate într-un sistem neuronal (reţea), ieşirea lor poate fi interpretată ca valoarea de adevăr a simbolurilor asociate. Sunt însă şi situaţii când se doreşte valoarea exactă a funcţiei *tanh*, când folosim de exemplu un sistem logic multivalent, sau factori de certitudine [Shortliffe76]. De exemplu, în sistemele expert dezvoltate de mine şi de colegii mei în colaborare cu medici [Dragomirescu&al94], am folosit o variantă de logică cuadrivalorică, ale cărei valori de adevăr se obţineau printr-

o transformare simplă din valorile de ieşire ale neuronilor (care aveau funcţia de transfer *tanh*). Modelul care va fi prezentat în continuare (numit şi model Towell [Towell91]) lucrează doar cu două valori de adevăr.

Foarte rar se întâmplă să existe aplicaţii pentru un singur neuron. Când sunt folosiţi mai mulţi, sunt legaţi între ei şi formează reţele. Modul în care sunt legaţi defineşte *topologia* reţelei. Există multe topologii posibile [Hecht90], dar două sunt cele mai folosite: feedforward şi complet recurentă (aceasta din urmă va fi comentată în detaliu în capitolul următor). Modelul unificat Towell foloseşte o reţea cu topologie feedforward. Cea mai simplă reţea feedforward este reţeaua cu un singur strat. Aceasta este formată dintr-un număr de *m* neuroni care au în comun faptul că toţi au acelaşi număr de intrări *n*. Funcţionarea acestei reţele este foarte simplă: ca intrare se furnizează un vector (de obicei real) de lungime *n*, vector care este intrare pentru fiecare neuron. Ieşirea va fi un vector de lungime *m*, binar în cazul în care funcţiile de transfer sunt toate de tipul descris de formula (1-2), sau real, când cel puţin una din funcţii este continuă. Un vector real se poate transforma în unul binar, aplicând formula (1-3), cu condiţia ca valorile sale să fie în $(-1,0.5]\cup[0.5,1)$.

Reţeaua descrisă mai sus poate fi folosită ca şi element constructiv pentru reţelele feedforward multistrat. Dacă avem o reţea cu *n1* intrări şi *n2* ieşiri, putem cupla la ieşirea acesteia o nouă reţea de acelaşi tip, singura restricţie fiind ca aceasta să aibă *n2* intrări. Se poate lega o altă reţea în continuare şi aşa mai departe, numărul de reţele legate în lanţ definind numărul de straturi al reţelei finale. Teorema Vapnik-Chervonenkis [Hertz&al91] arată că pentru marea majoritate a aplicaţiilor este suficent să folosim două straturi cu unităţi continue, dar pentru reţelele Towell, numărul de straturi este dat de forma cunoştinţelor simbolice folosite pentru construirea sistemului. Pentru o reţea multistrat, care are toţi neuronii dotaţi cu aceiaşi funcţie de transfer *g*, funcţia de transfer globală este:

$$o = \Gamma(W_s \times \Gamma(W_{s-1} \times \ldots \Gamma(W_1 \times x) \ldots)) \qquad (1\text{-}4)$$

unde *s* este numărul de straturi, iar Γ este o funcţie vectorială care aplică funcţia de transfer a neuronilor folosiţi în reţea (*g*) peste un vector, element cu element. $\mathbf{W}_i$ (i=1,s) sunt matricile care definesc reţeaua, ale căror linii sunt formate din vectorii de ponderi transpuşi ai neuronilor de pe fiecare strat. Se poate observa că reţeaua se poate implementa foarte simplu, atât software cât şi hardware [Beiu96].

O astfel de reţea poate primi la intrare un vector de o lungime fixată şi poate genera la ieşire un alt vector de altă lungime, deasemenea fixă. Timpul de calcul necesar este foarte mic, răspunsul unei astfel de structuri computaţionale fiind unul dintre cele mai scurte, comparativ cu alte implementări care fac aceiaşi transformare a vectorului de intrare în vector de ieşire. Acest tip de transformare poate fi descris prin două metode cât se poate de diferite. Prima, riguroasă, descrie maparea între cele două spaţii vectoriale (de intrare şi ieşire) printr-o formulă matematică. A doua, experimentală, descrie maparea printr-o mulţime de perechi de vectori (cât mai mare),

fără a oferi nici o exprimare formală. Foarte multe din problemele din tehnică sunt exprimate prin a doua variantă, fiind extrem de dificil a se construi o funcţie care să exprime legătura dintre intrarea şi ieşirea unui sistem.

Fiind dată o mulţime de perechi de vectori $\{(\mathbf{x}_p,\mathbf{o}_p),p=1,P\}$, există metode prin care se poate realiza maparea între spaţiul de intrare unde sunt fixaţi vectorii $\mathbf{x}_p$ şi spaţiul de ieşire, unde sunt fixaţi vectorii $\mathbf{o}_p$, folosind o reţea neuronală feedforward multistrat. Acestea realizează o aproximare locală a funcţiei de mapare (care nu poate fi identificată formal) prin aşa zişii *algoritmi de antrenare* care calculează prin metode bazate pe minimizarea funcţiilor pe bază de gradient, valorile matricilor de ponderi ale reţelei. Pentru o reţea de tipul descris, cel mai folosit algoritm este clasicul *backpropagation* (o foarte bună descrire, în [Haykin94]). O versiune adaptată folosită pentru topologiile complet recurente este discutată pe larg în capitolul 2.

1.2. Modelul Towell

Cea mai răspândită formă de reprezentare a cunoştinţelor la nivel simbolic sunt **regulile**. Acestea pot fi descrise la nivel propoziţional sau la diferite nivele predicative [Luger89]. Forma generală a unei reguli propoziţionale conjunctive este:

C :- P_1 and ... and P_n.

unde C şi P_i sunt propoziţii atomice. Semantica unei astfel de piese de cunoaştere se poate exprima prin propoziţia: "*C este o propoziţie adevărată, daca şi numai dacă toate propoziţiile P_i sunt adevărate*". Există şi reguli disjunctive, unde este suficient ca o singură propoziţie P_i să fie adevărată. O a treia categorie este cea a regulilor combinate, dar este simplu de arătat [Towell91] că un set de reguli combinate poate fi redus la un set de reguli pur disjunctive şi un set de reguli pur conjunctive. O astfel de colecţie de reguli poate fi privită ca o bază de cunoştinţe foarte simplă. Există două metode de a utiliza static această bază de cunoştinţe, folosind două mecanisme de inferenţă diferite, *forward-chaining* sau *backward-chaining*. Modelul Towell precizează calea de implementare a unei astfel de baze de cunoştinţe care este interogată prin mecanismul de forward-chaining, folosind o reţea neuronală feedforward cu mai multe straturi.

Dacă avem o regulă ca cea de mai sus (conjunctivă), asociem valorile de adevăr TRUE şi FALSE cu valorile numerice 1 şi -1 şi putem folosi ca automat de evaluare a consecinţei C a regulii un neuron cu *n* intrări asociate premiselor P_i. Acest neuron va avea de acum încolo un simbol asociat (C). Dacă folosim următoarea funcţie de transfer:

$$o=\begin{cases} 1 & daca\ h\geq 1 \\ -1 & daca\ h<1 \end{cases} \; ; \; h=\sum_{i=1}^{n} w_i \cdot x_i \; ; \; w_i=\frac{1}{n} \; (i=1,n) \qquad (1\text{-}5)$$

unde:

$$x_i = \begin{cases} 1 & daca \quad P_i=TRUE \\ -1 & daca \quad P_i=FALSE \end{cases} \qquad (1\text{-}6)$$

se observă că făcând transformarea inversă faţă de (1-6) pentru ieşirea *o* se poate obţine o valoare de adevăr pentru simbolul C, dacă sunt cunoscute valorile de adevăr pentru simbolurile P_i, iar acest calcul este corect din punct de vedere logic [Shavlik&Towell91]. Pentru regulile disjunctive (1-5) se schimbă în:

$$o=\begin{cases} 1 \ daca \ h\geq 1 \\ -1 \ daca \ h<1 \end{cases} ; \ h=\sum_{i=1}^{n} w_i \cdot x_i \ ; \ w_i=1 \ (i=1,n) \qquad (1\text{-}7)$$

unde diferă doar metoda de setare a ponderilor. Neuronii astfel obţinuţi pot fi transformaţi în neuroni cu funcţie de transfer continuă prin înlocuirea funcţiilor de mai sus prin tangenta hiperbolică. Această modificare presupune mutarea treptei în origine:

$$\sum_{i=1}^{n} w_i x_i \geq 1 \Rightarrow \sum_{i=1}^{n} w_i x_i - 1 \geq 0 \Rightarrow \sum_{i=0}^{n} w_i x_i \geq 0 \ (w_0=-1; \ x_0=1)$$

fapt care introduce o intrare suplimentară, setată mereu pe TRUE, şi o pondere suplimentară, care are din start valoarea valoarea -1, indiferent dacă regula este disjunctivă sau conjunctivă. Pentru a avea o treaptă cât mai abruptă, asemănătoare cu funcţia discretă, funcţia de transfer $o=tahn(h)$ se înlocuieşte cu $o=tanh(\beta h)$. Valoarea β se alege în funcţie de numărul de întrări iniţiale în neuronul respectiv, o valoare propusă prin experiment [Towell91] fiind $\beta=n/1.65$. Pentru a avea o funcţionare cât mai apropiată de funcţia treaptă, ponderea de indice zero (cea suplimentară) se setează pe valoarea -0.85, ca să fim siguri că în cazul în care *h* este mai mare decât 1, la ieşire vom avea o valoare apropiată de 1.0, transformată prin (1-3) în 1 întreg, iar în continuare, de inversa ecuaţiei (1-6) în TRUE.

Fiecare regulă din baza de cunoştinţe iniţială va constitui în final un neuron, care va avea un număr de intrări egal cu numărul de premise din partea dreaptă a regulii plus o intrare suplimentară setată mereu pe TRUE (pe aceasta o vom ignora în continuarea prezentării, ea considerându-se implicită). Pentru ca avem două structuri neuronale care instanţiază reguli pur disjunctive sau pur conjunctive, este necesară o metodă de transformare a regulilor iniţiale *combinate* în reguli care sunt doar de aceste două tipuri. Ilustrată printr-un exemplu, metoda este următoarea:
-dacă avem regula

A :- (B and C and D) or (E and F and G).

regula va fi rescrisă în trei reguli care au aceiaşi semantică:

A :- A' or A".
A' :- B and C and D.
A" :- E and F and G.

rezultând trei neuroni, care vor avea ponderile egale cu 1/3 pentru regulile cu partea stângă A' şi A", iar pentru regula cu partea stângă A cele două ponderi vor fi egale cu 1. Având o bază de cunoştinţe astfel transcrisă şi care nu are reguli izolate (nu este formată din simboluri care nu aparţin şi altor reguli), transformând toate regulile în neuroni, aceştia pot fi legaţi între ei. Ieşirea unui neuron care are asociat un simbol X (partea stângă a regulii este X) va fi o intrare a altor neuroni care sunt asociaţi unor reguli care îl conţin pe X în partea dreaptă. Exemplul din figura 1-1 (a., b., c. şi d.) arată cum se leagă neuronii între ei, pornind de la un mic set de reguli. În [Towell91] este dată şi soluţia pentru legarea în reţea a unor simboluri precedate de operatorul logic unar "NOT", prin folosirea unor ponderi cu valoare negativă şi prin schimbarea semnului valorilor de ieşire din neuroni, dar nu voi intra aici în astfel de amănunte.

Unităţile (neuronii) din figură care sunt desenaţi ca triunghiuri, au doar rol de a transforma valorile logice de adevăr ale simbolurilor asociate în valori numerice şi invers. Se poate observa că rezultă o reţea feedforward, care are ca intrări valorile asociate acelor simboluri care apar doar în partea dreaptă a unei reguli din baza iniţială de cunoştinţe (posibile premise de raţionament). Ieşirile acestei reţele sunt asociate unor simboluri care apar numai în parte stângă în setul de reguli (posibile concluzii finale de raţionament). Dacă ştim despre un subset de posibile premise că sunt adevărate şi pornim reţeaua cu aceste valori, vom obţine la ieşire un subset de simboluri care sunt concluzii finale şi care au valoarea logică de adevăr TRUE. În [Towell91] se demonstrează izomorfismul funcţional dintre un sistem de raţionament bazat pe *modus-ponens* (forward-chaining) şi această reţea. Modelul Towell mai este cunoscut în literatură şi sub denumirea de KBANN (Knowledge Based Artificial Neural Networks). Cele două modele de raţionament (unul pur simbolic şi unul conexionist), care pot fi implementate în mod foarte diferit, au fiecare o serie de avantaje şi de dezavantaje.

Modelul pur simbolic poate fi uşor implementat printr-un mecanism de pattern-matching (de exemplu în LISP - chiar pe o maşină dedicată). Regulile pot fi exprimate uşor ca liste, manipularea lor şi interfaţa sistemului cu utilizatorul fiind facile. Marele avantaj al unui astfel de sistem este însă *plasticitatea* [Shavlik&Towell91]. Este foarte simplu să adăugăm sau să ştergem reguli într-un astfel de sistem. Dezavantajul major al modelului simbolic este o caracteristică numită "crispness" (termen propus în [Pearl88] - şi pe care nu am reuşit să-l traduc). Aceasta se manifestă prin următoarele două probleme:

- un astfel de sistem nu poate fi construit decât plecând de la cunoştinţe care pot fi exprimate prin reguli care sunt în orice context adevărate.
- Dacă lipseşte un singur element de intrare pentru un anumit raţionament, acesta este invalidat (ceea ce poate fi considerat şi o lipsă de robusteţe).

Aceste probleme apar mai ales atunci când:

- cunoştinţele care sunt folosite la construirea sistemului au un anumit grad de incertitudine, sau nu pot fi reprezentate prin reguli.

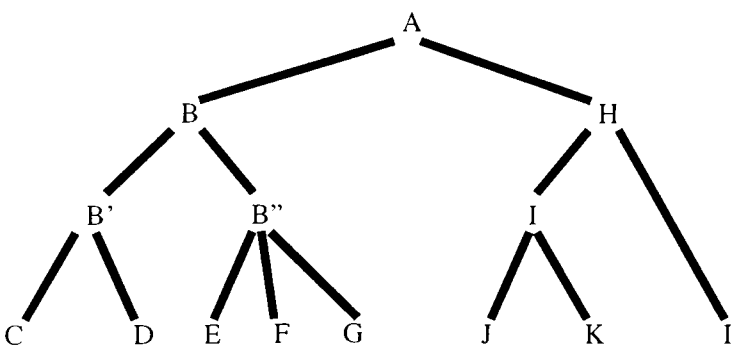
figura 1-1.

A :- B and H.
B :- (C and D) or (E and F and G).
H :- G and I.
G :- J and K.

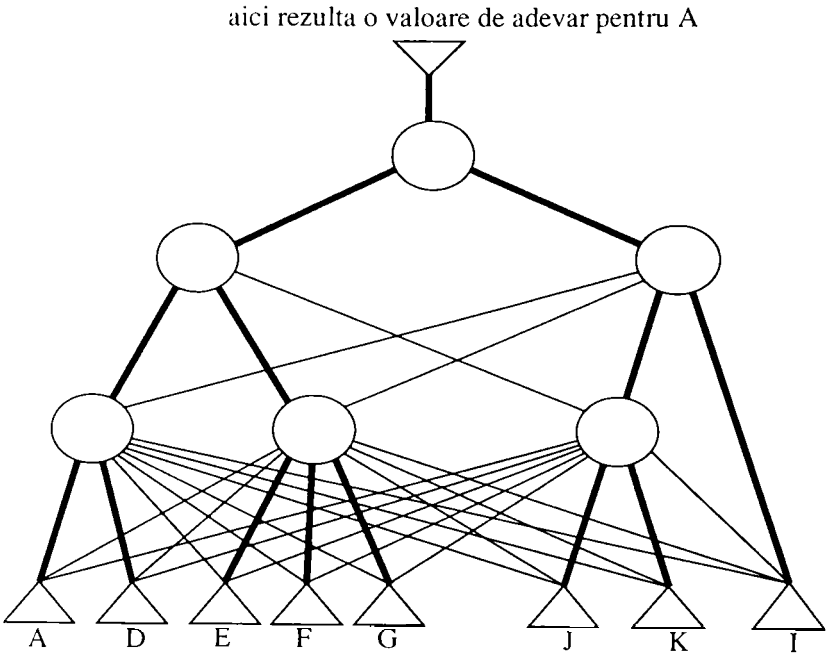
a. baza de cunostinte initiala

A :- B and H.
B :- B' and B".
B' :- C and D.
B" :- E and F and G.
H :- G and I.
G :- J and K.

b. baza de cunostinte rescrisa
(are doar reguli pur conjunctive si pur disjunctive)



c. graful de decizie construit pe baza regulilor



d. reteaua finala

Legenda:

ponderi setate initial prin formulele 1-5 si 1-7

ponderi practic nule (valori mici aleatoare)

figura 1-1 (continuare)

• datele de intrare au un grad de incertitudine sau sunt incomplete.

Pe lângă abordarea KBANN, au mai existat şi alte metodologii care au încercat surmontarea acestor probleme, ca MYCIN [Shortliffe76], ODYSSEUS [Wilkins88], RTLS [Ginsberg90] şi KR-FOCL [Pazzani&Brunk91]. Nici una până la acea dată (1991) nu a încercat însă o abordare neuronală.

Pentru KBANN, principalul dezavantaj este lipsa de plasticitate. Inserarea unor noi reguli în reţeua neuronală este dificilă, modificându-se de fiecare dată matricile de ponderi ale reţelei rezultate. Dar există o serie de avantaje faţă de modelul simbolic. Dacă lucrăm cu funcţii de transfer continue, valorile care pot apărea la intrarea şi ieşirea sistemului nu vor fi valori "crisp" de adevăr (TRUE şi FALSE) ci valori reale în intervalul [-1,1] care pot reprezenta un anumit grad de incertitudine în premise şi rezultate. Cel mai mare avantaj este însă posibilitatea inserării în sistem a unor cunoştinţe empirice, exprimate prin perechi de seturi intrare/ieşire, care pot avea şi o exprimare incertă (cunoştinţe imposibil de exprimat prin reguli). Putem antrena reţeaua (obţinută prin algoritmul prezentat în figura 1-1) cu aceste perechi, folosind backpropagation, modificând ponderile astfel încât sistemul să poată reproduce şi raţionamente empirice. Schema de antrenare [Towell91] presupune şi introducerea de noi noduri şi ponderi. Trebuie păstrat însă un echilibru între cunoştinţele iniţiale, exprimate prin regulile de start şi cunoştinţele empirice, pentru că o antrenare cu prea multe cunoştinţe empirice poate distruge structura iniţială a reţelei.

Există două posibilităţi de utiliza un astfel de sistem conexionist (care este unificat, pentru că odată implementat, nu mai are nevoie de regulile iniţiale sau de alt supor simbolic). Prima posibilitate este de a folosi sistemul ca şi sistem expert, construit pe baza unor cunoştinţe simbolice (chiar incerte, vezi [Dragomirescu&al94]) şi a unor cunoştinţe cu aspect statistic. A doua posibilitate este de a crea noi reguli, extrăgând din reţeaua finală, neuroni care au fost ataşaţi fără a li se asocia iniţial un simbol, dar care au dobândit pe parcursul procesului de antrenament cu cunoştinţe empirice, ponderi semnificative [Towell91]. Indiferent de stilul de utilizare, KBANN este o metodologie care arată că se pot integra cu succes cunoştinţe simbolice şi empirice (statistice), folosind o structură computaţională conexionistă relativ simplă.

1.3. Concluzii

Modelul Towell a avut succes nu numai ca "methodology demonstrator" ci şi in aplicaţii efective, de exemplu în conducerea proceselor chimice [Scott&al92] şi în cardiologie (sistemul SOAR -[Watrous&al92]). Pentru ca modelul iniţial era relativ simplist, s-au relevat o serie de probleme şi dezavantaje, în special imposibilitatea de a modifica reţeaua după ce cunoştinţele empirice au fost inserate prin antrenare neuronală. Altă problemă este folosirea reţelei ca estimator Bayesian, pornind de la un set de cunoştinţe statistice (de exemplu pacienţi pentru o aplicaţie medicală). S-a demonstrat că o astfel de reţea se comportă anormal ca estimator Bayesian în cazul în

care reţeaua are o structură predefinită şi are mai mult de două straturi (un estimator Bayesian perfect este o reţea cu numai două straturi de calcul, primul strat (cel "ascuns") având un număr cât mai mare de unităţi - infinit pentru cazul ideal - vezi [MacKay92]). Aceste probleme au fost parţial rezolvate de îmbunătăţiri ale modelului, cele mai importante fiind utilizarea unor metode statistice pentru calculul ponderilor [Koppel&al94] şi utilizarea unor algoritmi genetici pentru modificarea dinamica a topologiei reţelei prin adăugarea unor noi unităţi în timpul antrenării, ca în modelul REGENT [Opitz&Shavlik97].

Există un alt domeniu de cercetare, numit ML (machine learning), care include şi o parte din teoriile din reţele neuronale, mai exact, abilitatea lor de a stoca cunoştinţele. Metodologia KBANN şi derivatele ei au două laturi din punct de vedere al acumulării de cunoştinţe. Prima este dată de inserarea regulilor în sistem prin definirea unităţilor şi este numită într-un context mai general, *stocare constructivistă* [Kibler&Langley88] (care se poate face şi prin alte metodologii, nu neapărat cu reţele neuronale). Ceea ce este caracteristic pentru acest tip de învăţare este că cunoştinţele de plecare (*Prior Knowledge*) sunt întotdeauna reprezentate simbolic. Restul cunoştinţelor folosite pentru construirea sistemului pot fi reprezentate simbolic (dar nu sub formă de reguli), sau numeric (statistic) şi singura condiţie impusă asupra lor este translabilitatea în tipare de antrenare neuronală. Aceste cunoştinţe empirice sunt de regulă exemple de funcţionare ale sistemului dorit, fără a se cunoaşte fenomenologia care duce la comportamentul acestui sistem. Învăţarea din exemple se mai numeşte şi învăţare *inductivă* (inductive learning) [Kibler&Langley88]. Ea poate fi realizată nu numai prin metode conexioniste, existând mai multe căi posibile, parte din ele fiind deja puternic teoretizate [Dietterich&al95].

Remarcabil în modelul prezentat este faptul că stocarea constructivistă şi învăţarea inductivă sunt folosite împreună. Acest lucru este benefic, pentru ca într-o problemă tehnică complexă, suportul de cunoştinţe este în general format dintr-un set restrâns de cunoştinţe teoretice şi un set mai bogat de cunoştinţe empirice. Pentru rezolvarea problemei se folosesc în primul rând cunoştinţele teoretice existente, apoi încercându-se extragerea de noi cunoştinţe teoretice din cele empirice (acesta fiind şi scopul *in sine* al cercetării în general). Folosirea **în sistem** a cunoştinţelor empirice este un pas important, care pe de o parte simplifică munca celui care implementează sistemul şi pe de altă parte, îmbunătăţeşte robusteţea sistemului. De aceea am considerat că modelul Towell, cu toate slabiciunile sale (descrise pe larg în [Koppel&al94]) este un pas de cotitură în integrarea simbolic-conexionistă.

Am folosit modelul Towell la primele mele aplicaţii practice ale sistemelor expert cu componentă neuronală [Dragomirescu&al94], iar rezultatele au fost bune. Încercând însă să aplic acest model la probleme ceva mai complicate (tot din domeniul medical şi anume analiza procesului infecţios), am constatat o scădere drastică a performanţelor, în termeni de rezultate obţinute şi resurse computaţionale necesare. Observaţia mea a fost atunci a fost că de vină sunt:

• reţelele folosite, care sunt mult prea simple,
• paradigma că structura reţelei este cea care guvernează raţionamentul,
• structurile simbolice de tip regula "if-then", care sunt mult prea sărace pentru a captura semantica unor cunoştinţe care aveau un grad mult mai puternic de asociere şi ierarhizare.

Pornind de la aceste observaţii, am încercat să găsesc soluţii pentru aceste trei "slăbiciuni":

• pentru că "puterea" unei reţele este dată de gradul ei de conectare [Hecht90], am hotărât să folosesc reţelele complet recurente, pentru că ele au cel mai mare grad de conectare,
• structura unei reţele fiind de regulă fixă, iar raţionamentul fiind un concept dinamic, am stabilit că **starea** reţelei este cea care guvernează raţionamentul,
• am trecut la un nivel superior în materie de structuri simbolice folosite, mai exact spus, la *grafuri conceptuale* (vezi capitolul următor).

Important de notat este că noul model pe care mă pregăteam să-l dezvolt era unul eminamente unificat (având o componentă uşor hibridă - şi anume o parte translaţională). Problemele întâmpinate, descrise pe larg în capitolele următoare, au făcut ca acest model să se transforme într-unul hibrid, conform taxonomiei propuse în Introducere.

2. Structurile simbolice şi conexioniste utilizate

În cazul unui sistem bazat pe metodologia KBANN, utilizatorul trebuie să fie un bun cunoscător a două instrumente folosite des în inteligenţa artificială: sistemul expert cu mecanism de inferenţă tip forward chaining, construit din reguli propoziţionale şi reţeaua neuronală feedforward, care are de obicei un număr mic de unităţi, dar poate avea un număr mare de straturi. Pentru că în metodologia propusă de mine se folosesc două instrumente mai puţin cunoscute, care au un grad relativ mare de complexitate, am prezentat în acest capitol, pe larg, o serie de aspecte teoretice şi experimentale legate de acestea. În plus, am exprimat şi o serie de consideraţii personale, derivate nu atât din bibliografia folosită, ci mai mult din experienţa dobândită.

2.1. Grafuri conceptuale

Regulile "if-then" sunt o transcriere a cunoştinţelor exprimate în limbaj natural. Necesitatea formei lor rigide este datorată mecanismului de inferenţă folosit pentru raţionamente. Din păcate, la translaţia unor cunoştinţe din limbajul natural în reguli, se pierde foarte mult din conţinutul semantic al propoziţilor iniţiale. Din punct de vedere al reprezentării cât mai corecte a semanticii propoziţiilor, este recunoscut faptul [Allen87] [Roberts91] [Lendaris92] că cea mai buna alegere sunt grafurile conceptuale, o propunere de structura simbolică făcuită de John Sowa în 1984 [Sowa84], derivată din grafurile de dependenţă conceptuală ale lui Roger Schank [Schank81] [Luger&Stubblefield89]. Pe lânga informaţia semantică purtată de aceste structuri, se păstrează, mai ales datorită metodelor de translaţie, şi o parte din informaţia *sintactică* asociată propoziţiei. Nu vom insista asupra metodelor de translaţie, ele fiind tehnici de mult cunoscute [Riesbeck86] [Sowa95].

Poate ar fi fost util un studiu de caz pentru alegerea structurii de reprezentare

intermediare, dar există un argument foarte puternic pentru alegerea grafurilor conceptuale. Sowa [Sowa95] demonstrează că toate structurile de reprezentare a cunoştinţelor general acceptate şi utilizate pot fi transformate prin metode deterministe în grafuri conceptuale şi invers. Dar se poate observa că întotdeauna transformarea **graf conceptual → altă structură → graf conceptual**, implica o pierdere de conţinut (mai ales semantic), pierdere care se înregistrează în primul pas.

2.1.1. Reguli de construire şi utilizare

Ce este un graf conceptual?

Definiţie

Un graf conceptual este un graf finit, orientat, conectat şi bipartit. Arcele nu sunt etichetate şi nodurile sunt fie ***concepte****, fie* ***relaţii conceptuale****.*

Restricţie

Nodurile concept pot avea arce doar spre noduri relaţie, iar nodurile relaţie pot avea arce doar spre noduri concept.

Conceptele reprezintă obiecte concrete sau abstracte din universul discursului.

Definiţie

Se numeşte universul discursului, mulţimea formată din ierarhia de noţiuni, idei, cuvinte folosită pentru susţinerea unei teorii, unde numele teoriei se află în vârful acestei ierarhii (ex. discursul fizicii, discursul filozofic etc).

Pentru a avea un punct de plecare solid, ar trebui formalizăm matematic această noţiune de univers al discursului. Formalizare care ar implica o serie de definiţii topologice, pentru că relevant pentru limbajul folosit sunt relaţiile care există între cuvinte şi modul de ordonare al acestora. Dat find faptul că lucrăm cu cuvinte ale limbajului natural, acestea având în majoritatea cazurilor sensuri legate de contextul în care sunt folosite, este foarte dificil să realizăm o astfel de formalizare. O tentativă de a construi un astfel de sistem de relaţii este "dicţionarul lui Quillian" [Quillian69], care surprinde foarte multe relaţii semantice între cuvinte, modul de organizare fiind foarte simplu, dar care nu este utilizabil într-un sistem automat, fiind o structură eminamente declarativă şi deloc procedurală. Scrierea unor algoritmi care să utilizeze acest dicţionar, pentru rezolvarea unor probleme practice, s-a dovedit a fi deosebit de dificilă, modelul fiind abandonat. Interesant este de remarcat faptul că, spre deosebire de toate abordările anteriore, care insistau pe structura sintactică a informaţiei şi rolurile sintactice ale cuvintelor, acest model a abandonat complet problemele legate de sintaxă, concentrându-se numai asupra relaţiilor care există între cuvinte şi definiţii.

Motivul principal pentru care acest model nu a putut fi folosit niciodată, este că suferă de sindromul "solving the world". Autorii lui s-au gândit la o soluţie universal valabilă pentru toate aplicaţiile posibile. Datorită gradului prea mare de generalitate, modelul s-a dovedit inutilizabil în aplicaţiile practice. Din acest motiv, este indicat ca în orice aplicaţie care implică folosirea limbajul natural, modelul matematic folosit să fie pragmatic, adică să slujească idea aplicaţiei şi nu să dea soluţii generale pentru tratarea limbajului natural.

Din acest motiv, formalizarea universului discursului pe care am ales-o este gândită în aşa fel încât aceasta să fie cât mai simplu de folosit într-o aplicaţie în care structura de bază pe care o folosim sunt grafurile conceptuale.

Decizia principală în proiectarea acestui model pleacă de la următoarea observaţie: *Nodurile concept pot conţine două tipuri de concepte: generale (clase) si individuale (instanţe).*

2.1.2. Structurarea universului discursului

Sistemele expert care folosesc ca şi formă de reprezentare a cunoştinţelor, grafurile conceptuale, au un mecanism de inferenţă bazat pe regula de formare canonică "join" (compunere) [Sowa84] [Sowa95]. Spre deosebire de mecanismele de inferenţa bazate pe logică, inferenţa prin compoziţie nu oferă certitudinea păstrării riguroase a valorilor de adevăr.

Operaţia de de compunere (join) se aplică asupra a două grafuri conceptuale. Ea devine posibilă doar în cazul în care cele două grafuri conceptuale luate în considerare au cel puţin un nod concept comun. Dacă vom lua în considerare grafurile din figura 2-1a, rezultă prin compunere graful conceptual din figura 2-1b. Conceptul care este comun celor două grafuri, devine calea de legatură dintre ele, graful rezultat fiind o piesă de cunoaştere cu un conţinut semantic mai bogat decât cele două grafuri iniţiale, şi care din punct de vedere pragmatic este utilizabil în procesul eventual de generare a unor propoziţii (acestea vor avea în mod sigur sens, însă nu putem sa facem prea multe presupuneri despre valorile lor de adevăr).

Dacă grafurile conceptuale iniţiale au mai multe concepte comune, toate acestea vor fi reunite, în graful rezultat neputând apărea acelaşi concept de două ori. Se poate observa din exemplul următor (figura 2-2), că operaţia de combinare duce uneori la apariţia unor noduri relaţie redundante, fiind necesară o operaţie specială, numită simplificare. Relaţia r3 apare de două ori şi acest lucru nu este necesar.

Cazurile în care două grafuri conceptuale se pot compune nu sunt chiar atât de frecvente pe cum ar părea la prima vedere. Acest lucru este şi mai evident dacă mulţimea completă a grafurilor conceptuale folosite într-un sistem este relativ redusă, iar grafurile conţin concepte comune care nu apar des. Din acest motiv este necesară o operaţie care care să faciliteze operaţia de combinare. Pentru a introduce această

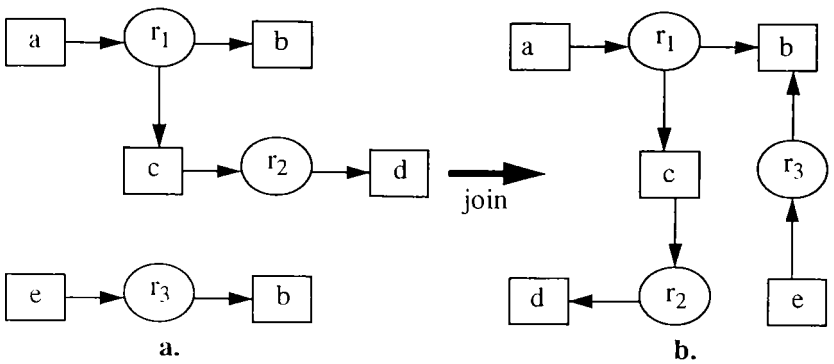


figura 2-1.

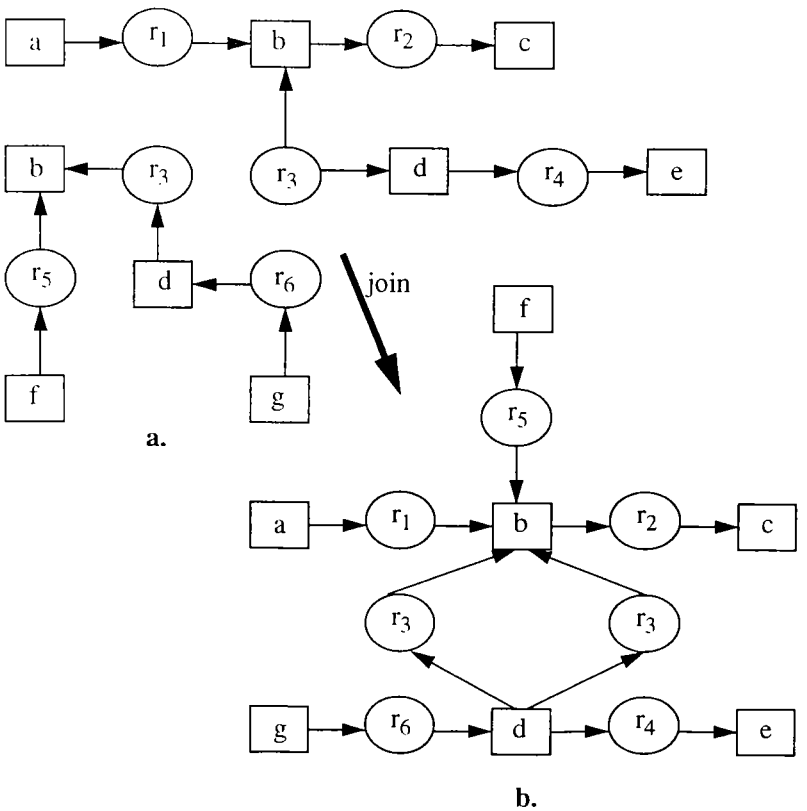


figura 2-2.

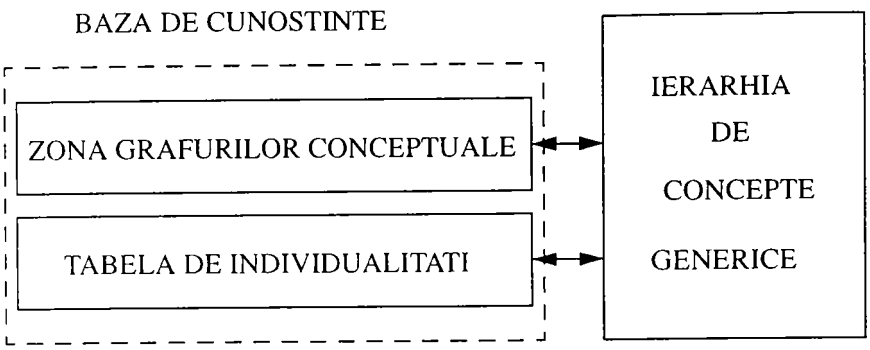


figura 2-3.

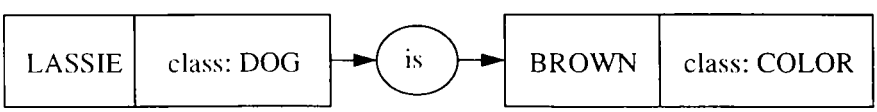


figura 2-4.

nouă operaţie, este necesară o rafinare suplimentară a nodurilor concept.

Conţinutul nodurilor concept este un simbol (şir de caractere) care denotă conceptul respectiv. Observând mulţimea conceptelor care aparţin universului discursului putem aprecia că între acestea există relaţii de moştenire de tip clasă/superclasă. Forţând această constatare, putem construi în toate cazurile practice, o ierarhie completă a conceptelor. Forţarea se datoreşte introducerii necesare a unor concepte suplimentare, (care nu aparţin iniţial universului discursului), pentru a putea crea toate lanţurile de moştenire clasă/superclasă complete.

Ceea ce urmărim este crearea unei structuri a ierarhiei de moşteniri de forma unui arbore, a cărui rădăcină să fie un supertip universal T, care nu trebuie să aiba o semantică definită. Important este faptul că fiecare concept aparţine unei anumite clase şi acest lucru este evident, pentru că în acest arbore, cu excepţia supertipului universal T, toate nodurile au unul sau mai multe noduri părinte (considerând posibil şi cazul moştenirii multiple). Operaţia nou introdusă, posibilă doar în contextul unui arbore de clase, care facilitează operaţia de combinare, se numeşte generalizare/specializare şi acţionează asupra valorilor conţinute în nodurile concept.

Dacă avem un nod concept care conţine o clasă (un concept din ierarhia de moşteniri), nod numit şi concept generic, putem instanţia această clasă, înlocuind simbolul care denotă clasa cu una din instanţele sale (specializare). Dacă avem un nod concept care conţine o instanţă a unei clase (nod numit şi concept individual), putem înlocui simbolul care îl denotă cu simbolul asociat clasei sale.

Pentru a uşura implementarea, este util să separăm conceptele individuale de cele generice. Se conturează astfel două structuri de date diferite: o tabelă de individualităţi şi o structură de concepte generice, care se poate observa că este chiar ierarhia iniţială de moşteniri, din care am eliminat toate conceptele individuale. Deci, pentru a putea efectua operaţi cu grafuri conceptuale, vom folosi schema de principiu a organizării datelor ilustrată în figura 2-3. Aceste trei zone de grupare a informaţiei care reprezintă universul discursului şi cunoştinţele înglobate în grafurile conceptuale sunt legate între ele în felul următor:

- conţinutul fiecărui nod concept aparţinând grafurilor conceptuale este format din doi pointeri: unul indicând către clasa din care face parte conceptul şi celălat indicând către o poziţie din tabela de individualităţi. Dacă nodul concept este generic şi denotă chiar clasa, atunci pointerul către tabela de individualităţi va avea o valoare specială, pe care o vom nota cu *. Pentru o implementare cât mai rapidă este util ca toate clasele care aparţin ierarhiei de moşteniri să aiba un pointer şi un număr întreg asociat către tabela de individualităţi, care să indice prima individualitate a acestei clase şi numărul de individualităţi de care această clasă dispune în universul curent al discursului. Evident că în acest caz, organizarea tabelei de individualităţi trebuie facută grupat pe clase.

O ultimă îmbunătăţire, care la prima vedere, pare redundantă, este ataşarea la

fiecare poziţie a tabelei de individualităţi a unui pointer către un nod din ierarhia de moşteniri care să indice clasa din care face parte individualitatea din poziţia respectivă. Toate acestea sunt ilustrate în exemplul următor:
- Considerăm graful conceptual din figura 2-4. În universul discursului există clasele DOG şi COLOR (concepte generice). În tabela de individualităţi există trei nume de câini: LASSIE, SCOOBY, DROOPY şi trei culori posibile: BLACK, BROWN, WHITE. O imagine parţială a structurii de date arată ca in figura 2-5. Dacă avem două grafuri conceptuale, ca în figura 2-6, putem aplica operaţia de specializare asupra conceptului generic DOG, iar în momentul în care ajungem la instanţa SCOOBY, putem combina grafurile şi vom obţine o informaţie mai completă (ca în figura 2-7a). Dacă generalizăm conceptul SCOOBY vom obţine graful din figura 2-7b care conţine mai puţină informaţie.

Putem însă ajunge şi la rezultate greşite din punct de vedere logic. Dacă vom considera cele două grafuri conceptuale din figura 2-7c, prin specializare şi combinare vom obţine graful conceptual din figura 2-7d, care reprezintă o piesă de cunoaştere falsă (pentru că avem deja o altă piesă de cunoaştere care afirmă ca LASSIE este BROWN).

2.1.3. Consistenţa logică a universului discursului

Dacă traducem, tripleţii de tipul CONCEPT-relaţie-CONCEPT care constituie grafurile conceptulale rezultate din ultima combinare, în axiome predicative şi trebuie comparăm sub forma unui test de consistenţă logică (nu trebuie să existe două axiome care se contrazic) cu toţi ceilalţi tripleţi extraşi din grafurile conceptuale existente înainte de operaţia de combinare efectuată, atunci operaţia "join" efectuată este corectă logic. Dacă apare o inconsistenţă logică, operaţia de combinare respectivă va fi abandonată.

Relaţiile care unesc conceptele pot fi şi ele standardizate, în primul rând poate fi facută o clasificare în verbe, locuţiuni şi însuşiri. Verbele pot fi la rândul lor clasificate, cea mai importantă clasificare fiind cea legată de timp (prezent, trecut, viitor, imperfect, gerunziu etc). O analiză mai serioasă a relaţiilor este realmente necesară, dar nu constituie pe moment subiectul acestei lucrări. Este o zona de interes, care trebuie în mod evident investigată, dar am considerat că este prioritar să ne concentrăm asupra problematicii conceptelor.

Totuşi o structurare primitivă a relaţiilor este necesară, astfel că în final vom avea şi un arbore al relaţiilor (conceput tot în genul unei ierarhii de moşteniri, acest lucru fiind necesar pentru un anume tip de procesare tensorială, folosită mai târziu în această lucrare). În final vom avea universul discursului formalizat în felul următor:

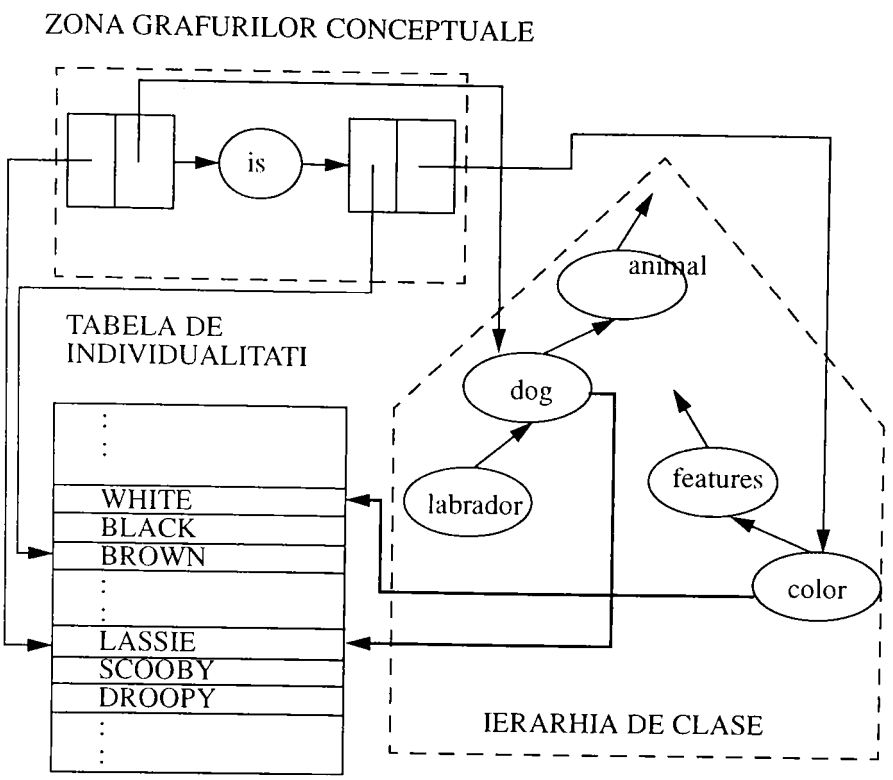


figura 2-5.

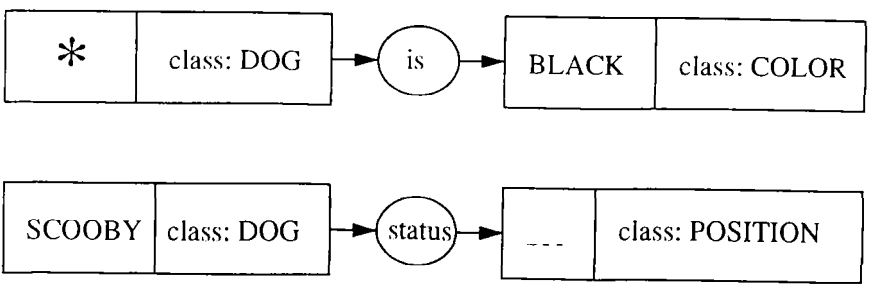
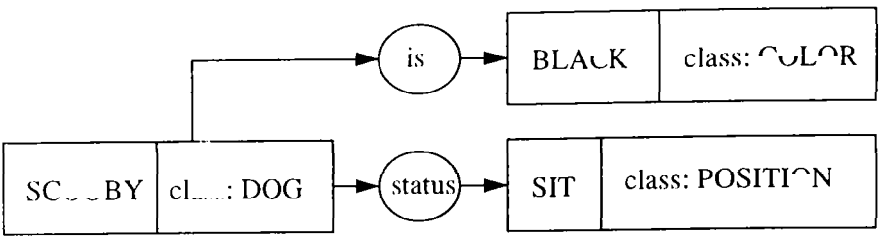
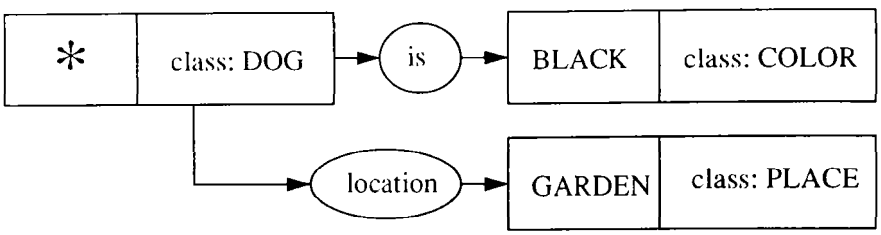


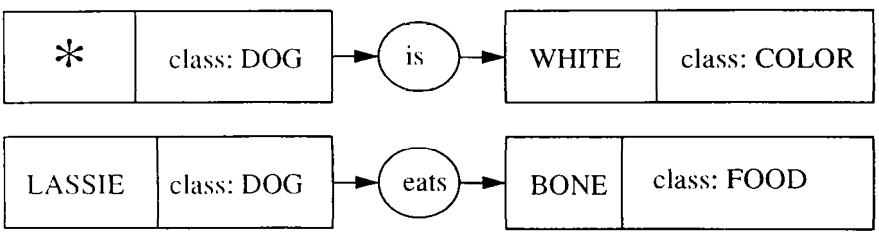
figura 2-6.



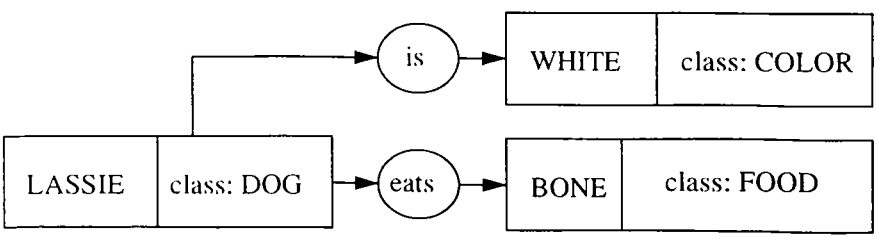
a.



b.



c.



d.

figura 2-7.

$$CG = \{\, cg \mid cg \subset Tr,\ cg \subset P \,\} \tag{2-1}$$

$$Tr = \{\, a(b,c) \mid a \in ;\ b,c \in CS \,\} \tag{2-2}$$

$$P = \{\, (pC,pI) \mid pC \rightarrow IC,\ pI \rightarrow TI \,\} \tag{2-3}$$

unde:
- R este mulţimea relaţiilor, Tr mulţimea tripleţilor, P mulţimea pointerilor.
- CS este mulţimea tuturor conceptelor.
- IC este ierarhia de clase, o mulţime parţial ordonată (sortată topologic):

$$IC = \{\, cl \mid cl \prec T,\ cl \succ \perp,\ \exists Scl \prec cl,\ \exists scl \succ cl,\ Scl, scl \in IC \} \tag{2-4}$$

- TI este mulţimea individualităţilor (tabelă ordonată şi grupată pe clase).
În plus, formalizarea mulţimii relaţiilor este dată de:

$$R = \{\, r \mid r \prec T,\ \exists sr \succ r,\ sr \in R \,\} \tag{2-5}$$

ceea ce defineşte o structură de arbore.

Clasa este la rândul ei un triplet:

$$cl(\, pI,\ n),\ \ pI \rightarrow TI,\ \ x(\, pI \,) \in IC,\ \ x(\, pI+i)_{i=0,n} \in IC \tag{2-6}$$

iar operaţia de combinare se reduce la :

$$cg1 \circ cg2 = cg3 \tag{2-7}$$

cu proprietatea că:

$$cg3T = \{\, a(b,c) \mid \exists (b \wedge c) \in (cg1 \vee cg2) \,\} \tag{2-8}$$

iar:

$$cg3P = \{\, (pC,pI) \mid (pC,pI) \in (cg1P \vee cg2P) \,\} \tag{2-9}$$

- cg3T este mulţimea tripleţilor rezultaţi din graful combinat,
- cg3P este mulţimea pointerilor,
iar graful combinat este complet determinat prin mulţimea neomogenă:

$$cg3 = cg3T \cup cg3P \tag{2-10}$$

Mulţimea rezultată din definiţia (2-8) poate fi vidă, iar în acest caz se consideră

operaţia de combinare ca fiind nereuşită.

Dacă mulţimea cg3T nu este vidă, vom aplica asupra tuturor tripleţilor ei, principiul rezoluţiei [Robinson65] pentru verificarea consistenţei. Acest principiu se poate aplica pentru grafurile conceptuale în felul următor:

1. Vom considera fiecare mulţime cg_iT ca pe o clauză multiplă de tip conjuncţie:

$$cg_iT \triangleq \bigwedge_{j=1,m_i} a_j(b_j, c_j) \qquad i=1,n \qquad (2\text{-}11)$$

unde n este numărul curent de grafuri conceptuale şi m_i numărul de tripleţi ai fiecărui graf.

2. Trecem clauzele de forma (2-11) în formă disjunctivă folosind ecuaţiile lui De Morgan.

3. Adăugăm pe rând fiecare triplet din mulţimea cg3T în formă negată la setul de ecuaţii disjunctive iniţial.

4. Dacă aplicând refutaţia prin rezoluţie (vezi ecuaţia 2-12), obţinem în final clauza vidă, înseamnă că nu avem nici o contradicţie şi operaţia de combinare verificată este consistentă logic.

$$\begin{array}{l} a_1 \vee .. \vee a_{i-1} \vee a_i \vee a_{i+1} \vee .. \vee a_f \\ b_1 \vee .. \vee b_{j-1} \vee \neg a_i \vee b_j \vee .. \vee b_f \quad \rightarrow \\ a_1 \vee .. \vee a_{i-1} \vee a_{i+1} \vee .. \vee a_f \vee b_1 \vee .. \vee b_{j-1} \vee b_j \vee .. \vee b_f \end{array} \qquad (2\text{-}12)$$

2.2. Reţele conexioniste recurente

Am avut mai multe alternative pentru alegerea unei topologii potrivite pentru componenta conexionistă a sistemului. Din start, am eliminat o clasă importantă de reţele neuronale, reţelele cu propagare înainte (feedforward). Am considerat că pentru procesarea neurosimbolică, o idee simplă este de a alege modele cât mai apropiate de modelul natural. În acest caz se pot face două observaţii.

Prima observaţie este de natură cantitativă. Creierul uman, singurul capabil în lumea vie cunoscută de a realiza activităţi de complexitatea procesarii simbolice, are în medie un număr de $2 * 10^{11}$ neuroni. Ordinul de mărime al celor mai mari reţele neuronale artificiale simulate pe cele mai puternice calculatoare actuale este de 10^5 unităţi de calcul (neuroni matematici). S-ar putea ca rezultatele pe care le vom obţine să fie alterate de această discrepanţă dimensională. În acest punct putem face o serie de presupuneri, care sunt confirmate cu un mare factor de certitudine de către

cercetătorii din domeniul neuro-psihologic [McClelland94].

a. Creierul uman foloseşte doar o mică parte a capacităţii sale pentru activităţile cognitive care ne interesează.

b. Pentru o problemă foarte simplă, "procesorul" cognitiv uman nu foloseşte întreaga sa capacitate de procesare simbolică, având o structura modulară, din componente slab cuplate, fiecare componentă fiind specializată în rezolvarea unui alt tip de problemă. Organizarea acestor componente (aşa cum sugerează şi teoria asociaţionistă, vezi [Quillian&Collins69]) este o structură ierarhică impusă de ierarhia conceptelor memorate de mintea umana.

La ora actuală acestea sunt doar presupuneri, dar deşi există posibilitatea ca ele sa fie infirmate de neuropsihologie si neurofiziologie, nici un experiment sau observaţie clinică nu le infirmă [Röbel94].

Cel mai puternic argument fizic este separaţia pe zone corticale a funcţiilor organismului. Nu vom intra în amănunte, dar un exemplu banal care se poate da, este specializarea emisferelor cerebrale pe activităţi legate de jumătatea opusa de corp. Un alt argument în favoarea acestei specializări a fost dat de comportamentul persoanelor cu "partial brain-damage", care suferă de amnezii locale [Elman90]. Aceste informaţii experimentale ne indică destul de clar localizarea unor informaţii cu semantică apropiată, în zone apropiate in creierul uman. Ca şi punct de plecare, putem considera ca plauzibilă ideea că un număr relativ mic de neuroni antrenaţi corespunzător sunt capabili să execute o procesare de nivel cantitativ mai scăzut, dar calitativ ridicată.

A doua observaţie este de natură calitativă. Modelul matematic al neuronului izolat este foarte depărtat de de modelul natural. Dar pentru o populaţie suficient de mare de neuroni artificiali comportamentul unei reţele este destul de apropiat de comportamentul unei reţele naturale. De exemplu, la un nivel simplist, retina a putut fi simulată folosind reţele artificiale. Esenţială este însă organizarea unei astfel de reţele. *Spre deosebire de creierul uman, noi putem crea virtual orice organizare posibilă.* Aici rezidă puterea reţelelor neuronale artificiale. Creierul uman în dezvoltarea sa fizică depune un efort imens şi foarte puţin eficient de a apropia zone corticale diferite prin procesul de "împachetare" a scoarţei cerebrale şi generare a circumvoluţiunilor. Zone corticale îndepărtate nu se vor putea uni niciodată, decât indirect. Lungimea axonilor fiind finită (dendritele fiind mobile), un neuron are in medie doar 10^4 vecini de care este legat direct.

2.2.1. Modele conexioniste inspirate din fizica statistică

Putem crea reţele conexioniste în care toţi neuronii să fie legaţi între ei. Iniţial toate ponderile vor avea valoarea 0, iar dacă doi neuroni vecini sunt în situaţia să se asocieze (pentru că zonele din care fac parte trebuie să se "apropie") ponderea legaturii

dintre ei se va modifica corespunzător. Pentru a implementa această idee, avem nevoie de un suport teoretic confirmat. Lucrarea de faţă nu îşi propune să caute noi modele matematice pentru arhitecturi conexioniste dedicate pentru procesarea simbolică neuronală, ci să investigheze modelele cunoscute pentru a detecta care sunt cele mai potrivite din punct de vedere al performanţelor. Este interesant de observat că modelele care vor fi prezentate în continuare au fost inspirate în special din fizică si mai puţin din biologie.

Considerăm un material cu proprietăţi magnetice, solid, cu o structură cristalină rectangulară. Atomii care compun structura cristalină sunt echidistanţi şi au toţi aceleaşi proprietăţi. Proprietatea principală este spinul magnetic. Acesta este o mărime vectorială, care indică o direcţie in spaţiu şi are lungime unitară. Pentru un astfel de material, atomii au un număr limitat de direcţii posibile pentru spinul magnetic. Un astfel de model a fost propus de Ising [Hertz&al91]. El a propus ca spinul magnetic să poată lua doar direcţii posibile, notate generic cu <1> si<-1>.

Câmpul magnetic generat de un atom va fi notat cu $\mathbf{b}_i$ şi va avea direcţia spinului său magnetic. Întreaga structură poate fi supusă unui câmp magnetic extern $\mathbf{B}_{ext}$. Pe lângă acest câmp, fiecare atom al reţelei este supus şi acţiunii combinate ale câmpurilor atomilor vecini. Factorul de influenţă dintre un atom *i* şi atomul *j* se notează cu $\mathbf{p}_{ij}$ şi depinde de poziţia relativă a celor doi atomi unul faţă de celălalt (distanţă şi unghi). Un atom este supus unei influenţe interne datorată tuturor atomilor, din reţea, influenţă care este totdeauna prezentă şi o influenţă externă care poate să lipsească. Ising a formalizat aceste fenomene prin formula:

$$\overline{b}_i' = (\sum_{i=1}^{N} \overline{p}_{ij} \cdot \overline{b}_j) + \overline{B}_{ext} \qquad (2\text{-}13)$$

Vectorul $\mathbf{b}_i'$ este rezultanta influenţei interne si externe. Se observă că o componentă a rezultantei interne este dată de valoarea "anterioară" a vectorului $\mathbf{b}_i$. Acest lucru ne obligă să dăm şi o dimensiune temporală acestei formule. Putem face acest lucru discret sau continuu. Deoarece în contextul discuţiei noastre această problemă reprezintă doar o problemă de implementare vom amâna discuţia despre factorul timp. Important este faptul că din interacţiunea acestor vectori se poate modifica spinul. Se poate demonstra că la un moment dat:

$$s_i = f(\sum_{i=1}^{N} |\overline{p}_{ij}| \cdot s_j) \qquad (2\text{-}14)$$

Unde *f* este o funcţie bipolară care trebuie în mod obligatoriu să aibă o zonă de saturaţie. Concluzia este că spinul unui atom depinde numai de spinii atomilor vecini din structură [Hertz&al91].

2.2.2. Reţelele Hopfield

Făcând o analogie cu modelul Ising, Hopfield (1982) a descris o reţea neuronală folosind aceleaşi formule. Dacă vom considera N neuroni, fiecare corespunzând cu un atom din modelul magnetic, putem folosi acelaşi principiu de transmitere a interacţiunilor. Într-o astfel de reţea, fiecare neuron îşi trimite semnalul de ieşire prin "fan-out", către toţi ceilalţi neuroni din reţea, inclusiv către el insuşi şi primeşte ca intrare suma ponderată a ieşirilor tuturor neuronilor. O astfel de reţea care nu respectă topologia clasică a perceptronului (reţeaua feedforward) şi permite şi legături înapoi, se mai numeşte şi reţea recurentă. Fiind şi o reţea complet conectată, reţeaua Hopfield se mai numeşte şi reţea recurentă complet conectată. Dacă vom nota intrarea ponderată într-un neuron i cu h_i şi ieşirea cu S_i, vom avea pentru un neuron i:

$$h_i = \sum_{j=1}^{N} w_{ij} \cdot S_j \qquad (2\text{-}15)$$

Hopfield a ales în analogie cu spinul magnetic care putea fi <1> sau <-1>, funcţia de ieşire a neuronului astfel încât ea să poată lua doar două valori: 1 şi -1. O funcţie potrivită pentru a genera o astfel de ieşire este funcţia *signum* care dă valoarea 1 pentru argumente pozitive şi -1 pentru argumente negative.

$$S_i = sgn(h_i) = \begin{cases} 1 & daca\ h_i \geq 0 \\ -1 & daca\ h_i < 0 \end{cases} \qquad (2\text{-}16)$$

Se poate observa o analogia cu formula spinului magnetic (2-14). Dacă rescriem formula vom obţine:

$$S_i = sgn(\sum_{j=1}^{N} w_{ij} \cdot S_j) \qquad (2\text{-}17)$$

O astfel de reţea reprezintă cel mai simplu model posibil pentru procesarea simbolică [Hertz&al90]. De ce? După cum am menţionat anterior, tocmai o astfel de arhitectură, unde fiecare neuron poate fi legat cu oricare alt neuron ne interesează. Se poate observa însă că numărul de legături între neuroni devine prohibitiv (N^2). Observaţia este importantă, pentru că ponderile sunt numere reale şi ne interesează şi precizia lor. În acest caz, dacă simulăm funcţionarea reţelei cu ajutorul unui program scris într-un limbaj de programare de nivel înalt, vom reprezenta aceste ponderi ca şi variabile în virgulă flotantă. Dacă am încerca să memorăm ponderile unei reţele cu un miliard de neuroni (10^9) care reprezintă cam a două suta parte din numărul de neuroni din creierul uman, am avea nevoie de $(10^9)^2=10^{18}$ variabile în virgulă flotantă, adică 8 miliarde de gigaocteţi de memorie (considerând că o variabilă în virgulă flotantă se reprezintă pe 8 octeţi).

Dar dacă ne vom baza pe consideraţiile făcute anterior, că pentru o activitate cognitivă simplă putem presupune ca suficienţi doar 10^4 neuroni adică 10^8 ponderi,

rezultă un necesar de 800 de megaocteţi, care este o valoare rezonabilă dacă vom memora extern ponderile, având în memoria RAM doar o parte din acestea. Se observă că în această situaţie toate ponderile pot fi grupate într-o matrice de dimensiune NxN, pe care o vom nota cu **W**. O linie *i* acestei matrici conţine ponderile asociate intrărilor într-un neuron de indice *i*, deci pentru calculul unei intrări ponderate h_i este suficient să cunoaştem doar linia respectivă din această matrice, şi ieşirile din toţi neuronii S_j (j=1,N). Acestea le vom grupa într-un vector S pe care îl vom numi *vector de stare* al reţelei Hopfield.

La reţelele feedforward, semnalul emis de un strat de neuroni (un vector) este transmis stratului următor, pornind de la primul strat si ajungând la ultimul, în ordinea activării straturilor. Această secvenţă de operaţii determină întotdeauna aceiaşi funcţionare a reţelei. La o reţea Hopfield, la fel ca şi la modelul Ising, apare şi factorul timp. Funcţionarea reţelei recurente poate fi formalizată dacă vom considera timpul ca o secvenţă discretă $t=0,1,2,....t_{stop}$ şi aplicăm funcţia de transfer pentru fiecare neuron astfel încât ieşirea să fie obţinută întârziată cu o cuantă de timp:

$$S_i(t+1) = sgn\left[\sum_{j=1}^{N} w_{ij} \cdot S_j(t)\right] \qquad (2\text{-}18)$$

Această relaţie se numeşte *legea de evoluţie dinamică a reţelei Hopfield discrete*. Putem rescrie relaţia vectorial:

$$\overline{S}(t+1) = SGN \times \mathbf{W} \times \overline{S}(t) \qquad (2\text{-}19)$$

Unde SGN este operatorul:

$$SGN \triangleq \begin{pmatrix} sgn(\cdot) & 0 & \dots & 0 \\ 0 & sgn(\cdot) & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & sgn(\cdot) \end{pmatrix} \qquad (2\text{-}20)$$

Din punct de vedere computaţional, o astfel de reţea are o altă utilitate decât reţelele feedforward. O reţea feedforward poate să înveţe un set de tipare duble. Un astfel de tipar este o pereche de vectori $\mathbf{i}=(i_1,...,i_n)$ si $\mathbf{d}=(d_1,...,d_m)$ unde *n* este numărul de intrări şi *m* este numărul de ieşiri (vezi capitolul anterior). După învăţare, prezentând la intrarea reţelei o componentă **i** dintr-un tipar vom obţine la ieşire un vector **o** foarte apropiat de vectorul **d** care este pereche cu vectorul de intrare din tipar. Deoarece intrarea poate fi diferită de ieşire, reţelele feedforward se mai numesc şi sisteme heteroasociative.

Reţelele Hopfield sunt reţele autoasociative. Ele pot învăţa un set de tipare singulare. Un astfel de tipar poate fi un vector N dimensional cu componente 1 si -1. Notăm baza de antrenare cu $L=\{\mathbf{X}^1,...,\mathbf{X}^p\}$. Funcţionarea reţelei Hopfield este următoarea: se ia un vector N dimensional oarecare, se transformă componentele sale

în valori binare 1 şi -1 şi se setează neuronii reţelei cu aceste valori. Considerăm această stare ca fiind asociată momentului t=0 şi o notăm cu $S(t_0)$. Aplicând legea dinamică de evoluţie, reţeaua va trece printr-o serie de stări până când va ajunge într-o stare din care nu mai poate să evolueze. Notăm această stare cu $S(t_{stop})$. De obicei se poate observa că această stare este echivalentă cu unul din tiparele din baza de învăţare. Se poate demonstra că tiparul atins la momentul t_{stop} este tiparul cel mai apropiat (considerând distanţa Hamming ca metrică pentru spaţii binare N dimensionale) de starea iniţială $S(t_0)$. Numărul de paşi ($t=1,2,...,t_{stop}$) diferă de la caz la caz.

Pentru a obţine această evoluţie, Hopfield a iniţializat ponderile reţelei (matricea **W**) aplicând o variantă a regulii de învăţare Hebbiene [Hecht90]. De exemplu dacă am avea un singur tipar de învăţare $\mathbf{X}=(X_1,...,X_N)$ formula de calcul pentru ponderi ar fi:

$$w_{ij} = \frac{1}{N} X_i \cdot X_j \qquad (2\text{-}21)$$

care semnifică următorul lucru:

a. Dacă doi neuroni au starea 1, ponderea dintre ei va fi 1/N (pozitivă sau excitatorie).

b. Dacă doi neuroni au stări diferite, ponderea este -1/N (negativă sau inhibitorie).

c. Dacă doi neuroni au starea -1, ponderea este tot excitatorie.

Dacă am avea mai multe tipare de învăţare $\mathbf{X}^u$ (u=1,p) ponderile sunt obţinute prin însumarea (superpoziţia) pentru fiecare tipar a ponderilor obţinute cu (2-21):

$$w_{ij} = \frac{1}{N} \sum_{u=1}^{p} X_i^u \cdot X_j^u \qquad (2\text{-}22)$$

Mecanismul care asigură convergenţa reţelei către un tipar învăţat poate fi explicat în felul următor. Se defineşte o funcţie de energie a reţelei [Hertz&al91]:

$$H(t) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} \cdot S_i(t) \cdot S_j(t) \qquad (2\text{-}23)$$

Se demonstrează că legea de evoluţie dinamică minimizează această funcţie. Variaţia de la un moment discret de timp la altul este:

$$\Delta H = H(t+1) - H(t) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} \cdot S_i(t+1) \cdot S_j(t+1) + \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} \cdot S_i(t) \cdot S_j(t) \qquad (2\text{-}24)$$

Demonstraţia se poate face în felul următor: considerăm că se modifică doar starea unui singur neuron *i*, $S_i(t+1) = -S_i(t)$. În acest caz (2-24) devine:

$$\Delta H=-\sum_i S_i(t+1)\sum_{j\neq i} w_{ij}\cdot S_j(t)+\sum_i S_i(t)\sum_{j\neq i} w_{ij}\cdot S_j(t) \qquad (2\text{-}25)$$

Înlocuind $S_i(t+1)$ cu $-S_i(t)$ în (2-25), vom obţine:

$$\Delta H=-\sum_i [-S_i(t)]\sum_{j\neq i} w_{ij}\cdot S_j(t)+\sum_i S_i(t)\sum_{j\neq i} w_{ij}\cdot S_j(t)=2\sum_i S_i(t)\sum_{j\neq i} w_{ij}\cdot S_j(t) \qquad (2\text{-}26)$$

Dacă vom considera matricea **W** ca fiind o matrice simetrică (şi acest lucru este asigurat prin legea de învăţare dată de (2-22)) şi în plus având toate elementele de pe diagonala principală egale cu 0 (lucru care nu afectează nici învăţarea, nici funcţionarea) putem observa că:

$$h_i(t+1) = \sum_{j\neq i} w_{ij}\cdot S_j(t) \qquad (2\text{-}27)$$

Formula finală pentru variaţia energiei va fi:

$$\Delta H = 2\sum_i S_i(t)\cdot h_i(t+1) \qquad (2\text{-}28)$$

Semnul intrării ponderate $h_i(t+1)$ va fi întotdeauna acelaşi cu semnul ieşirii neuronului $S_i(t+1)$ datorită funcţiei signum. Deci semnul termenului $S_i(t)h_i(t+1)$ va fi întotdeauna negativ pentru că $S_i(t+1) = -S_i(t)$. Energia reţelei descreşte la orice modificare de stare în reţea, care este conformă cu legea dinamică de evoluţie. Privind reţeaua dintr-o perspectivă fizică, ea reprezintă un sistem care se îndreaptă întotdeauna spre o stare de minim energetic.

Dar cel mai important fapt este următorul: *tiparele de învăţare reprezintă minime pentru funcţia de energie*. Aceasta poate fi reprezentată ca o suprafaţă în raport cu spaţiul 2^N dimensional al tuturor stărilor posibile şi este simplu de demonstrat (pornind de la formulele energiei şi ale legii dinamice de evoluţie) că punctele $\mathbf{X}^u$ (u=1,p) corespunzătoare pe această suprafaţă sunt puncte de minim, care mai sunt denumite şi atractori.

O stare iniţială oarecare $S(t_0)$ va duce la obţinerea celui mai apropiat minim de pe suprafaţa de energie. Din păcate punctele de minim corespunzătoare tiparelor de învăţare nu sunt singurele puncte de minim de pe această suprafaţă. În primul rând, se observă că pentru fiecare tipar de învăţare $\mathbf{X}^u$ (u=1,p) apare şi un minim suplimentar numit şi *minim opus* care este corespondent punctului $-\mathbf{X}^u$, care este imaginea inversată binar a vectorului de învăţare. Atingerea unui astfel de minim are următoarea semnificaţie funcţională: starea iniţială este cea mai diferită posibil de tiparul peste a cărui negată am ajuns. Un astfel de răspuns este de obicei destul de util pentru că ne indică foarte clar că starea iniţială nu seamănă deloc cu tiparele învăţate.

Slăbiciunea fundamentală a acestui model o reprezintă *stările mixate* (spurious states [Hertz&al91]). Ele apar atunci când avem mai mult de două tipare de învăţare.

De exemplu pentru trei tipare $\mathbf{X}^1$, $\mathbf{X}^2$, $\mathbf{X}^3$ vom avea:

$$X_i^{mix} = sgn(\pm X_i^1, \pm X_i^2, \pm X_i^3) \qquad (2\text{-}29)$$

Avem 8 posibilităţi pentru toate combinaţiile de semne, care vor duce la apariţia a 8 puncte de minim local, care nu corespund nici unei stări identice cu unul dintre cele trei tipare de învăţare. Deoarece prin legea de evoluţie dinamică nu putem să mergem decât în sensul descreşterii energiei (nu putem "urca" pe suprafaţa de energie) orice apropiere în funcţionarea reţelei către un minim local ne va conduce inevitabil la o fixare în această stare nedorită din care nu vom mai putea ieşi.

Concluzia pe care o putem trage este ca reţeaua Hopfield este un model computaţional care nu funcţionează întotdeauna [Hecht90] [Haykin94]. Avem nevoie de modele mai puternice care să asigure o funcţionalitate mai robustă.

2.2.3. Modelul Hopfield continuu

Sugestia lui Hopfield de a folosi în scopuri computaţionale sisteme inspirate din modele fizice, a fost studiată de specialiştii în modelarea proceselor fizice. Una din cele mai interesante abordări, este [Pineda87]. Alternativa propusă este de a studia o reţea neuronală (cu topologie Hopfield - toţi neuronii sunt legaţi între ei) ca *sistem dinamic*. Modelul matematic al sistemelor dinamice a fost folosit cu succes în magnetism, supraconductibilitate, mecanica fluidelor etc. şi este bine pus la punct din punct de vedere teoretic [Wiggins90]. O reţea neuronală privită prin prisma sistemelor dinamice neliniare (numim în acest caz reţeaua *sistem neurodinamic*) are trei proprietăţi principale:

i. *Este un sistem cu foarte multe grade de libertate.* Dimensiunea spaţiului în care evoluează starea unui sistem neurodinamic este dată de numărul de neuroni ai sistemului. Vectorul de stare al sistemului are dimensiunea N, fiecare componentă reprezentând starea momentană a unui neuron. În cazul analizat vom avea starea reprezentată de un număr real, spre deosebire de modelul Hopfield unde aveam doar două posibilităţi discrete 1 şi -1. Starea sistemului va fi un punct în spaţiul euclidian N dimensional. Concluzia interesantă ce se poate trage relativ la această proprietate este că puterea de calcul a unui astfel de sistem este dată de numărul de grade de libertate. Simplificând, am putea spune că creierul uman este o maşina computaţională deosebit de puternică pentru că are aproximativ $2*10^{11}$ grade de libertate.

ii. *Este un sistem neliniar.* Ecuaţiile care descriu starea sistemului în timp sunt obligatoriu ecuaţii neliniare. Conform demonstraţiilor lui Minsky [Minsky&Papert69] şi Smolensky [Smolensky86] un sistem neurodinamic liniar nu este o maşină computaţionala adecvată. Ea poate rezolva doar un număr limitat de probleme simple.

iii. *Este un sistem disipativ.* Sistemele dinamice disipative au proprietatea de stabilitate

asimptotică globală. Ceea ce inseamnă că sistemul se va opri într-o stare stabilă indiferent de starea iniţială din care este pornit. Modelul Hopfield care prin legea sa de evoluţie dinamică minimizează funcţia de energie a sistemului, este un sistem disipativ (dar discret).

Un sistem dinamic disipativ, neliniar, continuu, cu N grade de libertate, este de obicei descris printr-un set de N ecuaţii diferenţiale cuplate:

$$\frac{dx_i}{dt} = G_i(\bar{x}) \qquad (i=1,N) \qquad (2\text{-}30)$$

Unde x_i sunt componentele unui *vector de stare* N dimensional **x**. Fiecare stare este cuplată de celelalte stări prin funcţia neliniară G_i. Traiectoria vectorului **x** în spaţiul euclidian N dimensional se numeşte flux N dimensional. Setul de ecuaţii asigură convergenţa acestui flux într-o stare stabilă [Wiggins90].

Sistemele dinamice disipative pot avea o comportare relativ complicată în raport cu stările stabile pe care le pot atinge. O stare stabilă poate fi:

- O orbită într-un spaţiu M dimensional (M << N) numită şi atractor oscilant.
- O orbită într-un spaţiu cu dimensionalitate fracţionara (Hausdorf-Besicovitch) numită şi atractor straniu [Wiggins90].
- Un punct în spaţiul N dimensional.

Din punct de vedere computaţional ne interesează doar ultimul caz şi vom fi obligaţi să alegem doar modele matematice care asigură acest tip de convergenţă.

Dacă alegem setul de ecuaţii:

$$\frac{dx_i}{dt} = G_i(\bar{W}_i, I, \bar{x}) \qquad (i=1,N) \qquad (2\text{-}31)$$

Unde $\mathbf{W}_i$ este un vector $(w_{i1}, w_{i2},..., w_{iN})$ care reprezintă constantele de cuplare între mărimea x_i şi restul de mărimi de stare x_j (j=1,N), **I** reprezintă un set de parametrii care influenţează din exterior sistemul, va fi asigurată convergenţa sistemului într-un punct. Aceste puncte sunt soluţii ale sistemului neliniar de ecuaţii în x_i (i=1,N):

$$G_i(\bar{W}_i, I, \bar{x}) = 0 \qquad (i=1,N) \qquad (2\text{-}32)$$

Vom nota aceste soluţii cu $\mathbf{x}^{inf}$.

Un astfel de sistem de ecuaţii se rezolvă iterativ, plecând de la un vector de stare iniţial $\mathbf{x}^0$, simulând practic evoluţia sistemului în timp până într-o stare stabilă $\mathbf{x}^{inf}$. Comportamentul unui astfel de sistem neurodinamic poate fi exploatat pentru a executa două tipuri de activităţi:

• autoasociere (avem atunci o reţea Hopfield continuă)
• heteroasociere temporală

Ce este heteroasocierea temporală? O reţea feedforward obişnuită este capabilă de heteroasociere, dar aceasta se petrece independent de timp. Asocierea intrare-ieşire se face practic instantaneu. Prezentăm o intrare reţelei şi după propagarea înainte (teoretic instantanee) a semnalelor, obţinem ieşirea reţelei de pe ultimul strat. Dacă am avea o secvenţă de intrări ordonate in timp, în cazul cel mai general o funcţie vectorială de timp care generează un vector *n* dimensional ($n << N$ - numărul de grade de libertate), $f_{in}(t)=\mathbf{x}'$, am putea selecta dintre cele N stări ale sistemului, *n* stări pe care să le forţăm pe valorile $(x'_1,...,x'_n)$. Neuronii asociaţi unor astfel de stări vor reprezenta neuroni de intrare în reţea. La fel, am putea considera o submulţime **O** a componentelor vectorului de stare al sistemului $\mathbf{x}=(x_1,...,x_N)$, $\mathbf{O}=\{x''_1,x''_2,...,x''_m\}$ ($m << N$) ca reprezentând un vector de ieşire $\mathbf{x}''$. Mai simplu spus, am selectat dintre neuronii reţelei o mică parte, pentru a avea ca la reţelele feedforward o intrare şi o ieşire. Restul de neuroni îi considerăm neuroni ascunşi şi starea lor nu ne interesează în timpul funcţionării reţelei.

Discutând în termeni de sisteme dinamice, dacă funcţia $f_{in}(t)$ este o funcţie continuă (descrie un flux *n* dimensional fără discontinuităţi) atunci secvenţa continuă de valori obţinută la ieşire până la atingerea unei stări stabile va reprezenta graficul unei funcţii vectoriale $f_{out}(t)=\mathbf{x}''$. Se pune problema dacă ieşirea, care reprezintă un flux *m* dimensional, este continuă. În cazul în care cele *n* stări pe care le considerăm de intrare sunt forţate pe nişte valori impuse se pot întâmpla două evoluţii nedorite.

În primul rând, forţarea unor stări poate duce la imposibilitatea atingerii stărilor de echilibru de tip punctiform. Am spus că un astfel de fenomen este exclus la clasa de sisteme descrisă, dar stabilitatea punctiformă este respectată doar dacă sistemul este lăsat să evolueze liber, fără a modifica mărimile de stare.

În al doilea rând, chiar dacă am obţine la ieşire doar stări stabile punctiforme, succesiunea lor în timp nu ar reprezenta un flux *m* dimensional continuu. S-a demonstrat că un astfel de sistem are în mod cert o comportare discontinuă [Pineda87]. Sistemul poate fi influenţat din afară şi prin alte căi decât alterarea forţată a stărilor care sunt considerate de intrare. În ecuaţia (2-31) s-a prevăzut posibilitatea introducerii unei influenţe exterioare prin setul de mărimi **I**. Când Hopfield a încercat să construiască un model continuu plecând de la modelul său discret a gândit această mărime ca fiind echivalentă cu valorile de prag asociate neuronilor (threshold). Fiecare neuron (stare a sistemului) va avea asociată o astfel de mărime care este constantă pe timpul funcţionării reţelei şi care trebuie calculată la învăţare. Pentru funcţionarea reţelei continue, Hopfield a propus următorul model descris prin setul de ecuaţii diferenţiale:

$$\frac{du_i}{dt} = -u_i + g(u_i + I_i) \qquad (i=1,N) \qquad (2\text{-}33)$$

Unde u_i sunt intrarile asociate neuronilor. O astfel de ecuaţie poate fi interpretată în felul următor:

Variaţia infinitezimală a mărimii de stare este egală cu diferenţa dintre starea curentă u_i şi starea următoare $g(h_i)$, considerând că timpul necesar aplicării funcţiei de transfer este egal cu cuanta infinitezimală dt.

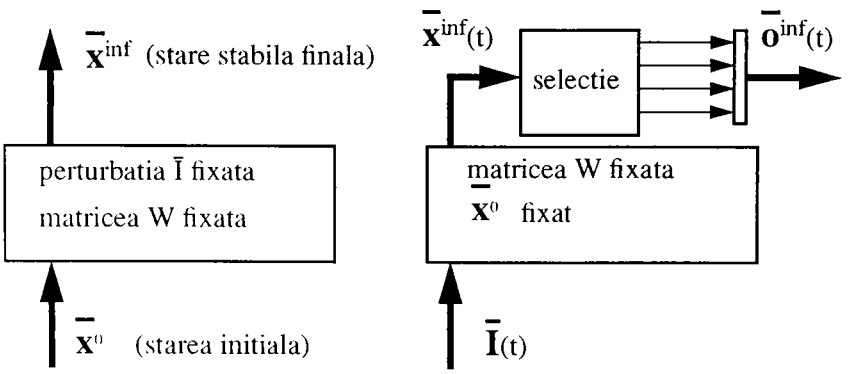
Valoarea I_i va fi intepretată ca o valoare de prag, care va decide prin depaşirea ei, în cazul fiecărui neuron dacă acesta este aprins sau nu. Această decizie este necesară pentru că de obicei la o reţea neuronală care are funcţii de transfer reale si continue (de exemplu sigmoida) nu ne interesează valorile reale pe care le au la ieşire neuronii, ci faptul dacă ei sunt aprinşi sau nu.

Pineda a dat acestor mărimi sensul de influenţă exterioară. Putem considera că intrarea într-un astfel de sistem este dată de setul I_i. Alegând o stare iniţială $\mathbf{x}_0$ şi aplicând un vector de intrare **I** de dimensiune *n* (numai unui număr limitat de stări) sistemul, neavând alterate stările din afară, va putea ajunge într-o stare stabilă punctiformă. Putem ilustra (figura 2-8) diferenţa care apare între o reţea Hopfield continuă (numită uneori şi asociatorul continuu) şi un asfel de sistem imaginat de Pineda (numit de el filtru adaptiv).

Dacă în figura 2-8a se observă că valorile **I** şi **W** sunt fixate, intrarea este o stare $\mathbf{x}_0$ iar ieşirea este o stare $\mathbf{x}^{inf}$, în cazul ilustrat în 2-8b se va fixa starea $\mathbf{x}_0$ si **W**, **I**(t) va fi intrarea, ieşirea fiind o fracţiune m-dimensională din fluxul $\mathbf{x}^{inf}(t)$. Dar pentru că scopul pe care ni l-am propus este heteroasocierea temporală, trebuie analizat cum cum se comportă un astfel de sistem în cazul în care intrarea **I** se modifică in timp. Înainte de a studia filtrul adaptiv a lui Pineda din punctul de vedere al reţelelor neuronale (implementare, reguli de învăţare) este important de făcut o analiză calitativă mai profundă a comportării acestui sistem. Pentru început trebuie explicate în detaliu câteva din proprietăţile atractorilor (punctelor de stabilitate punctiformă). Dacă sistemul descris de ecuaţiile (2-31) este lăsat să ruleze liber cu **I**=0 (fără perturbaţii externe) el se va opri într-un punct pe care îl notăm cu $\mathbf{x}^0$. De obicei pentru un sistem cu multe grade de libertate vom avea şi un număr mare de atractori (nr. atractori > N). Atractorul în care sistemul ajunge la atingerea stării de echilibru este întotdeauna determinat de starea iniţiala $\mathbf{x}_{in}$ (ca şi la reţeaua Hopfield discretă). Fiecare atractor are asociată o zonă din spaţiul euclidian N dimensional, numită *bazin de atracţie*.

Definiţie

Toate punctele $\boldsymbol{x}$ *care fiind considerate ca stări iniţiale* $\boldsymbol{x}_{in}$ *duc la evoluţia*



a. asociatorul continuu **b.** filtrul adaptiv

figura 2-8.

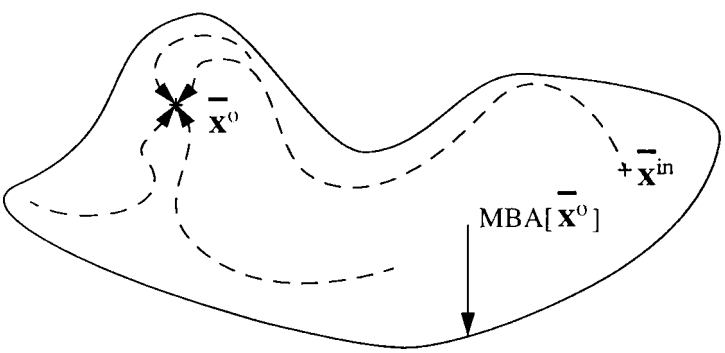


figura 2-9.

sistemului într-un atractor x^0, formează bazinul de atracţie al punctului x^0.

Aceste bazine pot avea discontinuităţi (pot fi formate din zone disjuncte ale spaţiului euclidian N dimensional) sau pot avea margini de tip fractal. Vom neglija aceste complicaţii posibile pentru că ele nu sunt relevante pentru analiza noastră. Figura 2-9 reprezintă un astfel de bazin de atracţie. Notăm cu MBA[$\mathbf{x}^0$] marginea bazinului de atracţie. Toate punctele din interiorul conturului aparţin bazinului de atracţie, cele din exterior reprezentând un alt bazin de atracţie. Dacă vom altera starea staţionară $\mathbf{x}^0$ a sistemului din acest bazin, aplicând o perturbaţie exterioară **I**, va apare un alt atractor notat cu $\mathbf{x}^{inf}(\mathbf{I})$ şi un alt bazin de atracţie de o alta formă, ca în figura 2-10. Noul atractor poate conţine (ca şi în figura 2-10) pe $\mathbf{x}^0$ în bazinul său de atracţie. Acest fenomen depinde de $\mathbf{x}^0$, de forma bazinului său de atracţie şi de **I**. Este foarte indicat să se întâmple aceasta, pentru că o perturbaţie **I** care "pierde" starea $\mathbf{x}^0$ din bazinul de atracţie al stării stabile $\mathbf{x}^{inf}(\mathbf{I})$ va duce la evoluţii nedorite, ceea ce se va vedea şi în continuare. Dacă perturbaţia exterioară **I** variază continuu şi lent (adiabatic) în timp, ne putem imagina următoarea evoluţie (timpul este în intervalul continuu $[t_0,t_{stop}]$):

i. Sistemul este stabil şi se află într-un atractor $\mathbf{x}^0$ şi este perturbat cu valoarea $\mathbf{I}(t_0)$ pe care o notăm cu $\mathbf{I}_0$.

ii. Într-un interval de timp foarte scurt (practic instantaneu) sistemul se va relaxa şi va ajunge în atractorul $\mathbf{x}^{inf}(\mathbf{I}_0)$. Putem considera că în acest timp perturbaţia **I**(t) nu îşi va modifica valoarea (va rămâne tot $\mathbf{I}_0$).

iii. Sistemul este adus din nou în starea $\mathbf{x}^0$.

iv. Se aplică o nouă valoare a perturbaţiei $\mathbf{I}_1$, care este foarte puţin diferită de $\mathbf{I}_0$.

v. Sistemul va evolua într-o stare stabilă $\mathbf{x}^{inf}(\mathbf{I}_1)$.

vi. Se va repeta secvenţa **iii-iv-v** pentru o serie de perturbaţii **I**(t) cu $t=t_1,t_2,...,t_{stop}$ unde $\mathbf{I}_j$ şi $\mathbf{I}_{j+1}$ sunt două puncte foarte apropiate în spaţiul euclidian N dimensional. La fel, t_j-t_{j-1} este un interval de timp foarte mic, dar mult mai mare decât timpul necesar relaxării sistemului într-o stare stabilă.

De obicei se va observa că punctele $\mathbf{x}^{inf}[\mathbf{I}(t)]$ vor reprezenta un flux N dimensional continuu [Wiggins90]. Cu o singură observaţie. Considerăm că MBA[$\mathbf{x}^{inf}(\mathbf{I}_j)$] îşi va schimba forma în timp ca în figura 2-11. Dacă MBA[$\mathbf{x}^{inf}(\mathbf{I}_j)$] nu traversează punctul $\mathbf{x}^0$, în mod cert fluxul N dimensional $\mathbf{x}^{inf}(\mathbf{I}_j)$ este continuu. Dacă datorită perturbaţiei **I**(t) marginea bazinului de atracţie al stării stabile curente traversează punctul $\mathbf{x}^0$ în momentul t_k, se va produce în mod cert o discontinuitate, ca în figura 2-12. O soluţie pentru evitarea acestei situaţii este ca pasul **iii** din algoritmul care generează evoluţia sistemului să fie înlocuit cu următorul:

iii'. Sistemul este lăsat în starea stabilă $\mathbf{x}^{inf}[\mathbf{I}(t_j)]$ când se aplică noua perturbaţie $\mathbf{I}_{j+1}$.

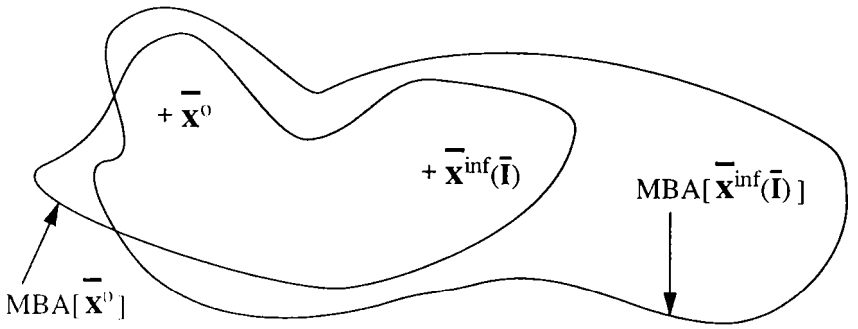


figura 2-10.

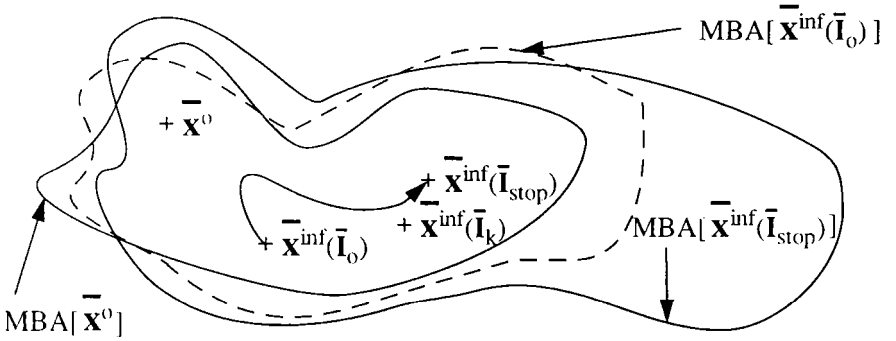


figura 2-11.

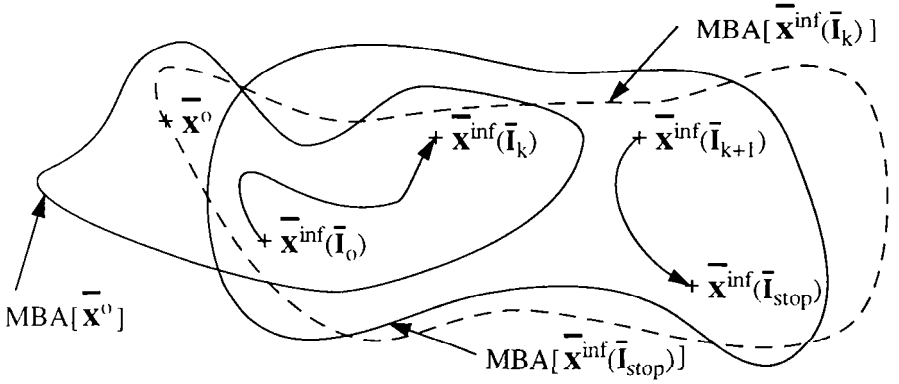


figura 2-12.

Din păcate această îmbunătăţire nu garantează că fluxul $\mathbf{x}^{inf}[\mathbf{I}(t)]$ va fi întotdeauna continuu. Dar, plecând de la această problemă se poate face o observaţie mult mai relevantă. Se observă din figura 2-12 că pentru perturbaţia $\mathbf{I}_k$ avem doi atractori diferiţi. Dar dacă lucrăm într-o zonă apropiată de o stare $\mathbf{x}^0$ care este un atractor al sistemului liber numai unul dintre aceştia poate fi atins. Numim această zonă *regiune de operare*. Alegerea acesteia este foarte importantă pentru că de ea depinde funcţionarea normală a sistemului.

Scopul iniţial pe care mi l-am propus este de a mapa peste o reţea neuronală (modelată şi implementată de exemplu de un astfel de sistem neurodinamic), un spaţiu semantic (conceptual) unde starea unor neuroni individuali sau a unor grupe de neuroni sau configuraţii ale stărilor acestora să reprezinte concepte (valori semantice). Este normal ca o regiune de operare să fie într-o relaţie de similaritate cu o regiune dintr-un spaţiu conceptual mapat peste spaţiul euclidian N dimensional.

2.2.4. Algoritmul de învăţare pentru un sistem neurodinamic

Modelul prezentat anterior se pretează cel mai bine la o implementare analogică şi nu una discretă. Este posibilă şi o implementare discretă care să simuleze evoluţia sistemului continuu descris de ecuaţiile (2-31). Ea va fi mult mai lentă dar poate da rezultate mult mai bune din punct de vedere funcţional decât alte tipuri de reţele. Deoarece factorul timp nu ne intereseaza în contextul acestei discuţii, vom prezenta algoritmul de învăţare fără a intra în detalii de implementare. Algoritmul de învăţare ce va fi prezentat este continuu şi o versiune discretizată a acestuia va fi prezentată în subcapitolul următor. Vom considera sistemul neurodinamic descris de ecuaţiile:

$$\frac{dx_i}{dt} = -x_i + g_i\left(\sum_{j=1}^{N} w_{ij} \cdot x_j + I_i\right) \qquad (i=1,N) \qquad (2\text{-}34)$$

Fiecare mărime de stare x_i (i=1,N) este asociată cu starea momentană a unui neuron. Am convenit să privim această reţea recurentă ca având o intrare şi o ieşire. O mică parte din neuronii reţelei vor reprezenta neuronii de intrare. Vom nota mulţimea lor cu **A** (card(**A**)=*n*). Mulţimea neuronilor de ieşire o vom nota cu **O** (card(**O**)=*m*). Un neuron poate fi in acelaşi timp şi de intrare şi de ieşire. Mulţimea **H** a neuronilor "ascunşi" reprezintă factorul care permite heteroascierea temporală. Pentru o funcţionare cât mai bună este indicat ca numărul neuronilor din **H** să fie mult mai mare decât numarul neuronilor din **A** şi **O**. Intrarea o vom nota ca şi la reţelele Hopfield cu X_i (i=1,N). Este util să definim funcţia:

$$\Theta_{i\Phi} = \begin{cases} 1 & \textit{daca neuronul i apartine multimii } \Phi \\ 0 & \textit{altfel} \end{cases} \qquad (2\text{-}35)$$

Intrarea în reţea este dată în (2-34) de vectorul **I**. Dacă unitatea i (i=1,N) este de intrare atunci $I_i=X_i$, altfel $I_i=0$. Acest lucru se poate ilustra prin relaţia:

$$I_i = X_i \cdot \Theta_{iA} \qquad (i=1,N) \qquad (2\text{-}36)$$

Se observă că doar *n* astfel de relaţii contează.

Vom presupune iniţial că intrările X_i sunt constante în timp. La ieşire vom dori să obţinem un set de valori notate d_i (valori dorite). Funcţionarea dorită a reţelei este dată de relaţia:

$$d_i = x_i^{\infty} \cdot \Theta_{iO} \qquad (i=1,N) \qquad (2\text{-}37)$$

La fel, doar *m* astfel de relaţii sunt nenule. Pentru a obţine o astfel de comportare a sistemului, trebuie ajustate ponderile w_{rs} (r,s=1,N) în mod corespunzător. Vom aplica metoda utilizată în algoritmul de propagare înapoi a erorilor de la reţelele feedforward. Se va minimiza o funcţie **E** care masoară distanţa euclidiană între poziţia atractorului dorit şi cea a atractorului obţinut. Funcţia este definită în modul următor:

$$E = \frac{1}{2}\sum_{i=1}^{N} J_i^2 \qquad (2\text{-}38)$$

unde:

$$J_i = (d_i - x_i^{\infty}) \cdot \Theta_{iO} \qquad (i=1,N) \qquad (2\text{-}39)$$

Vom considera că de data un set de ecuaţii similare cu (2-34) ale cărui variabile sunt elementele matricii **W**. Putem spune că matricea **W** defineşte un punct în spaţiul NxN dimensional şi trebuie deplasat acest punct astfel încât să minimizăm funcţia **E**. Traiectoria din spaţiul N^2 dimensional pe care trebuie să ne deplasăm este paralelă cu gradientul lui **E**, dar în direcţia opusă. Pentru un singur element w_{rs} al matricii **W** ecuaţia acestei mişcări este:

$$\tau_w \frac{dw_{rs}}{dt} = -\frac{\partial E}{\partial w_{rs}} \qquad (r,s=1,N) \qquad (2\text{-}40)$$

Unde τ_w este o constantă de timp (foarte mică) care defineşte deplasarea (obligatoriu mică) cu care w_{rs} se modifică.

Ecuaţiile (2-40) descriu regula de învăţare a unui astfel de sistem. Pentru a obţine performanţe mai bune am putea folosi metoda momentului de inerţie [Rumelhart&al86] ori altele care implică şi calculul derivatei de ordinul doi [Pearlmutter90], dar aceste îmbunătăţiri ţin mai mult de implementare. Componentele gradientului se calculează prin derivatele parţiale:

$$\tau_w \frac{dw_{rs}}{dt} = -\frac{1}{2}\frac{\partial}{\partial w_{rs}}\sum_{k=1}^{N} J_k^2 = \sum_{k=1}^{N} J_k \frac{\partial x_k^{\infty}}{\partial w_{rs}} \qquad (r,s=1,N) \qquad (2\text{-}41)$$

Derivata lui x_k^{inf} în raport cu w_{rs} se poate calcula ţinând cont de faptul că $\mathbf{x}^{inf}$ este o soluţie a sistemului de ecuaţii (2-32) deci satisface ecuaţia algebrică neliniară:

$$x_i^{\infty} = g_i\left(\sum_{j=1}^{N} w_{ij} \cdot x_j^{\infty} + I_i\right) \qquad (i=1,N) \tag{2-42}$$

Derivând în raport cu w_{rs} ambele părţi ale ecuaţiei vom obţine:

$$\frac{\partial x_i^{\infty}}{\partial w_{rs}} = g_i'\left(\sum_{j=1}^{N} w_{rj} \cdot x_j^{\infty} + I_r\right) \cdot \sum_{j=1}^{N} \left\{\frac{\partial w_{ij}}{\partial w_{rs}} x_j^{\infty} + w_{ij} \frac{\partial x_j^{\infty}}{\partial w_{rs}}\right\} \qquad (i=1,N) \tag{2-43}$$

şi notăm cu:

$$u_r^{\infty} = \sum_{j=1}^{N} w_{rj} \cdot x_j^{\infty} + I_r \qquad (r=1,N) \tag{2-44}$$

Pentru simplificarea notaţiei vom folosi funcţia delta a lui Kronecker:

$$K_{ij}^{\Delta} = \begin{cases} 1 & daca \quad i=j \\ 0 & daca \quad i \neq j \end{cases} \tag{2-45}$$

Se observă că:

$$\frac{\partial w_{ij}}{\partial w_{rs}} = K_{ir}^{\Delta} \cdot K_{js}^{\Delta} \tag{2-46}$$

Cu ajutorul acestei funcţii putem rescrie (2-43) în felul următor:

$$\sum_{j=1}^{N} K_{ij}^{\Delta} \frac{\partial x_j^{\infty}}{\partial w_{rs}} = g_r'(u_r^{\infty}) \left\{K_{ir}^{\Delta} \cdot x_s^{\infty} + \sum_{j=1}^{N} w_{ij} \frac{\partial x_j^{\infty}}{\partial w_{rs}}\right\} \qquad (i=1,N) \tag{2-47}$$

Aducând toate derivatele în partea stângă obţinem:

$$\sum_{j=1}^{N} \left[K_{ij}^{\Delta} - g_r'(u_r^{\infty}) \cdot W_{ij}\right] \frac{\partial x_j^{\infty}}{\partial w_{rs}} = K_{ir}^{\Delta} \cdot g_r'(u_r^{\infty}) \cdot x_s^{\infty} \qquad (i=1,N) \tag{2-48}$$

Notăm cu:

$$L_{ij} = K_{ij}^{\Delta} - g_i'(u_i^{\infty}) \cdot w_{ij} \tag{2-49}$$

Şi vom obţine ecuaţia matriceală:

$$\mathbf{L}\cdot\frac{\partial\overline{\mathbf{x}}^{\infty}}{\partial w_{rs}} = K_{ir}^{\Delta}\cdot g_r'(u_r^{\infty})\cdot x_s^{\infty} \qquad (2\text{-}50)$$

Dacă vom înmulţi această ecuaţie cu ($\mathbf{L}^{-1}$) obţinem:

$$\frac{\partial\overline{\mathbf{x}}^{\infty}}{\partial w_{rs}} = (\mathbf{L}^{-1})\cdot K_{ir}^{\Delta}\cdot g_r'(u_r^{\infty})\cdot x_s^{\infty} \qquad (2\text{-}51)$$

Putem rescrie această formulă ca şi set de ecuaţii şi pentru a avea aceiaşi notaţie ca şi în setul de ecuaţii (2-41) vom înlocui indicele *i* cu *k*:

$$\frac{\partial x_k^{\infty}}{\partial w_{rs}} = (\mathbf{L}^{-1})_{kr}\cdot g_r'(u_r^{\infty})\cdot x_s^{\infty} \qquad (k=1,\mathbf{N}) \qquad (2\text{-}52)$$

Înlocuind **L** vom obţine:

$$\tau_w\frac{dw_{rs}}{dt} = g_r'(u_r^{\infty})\cdot x_s^{\infty}\cdot\sum_{k=1}^{\mathbf{N}} J_k\cdot(\mathbf{L}^{-1})_{kr} \qquad (2\text{-}53)$$

Care este tocmai rezultatul care ne interesează. Dacă vom privi mai atent această formulă vom observa asemănarea cu formula care reglează modificarea ponderilor la reţelele feedforward prin algoritmul de învăţare prin backpropagation.

Din păcate, pentru a efectua ajustările tuturor ponderilor va trebui sa calculăm inversa unei matrici, lucru care este extrem de intensiv din punct de vedere computaţional. O metodă care evită acest calcul este următoarea. Notăm cu:

$$y_r^{\infty} = g_r'(u_r^{\infty})\cdot\sum_{k=1}^{\mathbf{N}} J_k\cdot(\mathbf{L}^{-1})_{kr} \qquad (r=1,\mathbf{N}) \qquad (2\text{-}54)$$

Şi vom obţine:

$$\tau_w\frac{dw_{rs}}{dt} = y_r^{\infty}\cdot x_s^{\infty} \qquad (r,s=1,\mathbf{N}) \qquad (2\text{-}55)$$

Care pot fi interpretate ca ecuaţiile unui sistem dinamic. Rescriem (2-54) în felul următor:

$$\sum_{r=1}^{\mathbf{N}} L_{rk}\frac{u_r^{\infty}}{g_r'(u_r^{\infty})} = J_k \qquad (k=1,\mathbf{N}) \qquad (2\text{-}56)$$

Înmulţind cu $g_k'(u_k^{inf})$ şi înlocuind cu forma explicită a lui **L** (2-49) obţinem:

$$0 = - y_k^{inf} + g_k'(u_k^{inf}) \left\{ \sum_{r=1}^{N} w_{rk} \cdot y_r^{inf} + J_k \right\} \qquad (k=1,N) \qquad (2\text{-}57)$$

Se poate observa că soluţiile y_k^{inf} ale acestui sistem liniar de ecuaţii sunt atractori ai sistemului dinamic:

$$\frac{dy_k}{dt} = -y_k + g_k'(u_k) \left\{ \sum_{r=1}^{N} w_{rk} \cdot y_r + J_k \right\} \qquad (k=1,N) \qquad (2\text{-}58)$$

Se poate observa în metoda de mai sus, calculul unei matrici inverse prin metoda relaxării. Acest algoritm de învăţare poate fi implementat avantojos şi relativ simplu într-o modalitate analogică (metoda digitală e mai complicată şi mult mai lentă [Williams&Zipser90]).

2.2.5. Algoritmul de propagare înapoi a erorilor pentru reţele recurente discrete

Deoarece o implementare analogică este o soluţie doar pentru reţele dedicate, utilizate cu precădere în controlul automat, care folosesc un număr limitat de neuroni, trebuie găsită o modalitate de a implementa metoda de învăţare prezentată anterior pe un calculator cu arhitectură clasică, mergând pe un alt principiu, pentru a evita viteza foarte scăzută de procesare. Primul pas pentru realizarea unei astfel de metode de antrenare este discretizarea factorului timp. Vom considera timpul ca o succesiune $t=1,2,....stop$. Substituirea unei mărimi continue cu una discretă va modifica evident şi modelul matematic prezentat anterior, pe care va trebui să-l reevaluăm în această nouă interpretare. Ronald Williams şi David Zipser au dezvoltat un model matematic discret plecând de la modelul lui Pineda, reuşind să stabilească şi paşii unui algoritm de învăţare foarte eficient [Williams&Zipser89] [Williams&Zipser90] [Williams&Peng90].

Al doilea pas este descrierea unei arhitecturi adecvate pentru reţeaua neuronală, care să permită o mai uşoară interpretare a secvenţelor de timp discrete. Pentru aceasta, vom considera că reţeaua are două straturi. Primul, un strat de calcul, format din N neuroni care au o funcţie de transfer neliniară $g(h_i)$, unde h_i este suma ponderată a intrărilor. Pentru simplificare vom considera că toţi neuronii au aceiaşi funcţie de transfer (de exemplu sigmoida) deşi acest lucru nu este o restricţie, fiecare neuron sau grup de neuroni putând avea funcţia sa particulară de transfer $g_i(h_i)$. Al doilea strat, este format doar din neuroni de distribuire (fan-out) care trimite fiecare către toţi cei N neuroni din stratul de calcul semnalul pe care îl primeşte la intrare. Acest semnal provine de la unul din neuronii de calcul. De exemplu, putem considera că neuronul i din stratul de calcul îşi va trimite semnalul său de ieşire către neuronul i' din stratul de distribuire.

Pentru a defini intrarea şi ieşirea din această reţea vom proceda în felul următor. Vom considera un număr de m ($m << N$) neuroni de pe stratul de calcul ca fiind neuroni de ieşire, construind vectorul de ieşire **o**, semnalul lor de ieşire fiind transmis şi neuronilor corespunzători din stratul de distribuire. La fel, putem considera şi un număr de n neuroni din stratul de distribuire ca fiind neuroni de intrare, dar pentru o mai bună claritate este preferabil sa adăugam n neuroni suplimentari stratului de distribuire. Aceştia vor avea la intrare vectorul **x** de dimensiune n care va reprezenta influenţa externă asupra acestui sistem. Pentru o mai bună funcţionare se recomandă adăugarea unui neuron pe stratul de distribuire care va avea intrarea şi ieşirile întotdeauna forţate pe -1. Existenţa acestui neuron va duce la crearea factorului de prag pentru neuronii de calcul. Arhitectura generală a unei astfel de reţele este prezentată în figura 2-13.

Se observă că va rezulta o matrice de ponderi Nx(N+n) dimensională. Vom nota cu L=N+n numărul total de neuroni din stratul de distribuire. Pentru a avea o notaţie cât mai clară vom considera că toate ieşirile din stratul de calcul sunt V_i (i=1,N), iar intrările în aceşti neuroni sunt z_j (j=1,L). Pentru a introduce şi factorul timp, considerăm că pentru un neuron oarecare i din stratul de calcul este necesară o cuantă de timp de la r la $r+1$ pentru a genera ieşirea V_i. Această ieşire este instantaneu preluată de cei N neuroni corespunzători din stratul de distribuire, care tot în mod instantaneu, o transmit către toţi neuronii de calcul. Se observă că la trecerea fiecărei cuante discrete de timp, neuronii reţelei execută în mod sincron un pas de calcul. Relaţia matematică pe care o folosim pentru a ilustra funcţionarea reţelei este următoarea:

$$V_i(t) = g(h_i(t)) = g\left(\sum_{j=0}^{L} w_{ij} \cdot z_j(t-1)\right) \qquad (i=1,N) \qquad (2\text{-}59)$$

Recurenţa reţelei este dată de faptul că intrările z_j de pe stratul de distribuire provin în parte de la neuronii de pe stratul de calcul. Numai n neuroni de pe stratul de distribuire primesc intrarea din exterior (vectorul **x**). Evident, această influenţă exterioară trebuie să fie sincronă cu funcţionarea reţelei, deci mărimea **x**, care în mod normal este o funcţie de timp şi trebuie să fie eşantionată la momentele $t=1,2,...,stop$.

O observaţie importantă este că discontinuităţi prea mari ale lui **x** pot duce la fenomenul de discontinitate a fluxului de ieşire prezentat la reţelele continue. Mărimile z_j pot fi definite în felul urmator:

$$z_j(t) = \begin{cases} -1 & daca\ j=0 \\ x_j(t) & daca\ j=1,n \\ V_{j-n}(t) & daca\ j=n+1,L \end{cases} \qquad (j=0,L) \qquad (2\text{-}60)$$

iar ieşirea reţelei poate fi dată prin relaţia:

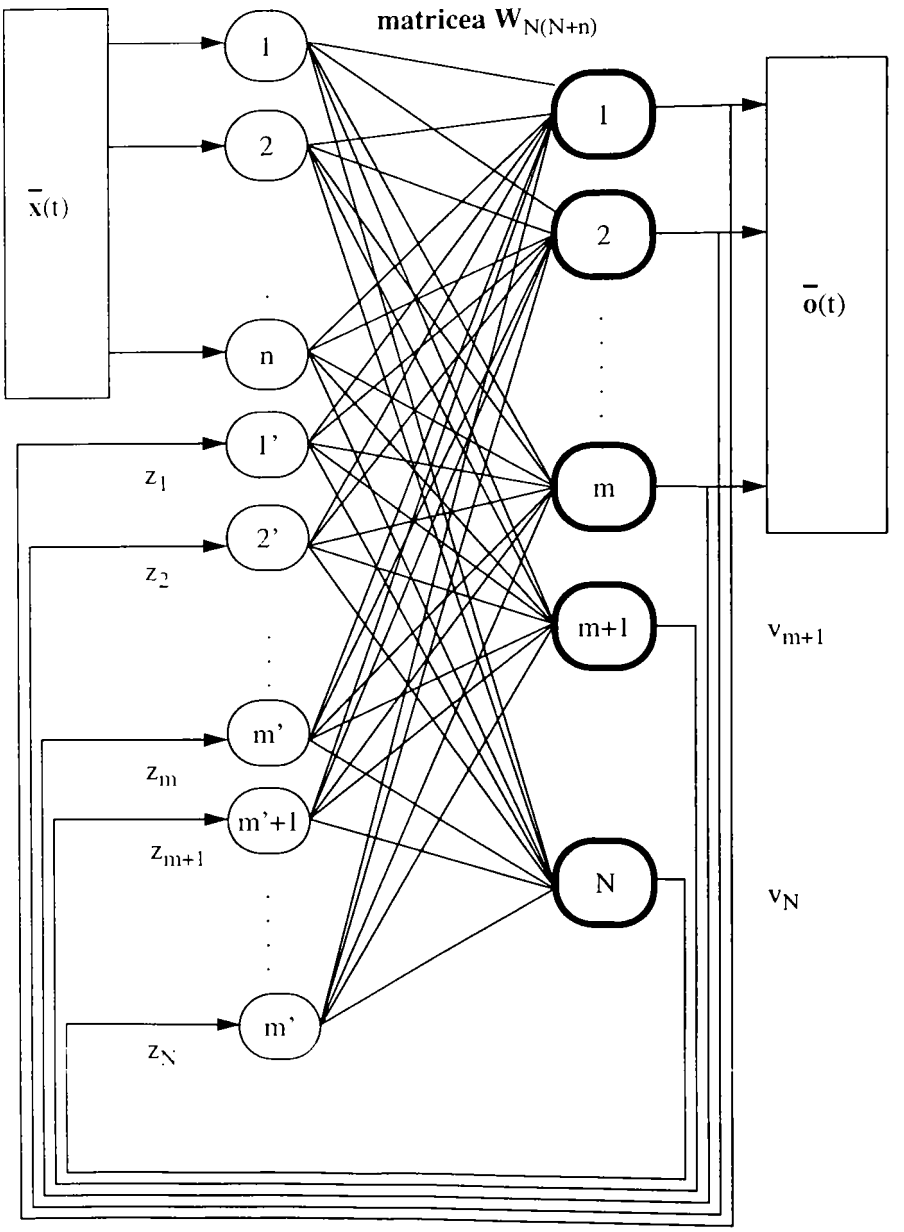


figura 2-13.

$$o_i(t) = V_i(t) \qquad (i=1,m) \qquad (2\text{-}61)$$

Aceste trei relaţii (2-59, 2-60, 2-61) sunt suficiente pentru a construi un algoritm care să simuleze funcţionarea acestui tip de reţea. Suplimentar, mai putem scrie:

$$h_i(t) = \sum_{j=0}^{L} w_{ij} \cdot z_j(t-1) \qquad (i=1,N) \qquad (2\text{-}62)$$

Deoarece reţeaua are o întârziere de o cuantă discretă de timp de la momentul aplicării intrării până la generarea ieşirii, apare următoarea problemă. Dacă la momentul 1 prezentăm reţelei intrarea **x**(1), abia la momentul 2 vom obţine o ieşire utilă **o**(2). Pentru o pornire normală este indicată următoarea procedură. Deplasăm intrarea cu o cuantă de timp înapoi ($t=0,1,2,....stop-1$) şi vom obţine ieşirile **o**(1), **o**(2), ..., **o**(stop). Se recomandă ca la momentul $t=0$ toate ieşirile V să fie forţate pe zero.

În concluzie, se poate observa că dacă vom prezenta reţelei o secvenţă de vectori ordonată in timp **x**(0), **x**(1), **x**(2), ..., **x**(stop-1) vom obţine la ieşire secvenţa de vectori **o**(t), $t=1,....stop$. Vom nota aceste secvenţe cu **X** şi **O**. O pereche (**X**,**D**) pe care dorim ca reţeaua să o poată reproduce cât mai fidel va reprezenta un tipar de învăţare. O astfel de reţea poate fi învăţată mai multe astfel de tipare {($\mathbf{X}^s$,$\mathbf{D}^s$) s=1,p}, adică o bază de învăţare.

Învăţarea trebuie să aducă elementele matricii **W** într-o stare în care $\mathbf{O}^s$ trebuie sa fie cât mai aproape de $\mathbf{D}^s$ pentru intrarea $\mathbf{X}^s$, pentru oricare $s=1,p$. Regula de învăţare se deduce într-un mod asemănător cu regula backpropagation şi regula de la reţelele recurente continue. Considerăm eroarea pătratică medie ca fiind o funcţie de ponderi:

$$E(\mathbf{W}) = \frac{1}{2}\sum_{s=1}^{p}\sum_{t=1}^{stop}\sum_{i=1}^{m}\left[d_i^s(t) - o_i^s(t)\right]^2 \qquad (2\text{-}63)$$

Pentru un singur tipar vom avea:

$$E^s(\mathbf{W}) = \frac{1}{2}\sum_{t=1}^{stop}\sum_{i=1}^{m}\left[d_i^s(t) - o_i^s(t)\right]^2 \qquad (s=1,p) \qquad (2\text{-}64)$$

Pentru fiecare tipar s vom putea calcula o ajustare a ponderii w_{ij} care se aplică în felul următor:

$$w_{ij}^{nou} = w_{ij}^{vechi} + \Delta w_{ij}^s \qquad (s=1,p) \qquad (2\text{-}65)$$

Pentru a calcula ajustarea, aplicând din nou metoda gradientului, obţinem:

$$\Delta w_{ij}^{s} = -\eta \frac{\partial E^{s}}{\partial w_{ij}} \qquad (2\text{-}66)$$

Derivata parţială a erorii se calculează în felul următor:

$$\frac{\partial E^{s}}{\partial w_{ij}} = \frac{1}{2} \sum_{t=1}^{stop} \sum_{k=1}^{m} \frac{\partial [d_k^s(t) - o_k^s(t)]^2}{\partial w_{ij}} =$$

$$= -\frac{1}{2} \sum_{t=1}^{stop} \sum_{k=1}^{m} 2 [d_k^s(t) - o_k^s(t)] \frac{\partial o_k^s(t)}{\partial w_{ij}} =$$

$$= -\sum_{t=1}^{stop} \sum_{k=1}^{m} \delta_k^s(t) \frac{\partial o_k^s(t)}{\partial w_{ij}} \qquad (i=1,N; j=1,L) \qquad (2\text{-}67)$$

unde:

$$\delta_k^s = [d_k^s(t) - o_k^s(t)] \qquad (k=1,m) \qquad (2\text{-}68)$$

Notăm cu:

$$r_{kij}^{s}(t) = \frac{\partial o_k^s(t)}{\partial w_{ij}} \qquad (2\text{-}69)$$

Această mărime reprezintă influenţa pe care o are o pondere asupra unei ieşiri o_k pentru un tipar s, la momentul t, şi se calculează în felul următor (considerăm pe i şi j ca fiind fixaţi):

$$r_{kij}^{s}(t) = \frac{\partial}{\partial w_{ij}} g(h_k^s(t)) = \frac{\partial}{\partial wij} g\left(\sum_{l=0}^{L} w_{kj} \cdot z_l^s(t-1)\right) =$$

$$= g'(h_k^s(t)) \sum_{l=0}^{L} \frac{\partial}{\partial w_{ij}} (w_{kl} \cdot z_l^s(t-1)) =$$

$$= g'(h_k^s(t)) \left[\frac{\partial w_{k0}}{\partial w_{ij}} \cdot z_1^s(t-1) + \ldots + \frac{\partial w_{kj}}{\partial w_{ij}} \cdot z_j^s(t-1) + \ldots + \sum_{l=0}^{L} w_{kl} \frac{\partial z_l^s(t-1)}{\partial w_{ij}} \right] =$$

$$=g'(h_k^s(t))\left[K_{ik}^{\Delta}\cdot z_j^s(t-1) + \sum_{l=0}^{L} w_{kl}\frac{\partial z_l^s(t-1)}{\partial w_{ij}}\right] \quad (k=1,m) \quad (2-70)$$

Al doilea termen al sumei se calculează ţinând cont de relaţia (2-60) care desparte neuronii de distribuţie alocaţi în parte pentru recurenţă şi în parte pentru aplicarea intrării x(t):

$$\sum_{l=0}^{L} w_{kl}\frac{\partial z_l^s(t-1)}{\partial w_{ij}} = \sum_{l=0}^{n} w_{kl}\frac{\partial x_l^s(t-1)}{\partial w_{ij}} + \sum_{l=n+1}^{L} w_{kl}\frac{\partial z_l^s(t-1)}{\partial w_{ij}} \quad (2-71)$$

Primul termen al sumei este egal cu zero pentru că vectorul **x** nu depinde de w_{ij}. În al doilea termen facem o schimbare a indicilor:

$$\sum_{p=1}^{N} w_{k(p+n)}\frac{\partial z_{p+n}^s(t-1)}{\partial w_{ij}} = \sum_{p=1}^{m} w_{k(p+n)}\frac{\partial o_p^s(t-1)}{\partial w_{ij}} + \sum_{p=m+1}^{N} w_{k(p+n)}\frac{\partial v_p^s(t-1)}{\partial w_{ij}} \quad (2-72)$$

Dacă vom considera că în acest caz (la învăţare), toate ieşirile sunt importante şi vom nota cu:

$$y_p^s(t) = \begin{cases} o_p^s(t) & daca\ p=1,m \\ V_p^s(t) & daca\ p=m+1,N \end{cases} \quad (2-73)$$

în acest caz membrul drept al relaţiei (2-72) va deveni egal cu:

$$\sum_{p=1}^{N} w_{k(p+n)}\frac{\partial y_p^s(t-1)}{\partial w_{ij}} \quad (2-74)$$

Putem considera că:

$$r_{pij}^s(t) = \frac{\partial y_p^s(t)}{\partial w_{ij}} \qquad (p=1,N) \quad (2-75)$$

Dacă vom rescrie relaţia (2-70) cu noua notaţie va rezulta:

$$r_{kij}^s(t) = g'(h_k^s(t))\left[K_{ik}^{\Delta}\cdot z_j^s(t-1) + \sum_{p=1}^{N} w_{k(p+n)}\cdot r_{pij}^s(t-1)\right] \qquad (k=1,N) \quad (2-76)$$

Relaţie care se observă că este recursivă relativ la timp. Pentru a calcula mărimea *r* la momentul *t*, pentru o ieşire *k*, trebuie să cunoaştem *r* pentru toate ieşirile la momentul *t-1*. Înlocuind în (2-70) vom obţine regula de învăţare:

$$\Delta w_{ij}^{s} = \eta \sum_{t=1}^{stop} \sum_{k=1}^{m} \delta_k^s(t) \cdot r_{kij}(t) \qquad (2\text{-}77)$$

Această formulă aplicată la antrenare se numeşte *regula de propagare înapoi în timp a erorilor* şi este metoda folosită în experimentele care sunt descrise în continuare în teză.

3. Realizarea legăturii între simbolic şi conexionist

Un sistem cognitiv acceptă la intrare un set de cunoştinţe, îl procesează şi generează la ieşire un alt set de cunoştinţe. Sistemul gândit de mine în prima fază, prelua un grup de propoziţii aflate într-o relaţie de ordine (reprezentate deja în formă de grafuri conceptuale) şi a le furniza în ordine unei reţele neuronale recurente. Aceasta genera la ieşire un alt set ordonat de grafuri conceptuale, care reprezentau rezultatele dorite. Perechile de seturi de propoziţii ($SET_{intrare}$, $SET_{ieşire}$) reprezentau baza de antrenare a reţelei. Două paradigme au fost enunţate în această fază:

I. *Procesul de generare al ieşirii este analog cu traiectoria stării reţelei văzută ca sistem dinamic, iar intrarea este privită ca perturbaţia exterioară aplicată sistemului.*

II. *O proiecţie a unui punct din spaţiul stărilor reţelei poate fi interpretată ca şi un graf conceptual (şi invers).*

3.1. Procesul cognitiv privit ca şi traiectorie

Paradigma care stă la baza metodologiei propuse în teză este că: un proces cognitiv poate fi realizat ca o succesiune de stări apropiate, într-un spaţiu cu foarte multe grade de libertate. Mai exact, o traiectorie, mai mult sau mai puţin continuă. Se pleacă dintr-un punct de start, dat pe de o parte de o regiune de lucru din spaţiul euclidian şi pe de altă parte de un punct iniţial care reprezintă intrarea sistemului (unul sau mai multe grafuri conceptuale). Sistemul poate evolua liber sau perturbat (de alte intrări - grafuri conceptuale), ajungând într-un punct final. Rezultatul poate fi acest punct final ori acesta plus o serie de puncte intermediare de pe traiectoria parcursă.

Ideea este cunoscută ([Rumelhart&al86] - şi fiind investigată încă din 1982) şi a dus la tentativa de realizare a raţionamentului ca traiectorie a unui sistem dinamic

[Smolensky86]. Trebuie însă precizată diferenţa între *raţionament* şi *proces cognitiv*. Un raţionament dă rezultate care sunt acceptate ca fiind adevărate din punct de vedere logic, sau măcar este indicat un grad de confidenţă (factor de certitudine, valoare de probabilitate). Termenul de proces cognitiv se referă în acest context la acea clasă de procesări simbolice care dau rezultate cu sens (ca şi operaţia de "join" de la grafurile conceptuale). Termenul este uzitat în comunitatea ştiinţifică anglo-saxonă sub numele de "common-sense reasoning" şi implică o robusteţe sporită a procesului la informaţii nerelevante sau contradictorii.

Dacă vom considera că o reţea Towell are o stare, dată de vectorul complet al ieşirilor neuronilor săi, iar reţeaua efectuează un raţionament (logic corect) prin mecanismul de "aprindere" succesiva a straturilor sale (vezi capitolul 1), iar vectorul este reprezentat într-un spaţiu euclidian cu un număr de dimensiuni egal cu numărul de unităţi din reţea, atunci şi aici putem vorbi de o traiectorie. Spaţiul folosit de reţea, se reduce la un hipercub cu latura 2, centrat în origine. Stările posibile sunt plasate aproximativ pe colţurile acestui hipercub. Atunci când un neuron îşi schimbă valoare de ieşire din aproximativ -1 în aproximativ 1 (şi putem presupune că îşi schimba continuu starea), "starea" reţelei se mută (pe o traiectorie) dintr-un colţ al hipercubului în alt colţ apropiat.

Smolensky a demonstrat formal [Smolensky86] că în cazul în care avem un spaţiu simbolic şi un spaţiu euclidian, se poate defini un izomorfism între un raţionament simbolic şi o traiectorie a unui sistem dinamic compus din unităţi conexioniste, cu condiţia ca unităţile să fie liniare. Tot în respectiva lucrare, el demonstra că izomorfismul nu e adevărat pentru unităţi neliniare. *Ceea ce doream eu să experimentez, era să antrenez o reţea neurodinamică cu unităţi neliniare (conform teoriei prezentată în capitolul 2) cu perechi de texte aflate în relaţie semantică şi să aflu dacă la prezentarea la intrare a unui text nou (format din cuvinte care au fost întâlnite toate la antrenare în alte texte), la ieşire se va obţine un text format din propoziţii care au sens, şi mai mult, dacă textul întreg formează un discurs unitar. Mai mult, doream să văd dacă elementele generice ale unui text cu semantică (introducere, dezvoltare, deznodământ) se păstrează şi în cazul unui text nou generat.*

O posibilă aplicaţie a unei astfel de reţele este prezentată în capitolul 4, şi poate fi utilizată pentru planificarea robustă a unor acţiuni. Pentru început, trebuia dovedit că o astfel de reţea poate fi antrenată, iar învăţarea va conduce la capacitatea de generalizare. Primul pas important în stabilirea metodologiei propuse a fost găsirea unei metode de translaţie a grafurilor conceptuale în vectori de activare conexionişti (puncte din spaţiul euclidian multidimensional).

3.2. Translatarea informaţiei simbolice în format conexionist

Privind o reţea neuronală ca şi sistem cognitiv, se observă imediat necesitatea definirii intrării şi ieşirii din acest sistem. Intrarea este de regulă o secvenţă de structuri

simbolice eşalonate în timp. Pentru că sistemul are o funcţionare impusă de intrare, putem considera că ieşirea este tot o secvenţă de forme abstracte. Rezultă din această structură funcţională că un astfel de sistem cognitiv este un sistem de dialog care poate fi antrenat astfel încât ca la un stimul (aparţinând unei mulţimi finite de stimuli) să dea un răspuns (aparţinând unei mulţimi finite de răspunsuri) [Wang&Waibel92]. Dacă aceste două mulţimi sunt suficient de largi şi acoperitoare pentru o anumită temă a dialogului, putem să ne folosim de capacitatea de generalizare a reţelelor neuronale pentru a furniza stimuli din acelaşi domeniu, dar care nu au fost învăţaţi şi a observa dacă răspunsurile sunt viabile. Apariţia după învăţare a unui astfel de comportament poate fi un argument experimental pentru a demonstra că metodologia folosită este potrivită.

3.2.1. Paradigma sub-simbolică.

Vom numi paradigma sub-simbolică orice structură teoretică formală care îşi propune transformarea unei forme abstracte (structură simbolică cu o formalizare exactă şi valoarea semantică asociată) într-o altă formă abstractă printr-un proces computaţional aparţinând unui model conexionist sau care are ca finalitate utilizarea informaţiei rezultate într-un sistem conexionist.

Fiind o meta-teorie, Paradigma Sub-simbolică poate fi privită ca o compunere armonioasă a mai multor aparate teoretice. Putem identifica un număr de cinci teorii necesare pentru a avea o acoperire completă a tuturor problemelor legate de această metateorie [Smolensky&al94]:

I. Reprezentarea: Este necesar un formalism matematic exact pentru a descrie o mapare a unei forme abstracte peste tiparul de activitate al unei reţele neuronale.

II. Procesarea: Trebuie descris într-un mod matematic detaliat cum procesele cognitive de nivel înalt se pot desfăşura într-un mod distribuit printr-o procesare conexionistă.

III. Formarea: Procesarea conexionistă duce în final la generarea unor noi tipare de activitate care trebuie interpretate printr-un aparat matematic astfel încât să putem reconstitui din acestea noi structuri simbolice.

IV. Gramatica: Orice structură simbolică trebuie descrisă în termeni precişi cu ajutorul limbajelor formale.

V. Semantica: Tot printr-un formalism unitar, structurilor simbolice (care au o informaţie semantică intrinsecă) trebuie să ataşăm informaţie semantică, prin definiţii şi asocieri (eventual ponderate).

Pentru a fi cât mai explicit, în continuare voi prezenta un exemplu de paradigmă sub-simbolică - reţeaua Elman - [Elman90] care s-ar preta *aparent* la scopul practic

propus. Voi descompune teoria care stă la baza reţelelor Elman relativ la cele cinci componente prezentate anterior.

I'. Reprezentarea în reţelele Elman: Intrarea într-o reţea Elman se construieşte în felul următor:
• Se stabileşte un vocabular V finit de cuvinte care vor putea fi acceptate de reţea.
• Se construieşte un strat de intrare format din card(V) neuroni de "fan-out", unde fiecare neuron va avea asociat un cuvânt din V.
• Se baleiază cu ajutorul unei ferestre de lungime mică (7 cuvinte de obicei) textul care trebuie prezentat sistemului. Lungimea ferestrei se alege astfel încât să nu poată apărea acelaşi cuvânt de doua ori în fereastră. Fereastra va avansa câte o poziţie către sfârşitul textului la fiecare moment de timp t_i (timpul este o succesiune de cuante discrete). Dacă avem de exemplu textul:

"Lucrarea mea de doctorat este realizată în mare parte în timpul meu liber şi în special duminica".

şi ne alegem o fereastră de lungime de 5 cuvinte, la momentul t_0 vom prezenta reţelei următoarea mulţime de simboluri:

$$v(t_0) = \{ \text{ lucrarea , mea , de , doctorat , este } \}$$

Vom alege din această mulţime ordonată un element pivot (de obicei cuvântul aflat în poziţie centrală). Pentru mulţimea $v(t_0)$ pivotul va fi cuvântul "de". Forţăm pe valoarea 1 neuronii de intrare asociaţi cuvintelor din mulţimea $v(t_0)$, restul rămânând setaţi pe valoarea 0. La momentul t_1 vom avea:

$$v(t_1) = \{ \text{ mea , de , doctorat , este , realizată } \}$$

cu pivotul "doctorat" şi vom seta pe valoarea 1 neuronii asociaţi cuvintelor din $v(t_1)$. Se repetă procedeul de baleiere până când se epuizează textul. Ieşirea se formează în felul următor:
- Se construieşte un strat de ieşire format tot din card(V) neuroni, având ca şi în stratul de intrare, câte un neuron asociat câte unui cuvânt din V. Dorim ca la momentul t_i, atunci când prezentăm reţelei mulţimea $v(t_i)$ cu pivotul corespunzător, la ieşire să se aprindă (să aiba valoarea cât mai aproape de 1) întotdeauna un singur neuron, cel asociat cuvântului imediat următor pivotului din $v(t_i)$, iar restul să aiba valori cât mai apropiate de 0. Această sarcină pare destul de ciudată pentru că la momentul t_i noi deja prezentăm reţelei cuvântul pe care îl dorim în acel moment la ieşire (împreună bineînţeles cu restul cuvintelor din fereastră), dar putem demonstra capacitatea de predicţie a sistemului.

II'. Procesarea în reţelele Elman: Pentru realizarea task-ului descris anterior ne alegem un set cât mai mare de texte formate pe cât este posibil din cuvinte aparţinând vocabularului V. Vom antrena reţeaua cu ajutorul acestor texte (ignorând cuvintele care

nu aparţin vocabularului ales). Elman a observat că scopul pe care şi l-a propus putea fi implementat doar cu o reţea semi-recurenta (şi nu una pur feed-forward) ca şi cea prezentată în figura 3-1.

Pentru învăţarea tiparelor care provineau din textele alese (un tipar este format dintr-o mulţime de cuvinte $v(t_i)$ pentru intrare şi un cuvânt reprezentând poziţia imediat următoare pivotului din $v(t_i)$), Elman a folosit o variantă a algoritmului backpropagation, adaptată pentru a suporta recurenţa dintre stratul ascuns si stratul de intrare. Învăţarea a fost rapid convergentă (mai ales datorită faptului că vectorii de ieşire sunt liniar independenţi) şi eroarea a putut fi scăzută sub 2% [Omlin&Giles92]. Evident că şi reţeaua Elman (la fel ca şi alte modele conexioniste) are un moment de saturaţie atunci când încercăm să o antrenăm cu prea multe tipare. Dar o alegere judicioasă a dimensiunii bazei de învăţăre duce la o funcţionare normală. La acest punct se poate observa că aparatul matematic folosit nu explică de ce reţeaua este capabilă să predicteze cuvântul care apare după pivot [Smolensky92].

III. Formarea: Metoda de a extrage informaţiile din acest sistem cognitiv este următoarea: se prezintă sistemului un singur cuvânt din vocabular şi se memorează într-o variabilă (de tip vector *m*-dimensional, cu componente reale) asociată acestui cuvânt starea indusă în stratul ascuns (care are *m* neuroni). Pentru toate cuvintele din vocabular vom obţine astfel câte un punct asociat în spaţiul euclidian *m* dimensional.

IV. Gramatica: Este un aspect ignorat în această paradigmă sub-simbolică. Singura informaţie gramaticală care este furnizează sistemului este topica (ordinea) cuvintelor în text.

V. Semantica: La fel ca şi aspectul sintactic, cel semantic este ignorat cu desăvârşire, o informaţie semantică posibilă fiind cea data de asocierile posibile între cuvinte apropiate.

Concluzia pe care o putem trage după analiza metodologiei Elman, interpretată ca şi paradigmă sub-simbolică, este că trebuie insistat mai mult pe aspectele gramaticale şi semantice care nu au fost încă suficient explorate de cercetarea în domeniul integrării procesării simbolico-conexioniste.

3.2.2. Reprezentarea intrării în reţeaua neuronală aleasă

Reţeaua neuronală pe care am ales-o are un număr de *n* neuroni de intrare (vezi figura 2-13) care au rol de "fan-out" şi pot primi şi distribui valori reale. Intrarea momentană în această reţea trebuie în mod necesar să apară ca şi un vector *n* dimensional cu componente reale. Problema este cum transformăm un text într-o succesiune de asemenea vectori. Pentru că în acestă lucrare dorim să construim o paradigmă sub-simbolică nouă, abordarea reprezentării intrării trebuie făcută intr-un mod cât se poate de formalizat, evitând soluţiile implementaţioniste (vezi critica

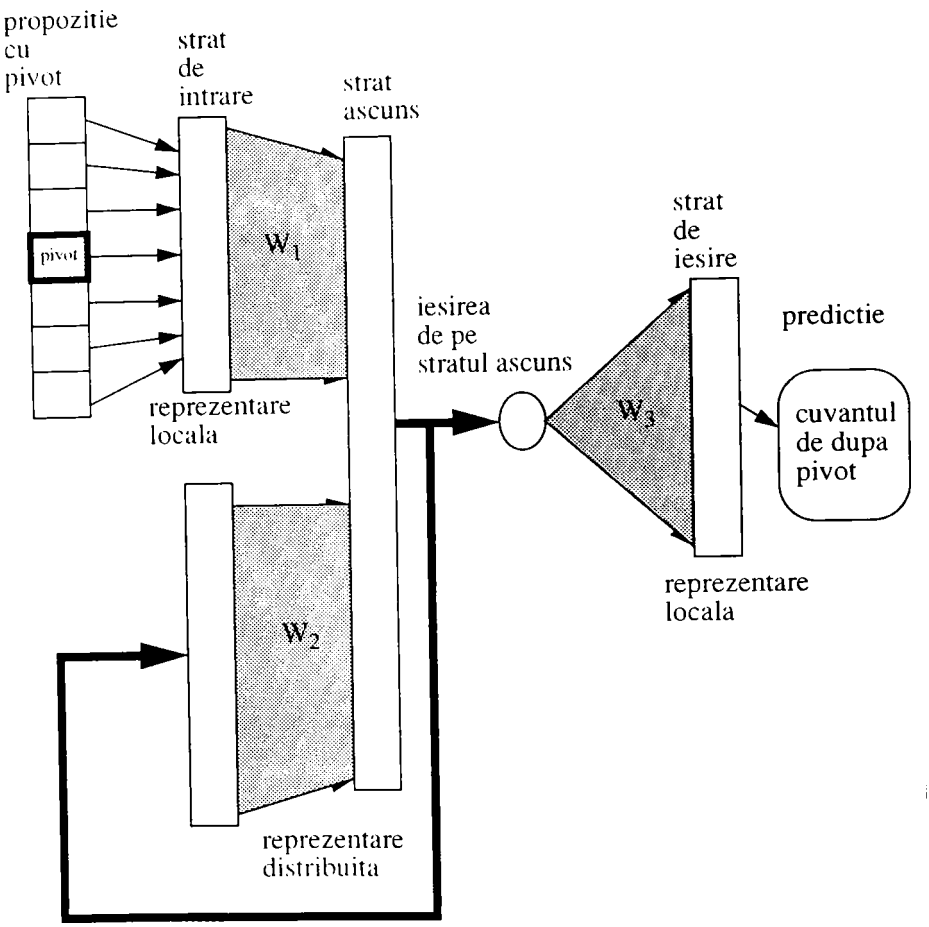


figura 3-1.

acestora în [Smolensky92]).

Reprezentările (transformarea intrării intr-o formă vectorială) pot fi clasificate în două categorii:
• locale
• distribuite.
Într-o reprezentare locală fiecărui neuron i se asociază o entitate din mulţimea entităţilor de intrare (cuvinte, eşantioane de semnal, porţiuni de imagine). De exemplu în reţeaua Elman fiecărui neuron de pe stratul de intrare i se asociază un singur cuvânt. Pentru procesarea simbolică, acest mod de reprezentare este tipic local. La fel, în reţelele Towell (un exemplu va apărea în continuare) fiecărui neuron din reţea (indiferent pe care strat se află) i se asociază un concept [Towell91].

• Majoritatea cercetătorilor (vezi analiza direcţiilor neurosimbolice din [Alexandre96]) apreciază că reprezentările locale reprezintă doar o fază de infantilism în acest domeniu şi nici o paradigmă sub-simbolică serioasă nu poate să se bazeze pe această metodă (care rămâne totuşi viabilă pentru aplicaţiile foarte simple). Se susţine (argumentat) ideea folosirii reprezentărilor *distribuite* ca fiind cele mai adecvate pentru paradigmele sub-simbolice [Ajjanagadde94].

• În acest caz o anumită entitate care trebuie reprezentată intr-un grup de neuroni nu va fi mapată peste acel grup prin aprinderea unui singur neuron asociat, ci printr-un *tipar de activitate* al acelui grup. Ca un foarte sugestiv exemplu, maparea dintre mulţimea cuvintelor din vocabularul V şi stările induse de acestea în stratul ascuns din reţeaua Elman, determină o reprezentare distribuită. Formal spus:

$$\Psi : Voc \rightarrow \boldsymbol{R}^n \qquad (3\text{-}1)$$

În cazul reţelei Elman această reprezentare se foloseşte exclusiv pentru obţinerea ieşirii finale. Pentru a descrie formal o reprezentare distribuită pentru intrarea în reţeaua aleasă voi prezenta următoarele definiţii (o varintă iniţială a lor poate fii găsită în [Smolensky&al94]:

Definiţie

Un ***tipar de activitate*** *al unei reţele neuronale (conexioniste) cu n neuroni (unităţi) este un vector n dimensional cu componente reale:*

$$\overline{v} \triangleq (v_1, v_2, \ldots, v_n) \qquad v_i \in \boldsymbol{R} \ (i=1,n) \qquad (3\text{-}2)$$

Definiţie

Spaţiul stărilor ***V*** *asociat unei reţele neuronale este zona spaţiului real n dimensional unde pot apărea tiparele de activitate ale reţelei.*

Aceasta poate fi infinită sau finită, depinzând de funcţiile de transfer ale neuronilor componenţi. De exemplu dacă toţi neuronii posedă ca şi funcţie de transfer sigmoida, această zonă va fi un hipercub *n* dimensional cu latura 1 care va avea un colţ în origine şi laturile de-a lungul axelor sau ortognale pe axe. Pentru tangenta hiperbolică, spaţiul **V** va fi un hipercub cu latura 2, centrat în origine.

Putem folosi mai multe tipuri de structuri simbolice pentru reprezentarea informaţiei (cunoştinţelor) simbolice, ca de exemplu secvenţa (string) de simboluri, arborele cu noduri conţinând simboluri etc.

Limbajul natural este format dintr-o mulţime finită de cuvinte. Dacă dorim să construim un sistem cognitiv care să manipuleze concepte denotate prin simboluri provenite din limbajul natural, acesta va putea lucra doar cu o mulţime finită de simboluri (pe care o vom nota cu F). Pentru o anumită aplicaţie particulară, F va fi o mulţime limitată ca dimensiune şi grupată din punct de vedere semantic [Winograd80]. Nu toate structurile simbolice formate cu ajutorul cuvintelor din F sunt utilizabile. După [Smolensky90] ele trebuie să îndeplinească trei condiţii:
i. Să fie corecte gramatical ("well-formedness").
ii. Să aibă sens (valoare semantica validă).
iii. Să fie adevărate (din punct de vedere empiric şi chiar logic acolo unde este posibil).

Definiţie

Se numeşte viabilă, orice structură simbolică, formată pe baza unei gramatici, având sens şi care este general acceptată ca fiind adevarată.

Definiţie

Se numeşte ***S*** *spaţiul structurilor simbolice viabile folosite pentru o anumită aplicaţie.*

În acest moment avem toate componentele necesare pentru a defini legătura dintre un text şi o reţea neuronală.

Definiţie

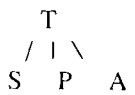
Se numeşte ***reprezentare conexionistă distribuită*** *aplicaţia:*

$$\Psi : S \rightarrow V \qquad (3-3)$$

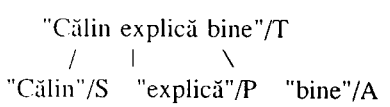
dacă aplicaţia este o bijecţie şi nici unei structuri simbolice din ***S*** *nu îi corespunde vectorul nul.*

Observaţie: Pentru o anumită alegere particulară a vectorilor din **V** reprezentarea poate fi locală.

Pentru a îndeplini cele două condiţii puse de definiţia anterioară trebuie construită o schemă de mapare între S şi V care să se bazeze pe o *regulă de decompoziţie* peste S. Pentru a defini acest concept trebuie explicat mai întâi ce inseamnă conceptul de *rol*. De exemplu într-o secvenţă care are o structură definită printr-o regulă de ordonare, fiecare poziţie poate fi privită ca o variabilă care poate lua anumite valori din mulţimea F. Deci putem interpreta structura ca o succesiune fixă de variabile numite roluri care pot fi completate cu simboluri. Pentru o secvenţă de exact trei simboluri putem construi macheta (r_1, r_2, r_3). În acest caz pentru structura simbolică particulară "a b a" se poate scrie forma (a/r_1, b/r_2, a/r_3) în care rolurile sunt completate (legate) cu valori din F [Smolenksy90]. La fel, putem construi arborii sintactici. Având de exemplu arborele:



Unde rolurile T (propoziţie), S (subiect), P (predicat), A (atribut) pot fi completate cu valori simbolice ca de exemplu:



Definiţie

Se numeşte regulă de decompoziţie aplicaţia:

$$\mu_{F\times R} : F\times R \rightarrow f/r(s) \qquad (s\in S,\ f\in F,\ r\in R) \tag{3-4}$$

unde:

- F este mulţimea simbolurilor (vocabularul)
- R este mulţimea rolurilor
- S este spaţiul structurilor viabile
- f/r(s) este o construcţie care denotă că legarea f/r este corectă pentru o anumită structură simbolică viabilă *s* (face parte din această structură sau poate face parte când structura este incompletă).

Pentru o structură simbolică *s* particulară putem avea mai multe perechi (f,r) care se potrivesc în această structură. Din acest motiv o regulă de decompoziţie poate induce aplicaţia (70), care construieşte o mulţime a tuturor perechilor (f,r) care sunt corecte pentru o anumită structură simbolică viabilă *s*.

$$\beta : S \rightarrow 2^{F\times R} \tag{3-5}$$

Deci:

$$s \mapsto \{(f,r) \mid f/r(s) \text{ corectă}\} \qquad (3\text{-}6)$$

Aplicaţia β se numeşte *reprezentare simbol-rol.* Pentru o anumită structură s_0 vom avea mulţimea finită:

$$\beta(s_0) = \{(f_1,r_1),(f_2,r_2),\ldots,(f_k,r_k)\} \qquad (3\text{-}7)$$

Construcţia $f_i/r_i(s_0)$ poate fi interpretată ca o construcţie clauzală de tip predicat(argument) unde legătura f_i/r_i poate fi interpetată ca şi un predicat logic având argumentul s_0.

Structura simbolică viabilă s_0 este din punct de vedere logic considerată adevărată, deci toate clauzele $f_i/r_i(s_0)$ le putem considera logic adevărate. În acest caz predicatul:

$$\Pi(s_0) = \bigwedge_{(f_i,r_i)\in\beta(s_0)} f_i/r_i(s_0) \qquad (3\text{-}8)$$

este adevărat [Szirbik93]. Dacă mulţimea F are cardinalitatea n putem construi foarte simplu o aplicaţie:

$$\Psi_F : F \to V_F \qquad (V_F=R^n) \qquad (3\text{-}9)$$

care mapează un simbol peste spaţiul stărilor unui grup de n neuroni. În mod necesar aplicaţia trebuie să fie o bijecţie şi nici un simbol să nu aibă asociat vectorul nul. Deci pentru fiecare simbol vom avea:

$$\Psi_F(f) = \overline{v}_f \qquad (3\text{-}10)$$

Este ideal ca mulţimea tuturor vectorilor de reprezentare $\{v_f\}$ să formeze o bază în spaţiul vectorial n dimensional (să fie liniar independenţi). În practică această condiţie este foarte greu de îndeplinit, dar putem construi această mapare astfel încât gradul de acoperire al vectorilor să fie cât mai mic.

Cel mai simplu ar fi să construim această mapare ca pe o reprezentare locală (ca şi la intrarea în reţelele Elman). Fiecărui simbol din F îi corespunde câte un neuron şi multimea vectorilor $\{v_f\}$ va reprezenta o bază. Două considerente ne împiedică să procedăm în acest mod:

i. Metodologic. S-a observat (şi în unele cazuri demonstrat teoretic) că o reprezentare distribuită este mai performantă [Smolensky90][Smolensky92].

ii. Practic.

ii-a) Numarul de neuroni n trebuie să fie în acest caz egal cu card(F). Se va observa în continuare că numărul de neuroni alocaţi acestei reprezentări influenţează stabilirea numarului de neuroni din intreaga reţea, numar care poate ajunge exagerat de mare [Maclin&Shavlik94].

ii-b) În cazul modificării lui F trebuie să adăugăm/eliminăm neuroni din reţea.

În concluzie, este mai productiv să ne stabilim din start un număr fix de neuroni (mai mic decât card(F)) şi să încercăm să creem iterativ $\{\mathbf{v}_f\}$ astfel încât orice nou simbol adăugat să fie asociat cu un vector $\mathbf{v}_f$ care diferă pe cât posibil de vectorii deja stabiliţi şi de combinaţiile lor liniare.

În acelaşi mod vom construi aplicaţia:

$$\Psi_R : R \rightarrow V_R \qquad (V_R=R^m) \tag{3-11}$$

care trebuie să indeplinească aceleaşi proprietăţi ca şi Ψ_F. Pentru rolurile care aparţin unui arbore binar se poate construi maparea într-un mod foarte elegant în felul următor [Smolensky92]: *în funcţie de adâncimea maximă a arborelui ne stabilim valoarea* ***m*** *a dimensionalităţii spaţiului vectorial* V_R *(sau altfel spus numărul de neuroni alocaţi pentru reprezentarea rolurilor).*

Stabilim calea de la rădăcină până în poziţia în care se află rolul respectiv. Parcurgem această cale construind vectorul $\mathbf{v}_r$ din componente 1 si -1 în funcţie de direcţia (stânga sau dreapta) pe care o avem de la un nod la altul. Dacă nu am ajuns la adâncimea maximă completăm restul vectorului cu 0. Un exemplu este dat în figura 2- care reprezintă un arbore binar cu adâncimea 4. Pentru r_1, r_2 şi r_3 vom avea:

$$\Psi_R(r_1) = (-1,-1,0,0) \qquad (m=4) \tag{3-12}$$

$$\Psi_R(r_2) = (1,-1,1,-1) \tag{6-13}$$

$$\Psi_R(r_3) = (1,1,1,0) \tag{3-14}$$

Smolensky a dat o soluţie şi mai elegantă, recursivă, bazată pe calculul tensorial, pentru arborii binari de adâncime teoretic infinită, mapare care se face peste un spaţiu V_R de dimensiune finită [Smolensky92].

Legătura simbol/rol o putem de asemenea reprezenta ca un tipar de activitate peste un grup de neuroni. Metoda matematică pe care o folosim este *produsul tensorial.* Dacă avem:

$$\Psi_F(f_i) = \overline{f}_i \tag{3-15}$$

şi:

$$\Psi_R(r_i) = \overline{r}_i \tag{3-16}$$

Reprezentarea legăturii f_i/r_i este:

$$\overline{f_i/r_i} = \overline{f}_i \otimes \overline{r}_i = \overline{b}_i \qquad (3\text{-}17)$$

ceea ce înseamnă următorul calcul la nivel de element. Dacă:

$$\overline{f}_i = (f_1, f_2, \ldots, f_n) \qquad (3\text{-}18)$$

$$\overline{r}_i = (r_1, r_2, \ldots, r_m) \qquad (3\text{-}19)$$

atunci tensorul b_i va fi format din toate produsele posibile dintre elementele vectorilor f_i şi r_i.

$$\overline{b}_i = (f_1 \cdot r_1, f_1 \cdot r_2, \ldots, f_1 \cdot r_m, f_2 \cdot r_1, \ldots, f_n \cdot r_m) \qquad (3\text{-}20)$$

sau altfel spus:

$$\overline{b} = \overline{f} \otimes \overline{r} \quad unde\ b_{\rho\phi} = f_\rho \cdot r_\phi \quad \rho = 1, n; \phi = 1, m \qquad (3\text{-}21)$$

Dacă Ψ_F şi Ψb_R sunt bijecţii, putem defini aplicaţia:

$$\Psi_b : F \times R \rightarrow V_F \otimes V_R \qquad (3\text{-}22)$$

$$f/r \mapsto \Psi_F(f) \otimes \Psi_R(r) \qquad (3\text{-}23)$$

care mapează legătura simbol/rol peste un grup de m x n neuroni pentru că:

$$V_R \otimes V_R = V_B \qquad (3\text{-}24)$$

unde V_B este un spaţiu vectorial cu *n x m* dimensiuni. Demonstraţia faptului că Ψ_b este tot o bijecţie este imediată.

Urmează să analizăm cum am putea combina tensorii rezultaţi din aplicaţia Ψ_b pentru a construi o reprezentare pentru predicatul $\Pi(s_0)$. Deoarece acesta este construit pe baza operatorului logic **şi**, întrebarea pe care ne-o punem în acest moment este:

- Cum se poate reprezenta în mod conexionist conjucţia logică?

Într-o reţea conexionistă, care susţine o reprezentare locală a conceptelor, conjuncţia se poate face cu ajutorul funcţiei de transfer unipolare sau chiar a unei sigmoide foarte abrupte [Towell91]. Vom da în continuare un exemplu ilustrativ. Dacă avem:

$$p_3 = p_1 \wedge p_2 \qquad (3\text{-}25)$$

$$p_1 = a_1 \wedge a_2 \wedge a_3 \qquad (3\text{-}26)$$

$$p_2 = a_4 \wedge a_5 \qquad (3\text{-}27)$$

Aceste relaţii se pot transforma într-o reţea feedforward cu mai multe straturi (ca şi reţeaua Towell prezentată în primul capitol). Dacă vom nota funcţia de transfer cu **f**, putem calcula ieşirea prin formulele:

$$o(p_1) = f(o(a_1) + o(a_2) + o(a_3)) \qquad (3\text{-}28)$$

$$o(p_2) = f(o(a_4) + o(a_5)) \qquad (2\text{-}28)$$

$$o(p_3) = f(o(p_1) + o(p_2)) = f(\sum_{i=1}^{5} o(a_i)) \qquad (3\text{-}30)$$

Rezultat care se datorează funcţiei unipolare de transfer. Observaţia pe care o putem face relativ la acest exemplu este că rezultatul se obţine printr-o superpoziţie (însumare).

Putem extinde această metodă de reprezentare conexionistă a conjuncţiei şi pentru reprezentările distribuite. Daca pentru o mulţime de propoziţii $\{p_i\}$ cu valoare de adevăr avem reprezentările distribuite:

$$\bar{r}_i = \Psi(p_i) \qquad (3\text{-}31)$$

Pentru conjuncţia lor vom avea:

$$\Psi(\bigwedge_i p_i) = \sum_i \Psi(p_i) \qquad (3\text{-}32)$$

Se observă că se păstrează proprietăţile de asociativitate şi comutativitate. Dacă în (85) înlocuim p_i cu clauzele predicative $f_i/r_i(s_0)$ vom obţine:

$$\Psi(\Pi(s_0)) = \Psi(\bigwedge_i f_i/r_i(s_0)) =$$

$$= \sum_i \Psi(f_i/r_i(s_0)) = \qquad (3\text{-}33)$$

$$= \sum_i \Psi_F(f_i) \otimes \Psi_R(r_i) \qquad (f_i, r_i) \in \beta(s_0) \qquad (3\text{-}34)$$

Şi astfel, în final putem obţine rezultatul dorit, adică reprezentarea:

$$\Psi_T : S \rightarrow V \qquad (V=V_F \otimes V_R) \tag{3-35}$$

prin descompunerea în două aplicaţii:

$$\Psi_T(s) = \Psi\Big(\bigwedge_{(f_i, r_i)\in\beta(s)} \Psi_b(f_i/r_i)\Big) = \tag{3-36}$$

$$= \sum_{(f_i, r_i)\in\beta(s)} \Psi_F(f_i) \otimes \Psi_R(r_i) = \overline{v}_s \tag{3-37}$$

Această translatare din spaţiul simbolic (conceptual) în spaţiul conexionist (euclidian) se numeşte **reprezentare tensorială**.

Observaţii:

• Dacă Ψ_F şi Ψ_R sunt bijecţii atunci şi Ψ_T este o bijecţie. Dacă $\{\mathbf{v}_f\}$ şi $\{\mathbf{v}_r\}$ sunt baze ale spaţiilor V_F şi V_R atunci si $\{\mathbf{v}_f \otimes \mathbf{v}_r\}$ vor fi o bază în spaţiul **V**. Acesta este din păcate un ideal greu de atins dar o alegere a reprezentărilor Ψ_F şi Ψ_R astfel încât să avem un grad de acoperire cât mai mic va induce aceeaşi proprietate în Ψ_T. Se poate demonstra că Ψ_T nu duce niciodată la o reprezentare prin vectorul nul, cu condiţia ca Ψ_F şi Ψ_R să nu aibă mapări de genul $\mathbf{v}_f=\mathbf{0}$ şi $\mathbf{v}_r=\mathbf{0}$.

• Dacă Ψ_F şi Ψ_R sunt reprezentări locale, Ψ_T va fi o reprezentare conexionistă tensorială locală.

• Dacă una din Ψ_F şi Ψ_R este locală şi cealaltă distribuită, Ψ_T va fi o reprezentare conexionistă tensorială semilocală.

• Dacă Ψ_F şi Ψ_R sunt distribuite, Ψ_T va fi o reprezentare conexionistă tensorială distribuită.

3.2.3. Implementarea conexionistă a reprezentării tensoriale.

Trecerea de la o structură simbolică la vectorul **v**(s) pare a fi deosebit de complicată, relativ la simplitatea modelului computaţional conexionist şi ar necesita o interfaţă masivă cu reţeaua aleasă, care din punctul de vedere al implementării ar fi mai complicată decât reţeaua însăşi. Problema poate fi destul de simplu rezolvată prin schemă de implementare dată în figura 3-2. Se observă că în acest caz vom avea m (m=card(F)) neuroni de "fan-out" care vor fi legaţi fiecare la p neuroni din zona I (de intrare) şi p (p=card(R)) neuroni de "fan-out" care vor fi legaţi fiecare la m neuroni din zona I, după schema de distribuţie din figură.

Dacă prezentăm reţelei un vector $\mathbf{b}_i$, rezultat al reprezentării Ψ_b pentru o legătură simbol/rol f_i/r_i vom aplica vectorul $\mathbf{f}_i$ peste neuronii f_j ($j=1,m$) şi vectorul $\mathbf{r}_i$ peste

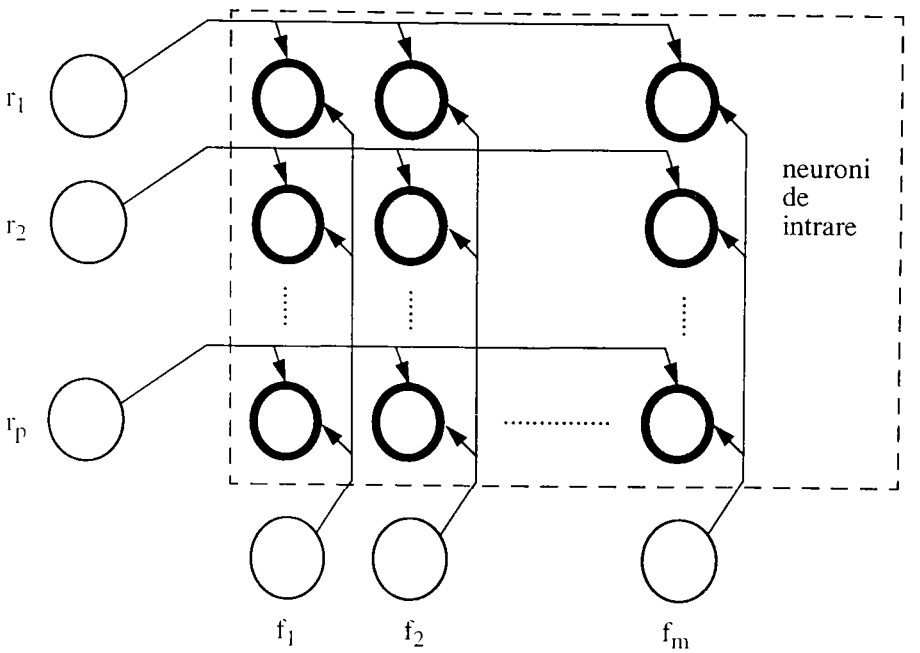


figura 3-2.

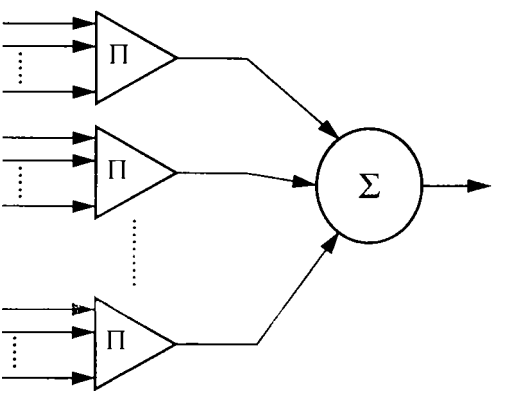


figura 3-3.

neuronii r_k ($k=1,p$). Pentru a putea implementa produsul tensorial, neuronii din zona I trebuie sa aiba următoarea schemă (relativ simplă) de funcţionare:
- Cele două intrări pe care le are fiecare neuron din zona I (una de la neuronii f, şi cealaltă de la neuronii r), trebuie înmulţite, rezultatul fiind starea de activare a neuronului respectiv.

De exemplu, pentru neuronul din poziţia $j=s$ şi $k=t$ din zona I activarea va fi dată de formula:

$$v_{st} = f_s \cdot r_t \qquad (3\text{-}38)$$

Prin această foarte simplă schemă am realizat primul pas al reprezentării tensoriale. Ea poate fi aplicată foarte simplu ca implementare software sau ca implementare hardware.

Pentru a implementa complet reprezentarea tensorială a unei structuri simbolice s mai avem nevoie de superpoziţia tuturor vectorilor $\mathbf{b}_i$ rezultaţi din reprezentarea Ψ_b a elementelor mulţimii $\beta(s)$. Implementarea se poate face în felul următor: după ce s-a calculat mulţimea $\beta(s)$ se calculează iterativ $\Psi_b(f_i/r_i)$ pentru fiecare pereche (f_i,r_i). Dupa fiecare iteraţie rezultatul este memorat aditiv de neuronii din zona I. Dacă avem $card(\beta(s))=u$ atunci vom obţine în final:

$$v_{st} = \sum_{i=1}^{u} f_s \cdot r_t \qquad (s=1,m;\ t=1,p) \qquad (3\text{-}39)$$

Formulă care reprezintă metoda practică de introducere a unei structuri simbolice într-o reţea neuronală recurentă prin intermediul a n neuroni ($n = m \times p$) de intrare.

Dacă u ar fi o valoare constantă (de obicei ea diferă de la o structură simbolică la alta, deşi s-ar putea normaliza la o valoare acoperitoare), am putea folosi pentru implementare (inclusiv hard) clasa de neuroni numită "sigma-pi". Aceasta a fost intens studiată şi există metode verificate de accelerare computaţională prin procesoare vectorizate şi/sau paralele, pentru implementările software şi VLSI analogic pentru implementările hardware [Hecht90][Beiu96]. Un astfel de neuron are funcţia de transfer:

$$v_i = \sum_{\sigma=1}^{u} w_{i\sigma} \prod_{j=1}^{z} w_{i\sigma j} \cdot I_{\sigma j} \qquad (i=1,n) \qquad (3\text{-}40)$$

unde u este numarul de vectori care trebuie însumaţi şi z numărul de intrări care trebuie înmulţite între ele. Se observă că se poate ataşa fiecărei intrări σ câte o pondere şi fiecărei intrări j câte o pondere. Schema unui astfel de neuron este dată în figura 3-2.

O reţea formată din astfel de neuroni poate implementa superpoziţia de produse tensoriale ponderate intre tensori de rang superior. În cazul nostru produsul este format întotdeauna doar din doi factori ($z=2$) f_s şi r_t. Ponderile sunt toate 1 iar u poate varia

între diferite valori. Privind schema de funcţionare a neuronului "sigma-pi" putem face observaţia că prin reprezentarea noastra folosim doar o mică parte a potenţialului computaţional oferit de astfel de neuroni şi ar fi o idee bună încercarea de a căuta noi metode care să folosească din plin resursele unor astfel de neuroni.

3.3. Metoda propusă pentru translatarea grafurilor conceptuale

Vom considera că avem pentru problema noastră, un univers al discursului structurat într-un mod prezentat în capitolul 2. Relativ la grafurile conceptuale din universul discursului vom defini trei mulţimi:

- **F** este mulţimea conceptelor individuale (fillers)
- **R** este mulţimea conceptelor generice - clasele (roles)
- **L** este mulţimea relaţiilor (links)

Mulţimea $\mathbf{S} = \mathbf{F} \cup \mathbf{R}$ este vocabularul de bază al universului discursului. Putem mapa aceste trei spaţii peste trei spaţii vectoriale, definind aplicaţiile:

$$\Psi_F : F \to V_F \qquad (3\text{-}40)$$

$$\Psi_R : R \to V_R \qquad (3\text{-}42)$$

$$\Psi_L : L \to V_L \qquad (3\text{-}43)$$

Cea mai bună reprezentare pentru aceste aplicaţii este una locală, care generează mulţimi de vectori reprezentând baze în spaţiile vectoriale $\mathbf{V_F}$, $\mathbf{V_R}$ şi $\mathbf{V_L}$. În acest caz se pot construi foarte uşor şi aplicaţiile inverse Ψ_x^{-1}.

Textul specificaţiei, care este stimulul de start, poate fi transformat într-o secvenţă de grafuri conceptuale $\mathbf{GC_1, GC_2, ..., GC_{stop}}$. O problemă foarte importantă este că între aceste propoziţii există *corelaţii de context*. Dacă vom translata fiecare graf separat într-un vector, aceste corelaţii vor fi pierdute. Soluţia este de a combina "temporal" grafurile între ele. Fiecarui graf i se va aplica operaţia canonică "join" cu toate grafurile care sunt **înaintea** lui în secvenţa temporală $\mathbf{GC_t}$ **(t=1,t-1)**. La extrem, se poate întâmpla ca ultimul graf (CG_{stop}) să fie combinat cu toate grafurile anterioare (de la *t=1* la *t=stop-1*). În [Thomason&Touretzky90] se apreciază ca redundanţa informaţiei semantice oferite unei reţele conexioniste este benefică şi cu cât această redundanţă este mai mare, cu atât reţeaua va funcţiona mai aproape de comportamentul dorit. Această operaţie de "join" are un efect benefic şi asupra traiectoriei parcurse în spaţiul stărilor [Pineda88]. Dacă pe parcursul introducerii punctelor corespondente grafurilor conceptuale, nu facem "salturi" prea mari în spaţiul vectorial al intrărilor, atunci nu vom avea "salturi" nici în spaţiul stărilor sistemului, respectiv în spaţiul

vectorial de ieşire. Dacă propoziţiile introduse ca puncte sunt apropiate ca semantică, se cere ca distanţa euclidiană între punctele corespunzătoare să fie mică. În acest spirit trebuie imaginată aplicaţia Ψ.

Vom considera o propoziţie oarecare (dupa ce asupra ei s-a aplicat "join") $\mathbf{GC_t}$. Avem o mulţime a conceptelor, a claselor şi a relaţiilor pentru acest graf. Pentru fiecare nod concept cu indicele **i**, vom calcula vectorul $\mathbf{b_i}$ (i=1,N - numărul de concepte din graful respectiv) cu formulele:

$$\overline{x_i} = \overline{f_i} \otimes \overline{r_i} \qquad (3\text{-}44)$$

unde:

$$\overline{f_i} = \Psi_F(concept) \quad şi \quad \overline{r_i} = \Psi_R(clasă) \qquad (3\text{-}45)$$

Vectorul $\mathbf{b_i}$ va reprezenta asocierea concept/clasă (filler-role binding). Pentru a reprezenta legăturile dintre un nod concept şi vecinii săi (presupunem că are $\mathbf{M_i}$ concepte vecine în graf) vom calcula:

$$\overline{rel_k} = [(\overline{f_k} \otimes \overline{r_k}) \otimes \overline{l_k}] \cdot \mathbf{cf_k} \qquad (3\text{-}46)$$

unde $\mathbf{f_k} \otimes \mathbf{r_k}$ este asocierea concept/clasă a vecinului **k** iar $\mathbf{l_k}$=Ψ(relaţia dintre conceptul i şi conceptul k). Valoarea $\mathbf{cf_k}$ este un factor de conexiune care depinde doar de direcţia arcelor care trec prin relaţia conceptuală respectivă. Dacă avem situaţia **i→relaţie→k** atunci **cf=1.0** şi dacă avem **i←relaţie←k** atunci **cf=0.5**. Mai există şi situaţii de tipul **i→relaţie←k** şi **i←relaţie→k**, dar în acest caz se va alege **cf=0.75**. După ce am calculat toţi vectorii care codifică relaţiile, le vom combina prin superpoziţie într-un singur vector:

$$\overline{srel_i} = \frac{1}{M_i} \sum_{k=1}^{M_i} \overline{rel_k} \qquad (3\text{-}47)$$

În final, pentru fiecare concept **i**, vom calcula un vector care conţine informaţie codificată despre semantica respectivului concept şi a clasei din care face parte precum şi a vecinilor săi şi măsura în care este legat de aceştia. Vom nota:

$$\overline{x_i} = \overline{x_i} \otimes \overline{srel_i} \qquad (3\text{-}48)$$

Pentru toate conceptele dintr-un graf conceptual $\mathbf{GC_t}$, prin superpoziţie vom calcula un singur vector:

$$\overline{v_t} = \sum_{i=1}^{N} \overline{b_i} \qquad (3\text{-}49)$$

care este un vector obţinut prin desfăşurarea unui tensor de gradul 5.

Se poate observa că aplicaţia tensorială propusă nu generează o bază în spaţiul vectorilor de intrare. Din acest motiv, va fi mai greu să definim o aplicaţie inversă Ψ^{-1}. Puteam defini aplicaţia Ψ în felul următor:

$$\overline{v_t} = \bigotimes_{i=1}^{N} \overline{b_i} \quad , \quad \overline{b_i} = (cf.25) \tag{3-50}$$

unde:

$$\overline{srel_i} = \bigotimes_{k=1}^{M_i} \overline{rel_k} \quad , \quad \overline{rel_k} = (cf.23) \tag{3-51}$$

dar această variantă ar avea rangul $(\mathbf{M_i(rel_k)+2(x_i)})*\mathbf{N}(v_t)$. Dacă avem doar 5 concepte şi în medie 2 vecini/concept, rangul unui astfel de tensor este 20 (mult prea mare pentru a fi fezabil). În plus, rangul tensorului (care dă şi dimensiunea vectorului de intrare) este variabil, în funcţie de numărul de concepte şi vecini. Acest lucru ar impune o reţea neuronală cu o dimensiune **maximă** pentru spaţiul vectorilor de intrare, maxim dat de cel mai voluminos graf conceptual care poate fi întâlnit în cursul raţionamentului. Mai mult, la ieşire, nu am putea şti care este dimesiunea curentă a vectorului de ieşire. Deci, se poate trage concluzia că avem nevoie de o dimensiune fixă şi prima variantă este mai bună.

Dacă avem dimesiunile $V_I=V_R=V_L=5$, pentru formula cu superpoziţie (25), dimensiunea spaţiului de intrare (şi prin consecinţă, şi a celui de ieşire) este $\mathbf{5^5=3125}$, care este enormă. Dacă pentru Ψ_x vom folosi reprezentări locale, înseamnă că vocabularul nostru e format din 5 concepte, 5 clase şi 5 relaţii, care duce la un univers al discursului atat de sărac, încât termenul de univers "este" superflu. Din acest motiv, pentru a mări vocabularul, fără a afecta dimensiunile spaţiilor de intrare/ieşire, trebuie să renunţăm la reprezentările locale pentru Ψ_x, folosind următoarele reprezentări distribuite [Szirbik&al95]:

- pentru Ψ_F vom codifica (binar de exemplu) numărul de ordine din tabela de individualităţi, pe un vector cu lungime fixa (pentru lungimea 5 vom avea 32 de individualităţi posibile).

- pentru Ψ_I, având o ierarhie de clase de adâncime fixă putem construi un vector binar de aceiaşi lungime, prin metoda "left-right" [Smolensky92] (pentru adâncimea 5 vom avea 31 de clase reprezentabile).

- pentru Ψ_L putem păstra reprezentarea locală, pentru că în general, numărul relaţiilor este mic în orice univers al discursului şi se obişnuieşte să se facă o standardizare a lor în contextul operaţiilor canonice cu grafuri conceptuale. Pentru că am descris in capitolul şi un arbore ierarhic al relaţiilor, putem aplica şi aici metoda "left-right", în caz că sunt prea multe relaţii. Se poate observa că vom avea în total, un vocabular de 68 de cuvinte.

3.4. Concluzii după prima etapă

Plecând de la cele două paradigme enunţate, putem să schiţăm o posibilă metodologie pentru realizarea unui sistem cognitiv bazat pe reţele neurodinamice şi translatare tensorială. Un astfel de sistem va primi la intrare un set de propoziţii, care sunt transformate în puncte în spaţiul intrărilor reţelei, formând o traiectorie. Această intrare va fi percepută ca o perturbaţie, reţeaua fiind iniţializată într-un atractor cu o regiune de lucru care conţine proiecţia intrării la nivel de spaţiu total al stărilor. Reţeaua va evolua pe o traiectorie, spaţiul de ieşire conţinând o proiecţie a traiectoriei complete, care este compusă din puncte care vor fi traduse în grafuri conceptuale.

Se poate observa că procesarea efectivă se face exclusiv prin reţeaua neuronală. Translatarea grafurilor conceptuale în vectori de activare neuronală (şi invers) este o sarcină auxiliară, care nu afectează absolut deloc evoluţia intrinsecă a sistemului. Din acest motiv, un astfel de sistem s-ar încadra din punctul de vedere al clasificării propuse în introducere, în clasa sistemelor neurosimbolice unificate. La prima vedere ar părea un sistem hibrid translaţional, dar în aceste sisteme, translaţia cunoştinţelor are şi rol de procesare suplimentară, pe când aici nu apare aşa ceva. Mai mult, după cum s-a prezentat în subcapitolul 3.2.3 translatarea tensorială poate fi implementată (în sensul graf→vector) prin o serie de arhitecturi conexioniste, care deşi nu sunt uniforme cu reţeaua recurentă folosită, se bazează pe aceiaşi paradigmă computaţională.

Se va vedea însă că un astfel de model este simplist şi are o serie de manifestări care nu permit funcţionarea sa corectă. *Soluţia este una hibridă, în care pe lângă procesarea pur neuronală, au loc şi procesări pur simbolice.* Capitolele următoare prezintă argumentele pentru susţinerea aceste afirmaţii.

Partea II-a
Dezvoltarea modelului propus

4. Modelul hibrid translaţional

Metodologia care foloseşte pentru realizarea unui proces cognitiv simbolic o reţea neurodinamică şi o schemă de translaţie tensorială a avut nevoie de o confirmare practică prin implementarea unei aplicaţii de test. Găsirea aceste aplicaţii s-a dorit a fi un proces de raţionament, în genul în care o reţea Towell poate efectua raţionamente. Aplicaţia pe care am intenţionat să o dezvolt, era bazată pe un *model în buclă închisă* (vezi mai jos), care presupunea dezvoltarea a două reţele neurodinamice şi a interfeţelor simbolice aferente, cu o aplicaţie imediată în planificare. Sarcina s-a dovedit a fi mult prea complexă, dezvoltându-se doar o singură reţea, dar o serie de rezultate şi observaţii s-au dovedit valoroase pentru dezvoltarea modelului pe alte căi (prezentate în capitolele 5 şi 6).

4.1. Modelul cognitiv conexionist în buclă închisă

Acest model cognitiv a fost propus de Rumelhart, Smolensky, McClelland şi Hinton [Rumelhart&al86]. El este constituit din două reţele neuronale (a căror topologie nu este precizată - dar din punctul meu de vedere sunt reţele neurodinamice) legate în buclă închisă. Se consideră că un sistem cognitiv, primeşte stimuli din mediul înconjurător, pe care îi analizează şi sintetizează, luând apoi decizii care sunt aplicate asupra mediului. O serie secvenţializată de stimuli va duce la generarea unei secvenţe de decizii. Dacă aplicăm asupra sistemului toţi stimulii pe rând, culegând din ieşire deciziile generate, fără să luăm în considerare influenţa lor imediată asupra mediului, vom comite o eroare. Sistemul este lipsit de "feed-back", deciziile fiind generate fără să se ţină cont de modificările pe care acestea le aduc asupra mediului. Necunoscând starea curentă a mediului, putem lua decizii greşite. (vezi figura 4-1)

Soluţia este de a modifica stimulii în raport cu mediul. Modificările pe care le facem asupra mediului prin aplicarea deciziilor, trebuie sa modifice în mod dinamic

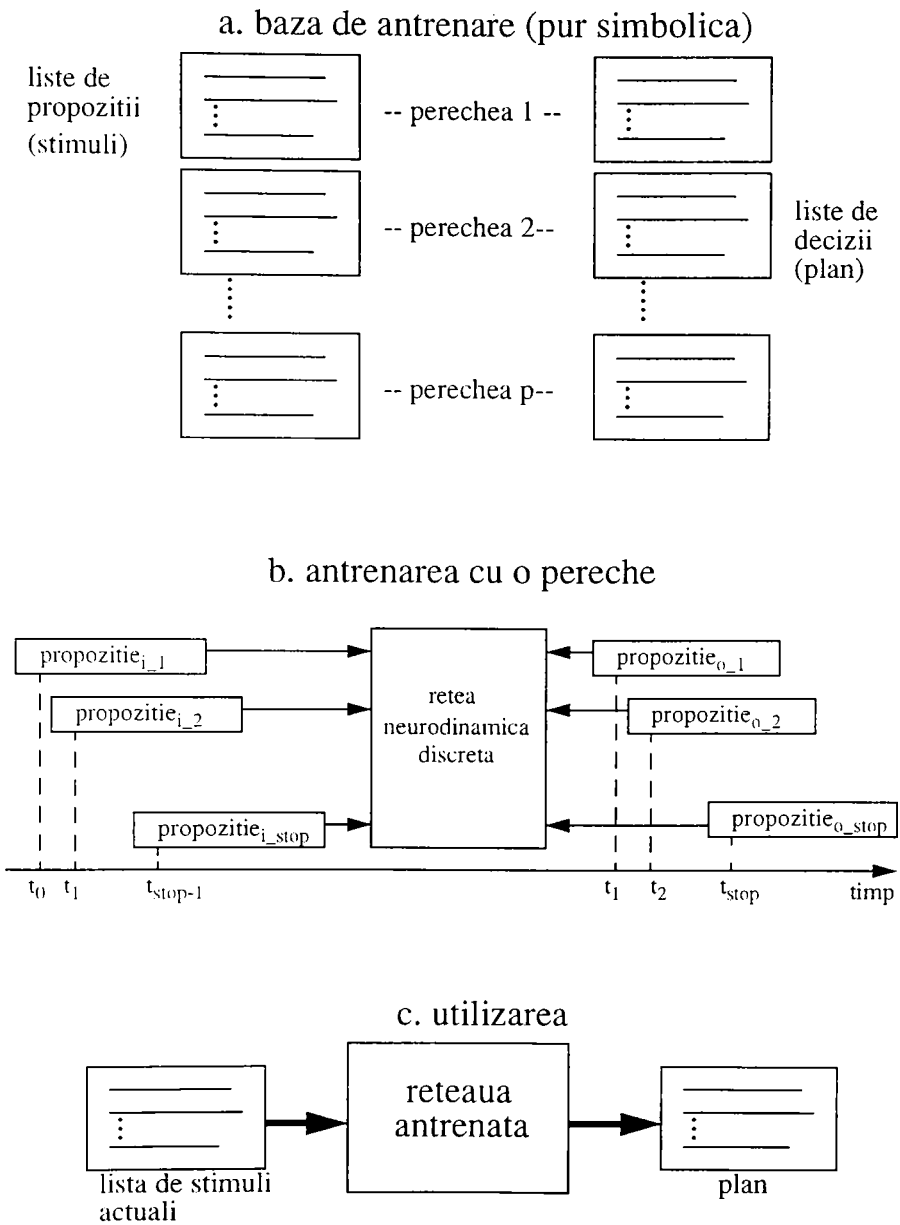


figura 4-1.

natura şi conţinutul stimulilor. Pentru aplicarea acestei strategii este însă necesar ca să construim sistemul cognitiv în cadrul mediului în care îşi va desfăşura activitatea (vezi figura 4-2). Experimental, acest lucru este foarte costisitor şi uneori imposibil, pentru că trebuie să creăm un mediu de laborator şi să "antrenăm" sistemul în funcţie de acesta. Acest mod de lucru este eficient într-o fază de testare, dar nu în una de construire (considerând construirea ca fiind de natură conexionistă, prin învăţare).

O soluţie mai bună este construirea unei alte reţele conexioniste, pe baza cunoştinţelor acumulate despre mediu, care să-l "simuleze". Practic, o astfel de reţea, ar avea la intrare, deciziile luate de prima reţea şi ar genera chiar stimulii pe care mediul l-ar oferi sistemului, ca in figura 4-3. În acest caz, dacă oferim sistemului doar un prim stimul, de "start" şi îl deconectăm de la mediul exterior, acesta va genera o primă decizie, care la rândul ei va genera prin reţeaua de modelare, un nou stimul, ş.a.m.d., până când nu se mai poate lua nici o decizie, sau nu se mai modifică mediul virtual. Evident, se poate întâmpla şi fenomenul unei ciclări infinite, care nu este de dorit. Acest model are o plauzibilitate psihologică, putând fi explicaţia pentru modul în care oamenii îşi creaza scenarii (de exemplu, cum ne imaginăm că se va desfaşura o discuţie cu o persoană cunoscută şi ale cărei reacţii le putem anticipa) [Miikulainen94].

Există două considerente legate de designul acestui sistem: sincronizarea şi comunicarea. Sincronizarea celor două reţele este legată de caracteristica temporală a reţelelor folosite. Dacă ambele reţele sunt continue, nu se pune explicit problema sincronizării, doar cel mult la nivel de realizare pseudo-continuă a traiectoriilor. Cazul în care o reţea este continuă şi una discretă nu merită a fi luat în considerare. Dacă ambele reţele sunt discrete, sincronizarea se poate face la nivel de paşi de evoluţie discretă (vezi capitolul 2), iar cea mai naturală cale este echivalarea "unu-la-unu" a paşilor (existând şi posibilitatea de "unu-la-doi", etc). O implementare cu două reţele continue este foarte costisitoare, pentru că impune fie o simulare digitală a unor sisteme dinamice neliniare, fie o implementare dedicată în hardware. Soluţia pe care am folosit-o în continuare a fost cea de sincronizare discretă "unu-la-unu".

Comunicarea se poate face pe două căi care par foarte asemănătoare ca efect, dar diferenţiază net sistemul rezultat pe două clase diferite la nivelul taxonomiei neurosimbolice (vezi Introducerea). Cea mai naturală pare a fi de a furniza ieşirea uneia dintre reţele, *în formă vectorială*, la intrarea celeilalte reţele. Pentru a obţine rezultatele în forma accesibilă de grafuri conceptuale, se vor traduce doar acei vectori de ieşire care sunt consideraţi necesari. Se poate observa că în acest caz sistemul îşi păstrează caracterul de uniformitate, procesarea făcându-se doar la nivel neuronal, chiar dacă este separată pe două reţele. Cealaltă alternativă este de a traduce fiecare vector generat de cele două reţele în graf conceptual, iar înainte de a-l furniza ca intrare, de a-l traduce din nou în formă conexionistă (prin translatarea tensorială prezentată în subcapitolul 3.3). Deşi o astfel de schemă pare redundantă şi costisitoare din punct de vedere al translatării, se va vedea în continuare că este singura posibilă.

Translatarea vectorilor de ieşire în grafuri conceptuale duce la discretizarea

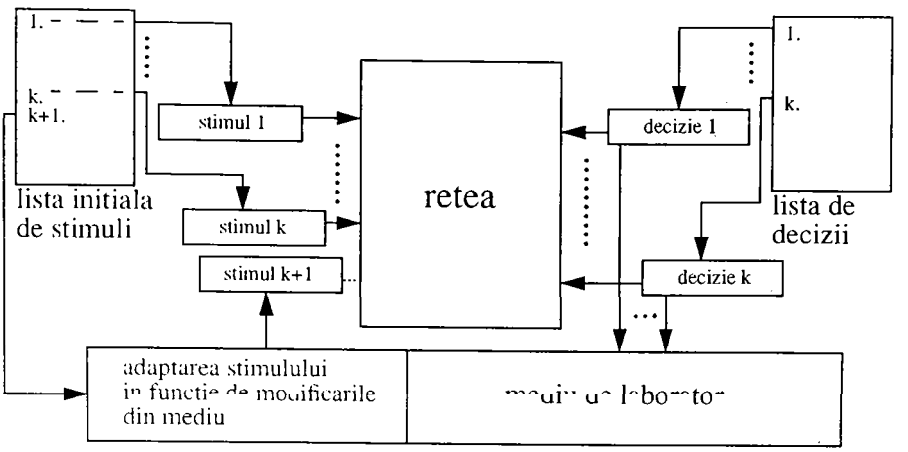


figura 4-2.

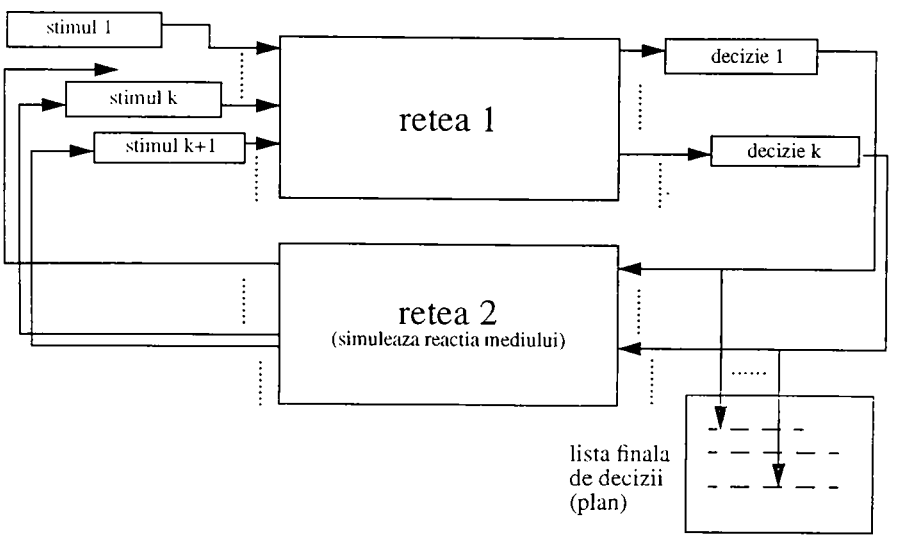


figura 4-3.

informaţiei, pentru că vectorii rezultaţi la ieşirea reţelelor au valori continue în intervalul [-1.0,1.0]. Mai mult, în cazul în care valorile sunt afectate de mici erori inerente legate de funcţionarea aproximativă a reţelelor, translatarea *vector* → *graf-conceptual* → *vector* duce la transformarea valorilor uşor eronate în valori corecte, printr-un proces de restabilire a informaţiei simbolice reprezentate de vectorii conexionişti de intrare. După cum se va vedea în continuare, această restabilire este necesară. Am denumit această schemă de comunicare, *translaţie-retranslaţie*.

4.2. Principul divergenţei

Datorită faptului că procesarea neuronală introduce inerent un factor de aproximare (sau mai bine spus, o eroare), desfăşurarea unui raţionament în buclă închisă va duce la o deteriorare progresivă a calităţii propoziţiilor generate. În loc să avem după un număr de iteraţii de raţionament o planificare bună, vom obţine una total necorespunzătoare. Acest lucru se petrece din două motive:

i. *Modelul lui Rumelhart este incomplet.* Plauzibilatatea lui este evidentă doar dacă există un mecanism de control, care supervizează cele două reţele conectate. În termeni psihologici, acest supervizor are ca şi corespondent gândirea serială, care analizează fiecare pas în dezvoltarea listei de decizii. În termeni computaţionali este vorba de un proces, care trebuie să verifice "calitatea" informaţiei vehiculată între cele două reţele.

ii. *Erorile din sistem se acumulează progresiv*, pe măsură ce se trece de la o iteraţie la următoarea. Acest fenomen poate duce la imposibilitatea translatării vectorilor în grafuri conceptuale, aceştia fiind prea afectaţi de "zgomotul" introdus de erorile acumulate.

Următoarea demonstraţie este foarte simplă şi imediată:

Propoziţia 1

Reţeaua 1 şi reţeaua 2 au fost antrenate cu bazele de învăţare $B_1=\{S_1,S_2,...,S_p\}$, *respectiv* $B_2=\{T_1,T_2,...,T_r\}$, *unde* $S_i=(S_i^{in},S_i^{out})$ *şi* $T_j=(T_j^{in},T_{jout})$, *cu condiţia* $\{T_j^{out}\}_{j=1,r}\subseteq\{S_i^{in}\}_{i=1,p}$.

Propoziţia 2

Dacă reţeaua 1 primeşte la intrare $X_1\approx S_i^{in}$ $(i\in\{w/w=1,p\})$ *ea va genera la ieşire* $Y_1\approx S_i^{out}$. *Se va introduce o eroare suplimentară nenulă, conform:*

$$|X_1-S_i^{in}|-|Y_1-S_i^{out}|=\epsilon_1 \qquad (13)$$

Propoziţia 3

Aplicând Y_1 *la intrarea reţelei 2, şi considerând că* $Y_1\approx T_j^{in}$ $(j\in\{w/w=1,r\})$ *aceasta va*

genera $X_2 \approx T_j^{out}$. *Se va introduce o eroare suplimentară nenulă:*

$$|Y_1 - T_j^{in}| - |X_2 - T_j^{out}| = \epsilon_2 \quad (14)$$

Propoziţia 4

Aplicând iterativ propoziţiile 2 şi 3 vom obţine un şir $\varepsilon_1, \varepsilon_2, ..., \varepsilon_n, ...$

Orice sistem conexionist respectă regula propagării erorii. Dacă erorile medii de la intrare sunt în relaţia $\varepsilon_1^{in} < \varepsilon_2^{in}$ atunci şi erorile de la ieşire sunt în aceiaşi relaţie $\varepsilon_1^{out} < \varepsilon_2^{out}$. Se poate observa din propoziţiile 2 şi 3 că:

$$|X_1 - S_i^{in}| < |X_2 - S_k^{in}| \quad (15)$$

unde $S_k^{in} = T_j^{out}$ (conform propoziţiei 1 trebuie să găsim o astfel de de identitate). Putem afirma:

Consecinţă (regula divergenţei)

Sirul ε_t *(t=1,f) nu este convergent.*

Eroarea crescând în mod progresiv, la un moment dat **t** vom ajunge în situaţia că nici una din valorile **X,Y** vehiculate, nu mai seamănă cu valorile din bazele de învăţare B_1, B_2, rezultatele generate fiind nefolositoare.

Această regulă a divergenţei (pe care am ridicat-o la rang de principiu), l-am observat experimental, în *toate* cazurile în care am lucrat cu reţele aflate în buclă închisă. Din acest motiv, folosirea comunicării bazate pe translaţie-retranslaţie este obligatorie. Acest fapt va schimba natura sistemului, din sistem uniform în unul hibrid, pentru că procesul de translaţie-retranslaţie este unul simbolic, în special prin faptul că apelează la structurile simbolice ierarhice ataşate grafurilor conceptuale utilizate. Pentru că sistemele hibride sunt puternic criticate de către susţinătorii modelului uniform, am considerat necesar să clarific într-un subcapitol separat această opţiunea a mea de a utiliza modelul hibrid.

4.3. Renunţarea la uniformitate şi folosirea unei structuri hibride

Punctul de plecare în argumentare poate fi *principiul uniformităţii*. Acesta este definit în felul următor:

Principiul uniformităţii

O problemă computaţională poate fi rezolvată şi implementată eficient dacă se foloseşte doar aparatul formal al unei singure arhitecturi computaţionale.

Principiul a fost introdus de Newell (1980), reluat şi rafinat de Langley (1987) (vezi [Newell90]), ambele abordări analizând cazul modelelor (arhitecturilor) cognitive clasice, iar de pe poziţiile conexionismului a fost argumentat de Smolensky [Smolensky1988] şi Jordan (1986). O mulţime de alţi autori ([Clark89], [Shavlik91], chiar [Smolensky92], în plus [VanGelder95], [Sharkey95]), au combătut în perioada 89-95 acest principiu.

Aplicarea principiului uniformităţii presupune definirea unui set de operaţii de bază, foarte simple, care să definească o arhitectură computaţională. Extremele care apar prin această aplicare duc la două direcţii (mai exact mentalităţi) de cercetare în modelarea proceselor cognitive (procesarea simbolică cu atribute de inteligenţă). Partizanii principiului prin aplicarea metodelor sintactice şi logice, susţin următoarea aserţiune:

Orice rezultat al unei activităţi cognitive poate fi explicat printr-un model matematic derivat din logică şi teoria limbajelor formale, sau probabilităţi.

Cei din tabăra conexionistă (aderenţi ai principiului uniformităţii) susţin că:

Orice rezultat al unei activităţi cognitive poate fi obţinut folosind modelul computaţional conexionist.

Argumentele pe care le vom dezvolta în continuare, sugerează că ne putem opune principiului uniformităţii, combătând avantajele sale pe toate planurile (claritate, implementabilitate, robusteţe, consistenţă matematică, plasticitate, transparenţă). Alternativa este de a propune o imagine a unui sistem cognitiv, ca o "combinare" a unei multitudini de arhitecturi computaţionale, fiecare fiind adaptată pentru anumite probleme specifice, fiecare fiind univoc definită de un set de operaţii de bază. Aceste operaţii sunt distincte de la o arhitectură la alta.

Pentru a ilustra dificultăţile care apar relativ la capacitatea de a realiza artificial procese cognitive cu cele două soluţii extreme prezentate anterior, vom folosi ca instrument ajutător, noţiunea de **funcţie**. Aparent, rezultatul cognitiv care se obţine prin aplicarea unei anumite funcţii asupra unor date iniţiale, se referă doar la CE funcţie este invocată şi nu ne interesează CUM este calculat rezultatul funcţiei. Dacă reducem modelul conexionist şi pe cel sintactico-logic la maşina Turing (şi este posibil [Giles&al95]) am putea demonstra că putem calcula la fel de bine aceleaşi funcţii cu ambele arhitecturi, evident cu rezultate diferite în ceea ce priveşte viteza de calcul, şi în acest caz soluţia de implementare este dată doar de indicii de performanţă.

Se poate considera că problema (mai exact greşeala) este dată de aserţiunea (probabil insuficient analizată) dată de propoziţia "calculând aceleaşi funcţii". Oare o maşină care calculează 7x8 prin asocierea dintr-o memorie asociativă a rezultatului 7x7=49 cu adunarea finală a lui 8, face "aceiaşi funcţie" cu o maşină care adună pe 7 de 8 ori într-un acumulator? În sensul că prezentând două numere şi o operaţie obţinem un rezultat; acelaşi în ambele cazuri, răspunsul ar fi cu certitudine **da**, e vorba de

aceiaşi funcţie. Modul în care se ajunge la rezultat va fi diferit. Din punct de vedere al vitezei şi al robusteţii, prima variantă (deşi neconvenţională) este mai bună. A doua variantă este mai simplu de implementat dar consumă timp, iar în cazul defectării sistemului chiar în timpul evaluării funcţiei, rezultatul primei versiuni va fi mai apropiat de rezultatul final în majoritatea cazurilor decât rezultatul celei de-a doua (paradoxal, modelul conexionist este mai bun într-un astfel de caz particular, de implementare a tablei înmulţirii).

Exemplul dat poate fi puţin derutant, pentru că se discută aici despre procese cognitive, nu de operaţii aritmetice, deci funcţia de mai sus trebuie privită prin prisma cercetătorului din domeniul Inteligenţei Artificiale şi nu a elevului de şcoala primară. Mai ales pentru că performanţa umană în domeniul procesării cognitive se măsoară în primul rând prin viteză, minimizarea erorii şi robusteţea raţionamentului. Deci, in **acest context**, este important să analizăm nu atât CE anume face funcţia, ci CUM este calea prin care face. Ajungând din nou la cele două soluţii extreme de implementare, memorarea asociativă a datelor este o caracteristică de bază a modelului conexionist. Funcţii din clasa celei prezentate pot fi realizate prin reţele conexioniste. Este însă exagerat a afirma că oferind o astfel de implementare (apropiată de modul de calcul uman, care are memorată asociativ tabla înmulţirii) putem explica microstructura tuturor proceselor cognitive. Există chiar o analogie favorită a partizanilor modelelor conexioniste uniforme, cea dintre modelul sintactico-logic cu fizica Newtoniană şi dintre modelul conexionist cu fizica cuantică. Putem cita:

"Se poate argumenta că procesarea simbolică convenţională oferă un model macroscopic, analog mecanicii Newtoniene, pe când modele noastre [conexioniste] oferă o analiză microscopică, analoagă teoriei Cuantice. Observaţi că, în domeniul lor de acoperire, aceste două teorii pot face predicţii precise asupra evoluţiei şi comportamentului obiectelor din lumea reală... Totuşi, în anumite situaţii, teoria Newtoniană nu este corectă. În aceste situaţii, trebuie să ne bazăm pe teoria Cuantică."

(Rumelhart & McClelland, PDP, 1986, p.125, sublinierile îmi aparţin)

Pare a fi o analogie rezonabilă, dar e corectă numai în cazul în care teoria Newtoniană ar fi corectă pentru o anumită clasă de fenomene. Dar nu este aşa. Mai exact, putem afirma, că în anumite situaţii, teoria Newtoniană (deşi incorectă la nivel de detaliu), **este aplicabilă**, iar teoria Cuantică explică şi de ce este aşa. Doar în acest caz am respecta şi principiul uniformităţii aplicat la fizică. Dar se poate observa că în această situaţie nu mai există aceiaşi analogie între modelul convenţional şi cel conexionist. Revenind la ideea paradigmei subsimbolice [Smolensky88], ar însemna că descrierile seriale, simbolice ale macroproceselor cognitive ar fi aproximări ale unei descrieri de detaliu, care este oferită de nivelul de microprocesare conexionistă. Putem cita din nou:

"Noi vedem macroteoriile ca aproximaţii ale unei structuri cu detalii de distribuire microscopică..."

(Rumelhart & McClelland, PDP, 1986, p.126)

Ceea ce evident că nu corespunde realităţii. Dimpotrivă, macroprocesele nu sunt deloc aproximaţii, ele sunt fenomene deterministe, modelate printr-un aparat formal, care este considerat chiar prea rigid. În contrast, modelul conexionist este tocmai cel care introduce aproximarea.

Această analogie greşită poate duce cercetătorul, într-o primă fază de entuziasm "conexionist" către o subestimare a modelelor convenţionale. Aplicând în mod necondiţionat principiul uniformităţii în realizarea proceselor cognitive conexioniste putem face o mare greşeală, folosind instrumente deloc adecvate pentru diverse clase de procese cognitive, care pot fi relativ uşor rezolvate cu arhitecturi convenţionale. Modelul conexionist NU ESTE uniform, relativ la domeniul cognitiv. Mai mult, sunt situaţii în care modelele convenţionale sunt complete, într-un mod în care fizica Newtoniană nu este niciodată completă.

Exemplele sunt uşor de dat. Dacă un subiect uman simulează efectiv un proces simbolic serial, în acelaşi mod în care acesta ar fi realizat de o maşină Van Neumann, procesul va fi **corect**. Punct. Nu "aproximativ corect". Evident că suportul fizic al unei astfel de simulări este tot un sistem conexionist. Dar nu este relevant din punct de vedere procesului simbolic ce se petrece la acest nivel, pentru că nivelul de definire (prin formalism) al procesului nu ia în considerare absolut deloc un model conexionist. Deci, în mod clar, în această situaţie principiul uniformităţii este dăunător. Practic, pentru acest tip de procese (să le numim seriale), putem introduce conceptul de maşină virtuală, care poate rula pe orice fel de maşină (Van Neumann, conexionistă, sau chiar mintea umană). Rezultatul este cel care contează şi el este mereu acelaşi (corect şi fără aproximări).

În acest caz, întrebarea care se pune, este cum poate mintea umană să aibă capacitatea de a rezolva probleme tipice pentru procesarea simbolică, serială, sintactico-logică, dacă ea este fundamental o arhitectură conexionistă, deci aproximativă. Speculaţia pe care o putem face, este că maşina conexionistă pe care o implementează creierul poate ocazional să simuleze o maşină Von Neumann. Dacă sistemul tipic uman de procesare al informaţiei îşi desfăşoară activitatea prin asociere şi "relaxarea" într-o soluţie, fără a folosi operaţiile logice, de ce putem efectua raţionamente bazate pe matematică, sau deducţii ştiinţifice solide? Cum putem raţiona noi pur logic, dacă operaţiile de bază nu au nimic în comun cu logica? Răspunsul este foarte interesant: *Capacitatea noastră de a executa procese formalizabile, guvernate de reguli precise este dată de abilitatea de a crea instrumente artificiale, exterioare sistemului cognitiv, care sunt reprezentări fizice ale problemelor*, pe care le putem manipula prin scheme foarte simple pentru a obţine în final rezultatele unor probleme abstracte foarte dificile.

Sunt două clase distincte de task-uri computaţionale de acest fel (clase pentru care modelele conexioniste nu se pretează):

i. procese de raţionament serial, în care ordinea operaţiilor este vitală.
ii. procese de raţionament "generativ", în care structuri simbolice întregi sunt produse prin aplicare unor reguli (de inferenţă simbolică - unificare şi pattern-

matching) asupra unei baze de cunoştinţe reprezentabile la nivel simbolic.

Ca exemple putem da la punctul 1: planificarea conştientă a acţiunilor, raţionamentul logic, demonstraţia matematică şi gândirea abstractă. La punctul ii., cel mai important task, pentru mintea umană, este generarea discursului.

În mod cert, genul de procese cognitive prezentat mai sus este "conştient" pentru persoana care le execută. Asocierile se execută însă mult mai des, iar în majoritatea cazurilor subiectul nu este conştient că un astfel de proces cognitiv (de asociere) este în curs de desfăşurare. Aceste procese de asociere pot fi mai multe active simultan, conştientizarea putându-se face doar în momentele de "stabilitate" conexionistă, când spunem că asocierea dă rezultate. Procesele seriale se desfăşoară însă într-un ritm mult mai lent, fiind necesară întreaga lor conştientizare şi fixarea în context pentru fiecare pas al procesului. Mai mult, pentru a nu pierde "cursul" unui astfel de proces, avem nevoie de memorie externă şi reprezentări semiotice adecvate, cu semantică asociată, astfel că în majoritatea cazurilor suntem nevoiţi să folosim hârtia şi creionul. Atunci când facem acest lucru, folosim instrumentele formale externe menţionate mai sus, pe care le-am învăţat să le manipulăm. Ele sunt inventate de noi, cel mai adesea făcând parte din fondul general de instrumente formale ale ştiinţei. Dacă de exemplu, avem o serie de numere mari şi trebuie să adăugăm 2 la fiecare şi să obţinem suma totală, în mod cert, o persoană obişnuită va avea nevoie de hârtie şi creion. Şi mai avem nevoie de acele reprezentări exterioare pentru numere şi operaţii aritmetice. Mediul exterior pe care ni l-am creat devine o extensie esenţială a minţii noastre. Chiar dacă vom învăţa să executăm aceste operaţii "în gând", renunţând la instrumentele exterioare, acest lucru este posibil pentru că ne construim un "model mental" al structurilor exterioare folosite, iar manipularea acestora nu se face exterior, ci interior, dar serial şi conştient. Dar nu trebuie să pierdem din vedere că funcţii de tipul 7x7=49 se realizează prin simplă asociere.

Făcând o sumă a acestor abilităţi cognitive, putem specula că sunt trei capacităţi combinate, care permit minţii umane să execute funcţii complexe, cu un grad mare de abstractizare:

a. capacitatea asociativă fundamentală (simplă în esenţă)
b. capacitatea de a modela mental mediul înconjurător
c. capacitatea de a manipula fizic instrumentele exterioare şi de a percepe efectele acestor manipulări.

Practic, putem reduce aceste capacităţi, la una de bază, care este pur neuronală ca operaţii de bază (relaxarea termodinamică perturbată). Dar acest lucru nu este relevant pentru că nivelele b. şi c. sunt specifice pentru maşina Von Neumann, care separă datele procesate de funcţiile care le procesează (lucru care nu se întâmplă în modelele conexioniste). Slăbiciunea arhitecturii Von Neumann aplicată în modelele cognitive este dată de faptul că suntem constrânşi să tratăm toate procesele cognitive ca depinzând de manipularea instrumentelor simbolice externe. Asta face ca modelele cognitive derivate prin această arhitectură să fie limitative şi rigide.

Noul start în acest domeniu, oferit de modelul conexionist, ne oferă o serie de mecanisme care pot completa neajunsurile arhitecturilor convenţionale. Greşeala este de a crede că aceste noi modele le pot înlocui complet. Problema este cum vom reuşi să completăm lacunele sistemelor convenţionale cu ajutorul modelului conexionist. O alternativă atractivă este de a construi sisteme conexioniste, care prin învăţare sau construcţie, să fie capabile de a executa procese seriale, abstracte. Din păcate, acest lucru nu este de loc simplu. Instrumentele de reprezentare a realităţii fizice s-au format în cursul istoriei, fiind în majoritatea cazurilor, rafinate prin mai multe generaţii, şi aduse în forma finală şi păstrate, prin transmitere culturală. În plus, numărul şi complexitatea lor este foarte mare. Implementarea acestor instrumente prin reţele conexioniste ar fi plauzibilă din punct de vedere al paralelismului model uman - maşină, dar nu ar fi eficientă tehnologic.

Concluzia pe care o putem extrage din această discuţie, este că avem nevoie de sisteme cognitive care au ambele modele încorporate. Un astfel de sistem ar avea două nivele: unul rapid, bazat pe modelul conexionist, care să genereze rezultate prin asociere, şi unul convenţional, care să obţină rezultatele prin procese seriale, pur simbolice. Implementarea celor două nivele trebuie să se facă renunţând la principiul uniformităţii, utilizând modelele convenţionale când este cazul şi modelul conexionist pentru calculul de tip "intuitiv". Legătura dintre cele două arhitecturi este o problemă de comunicaţie şi translaţie. Am arătat în capitolul dedicat paradigmei subsimbolice (3.2) că există posibilitatea de a folosi structuri simbolice în contextul modelelor conexioniste.

4.4. Antrenarea unei reţele neurodinamice cu grafuri conceptuale

4.4.1. Dimensiunea reţelei folosite

Se obişnuieşte să se aleagă în cazul utilizării sistemelor neurodinamice un număr de unităţi de calcul (neuroni) pentru care dimesionalitatea spaţiului de intrare/ieşire (însumate) să reprezinte 10-20%. De exemplu, pentru o aplicaţie de robotică cu trei grade de libertate la intrare şi 6 la ieşire este absolut necesar să avem 50-100 de neuroni în total (numărul depinzând şi de dimensiunea bazei de învăţare). Pentru cazul procesării simbolice propuse s-ar deduce că am avea nevoie de $3 \cdot 10^4$-$6 \cdot 10^4$ neuroni şi ar rezulta o matrice de ponderi cu 1-$2 \cdot 10^8$ elemente, adică sute de milioane de numere în virgulă flotantă, respectiv peste un GByte de memorie necesară. Din acest punct de vedere, la nivelul anului 1997, tehnologic şi practic problema nu reprezintă nimic deosebit, dar algoritmul de învăţare folosit [Williams90] are complexitatea $o(n^4)$, unde n este numărul de neuroni. Am avea deci nevoie de un număr de $4 \cdot 10^{16}$ iteraţii pentru o singură epocă (şi de obicei avem nevoie de mii de epoci) şi se observă imediat că aceste ordine de mărime pentru timpul de rulare necesar pentru o antrenare completă şi care sunt în jur de 10^{20}, sunt impracticabile.

O propunere interesantă, făcută de Smolensky este de a folosi pentru astfel de probleme (unde vectorii nu reprezintă mapări ale realităţii fizice -mărimi cu dimensiune

euclidiană - ci sunt combinaţii binare rezultate din produse si superpoziţii tensoriale) aceiaşi neuroni şi pentru intrare şi pentru ieşire, renunţând complet la neuronii ascunşi. Explicaţia este simplă: majoritatea intrărilor în reţea la un moment **t** sunt 0, iar în pasul următor, la ieşire se vor obţine puţine intrări setate pe 1. Astfel, perturbaţia totală care se aplică asupra reţelei şi modificarea în starea ei (ieşirea) de la un moment de timp la altul, sunt ambele foarte mici. Nu trebuie uitat faptul că starea unităţilor este o stare cuprinsă în intervalul [0,1], dar la ieşire, vom considera că orice stare care are valoarea mai mare decât 0.5 este echivalentă cu 1. La intrare, perturbaţiile de tip 1 vor fi ponderate de valorile calculate la învăţare pentru respectivele intrări.

Se poate compara această reţea cu una care are doar 10% din neuroni afectaţi pentru intrări şi ieşiri, dar în care modificările aparute în starea acestora între două momente succesive de timp sunt mult mai mari. Altă explicaţie este urmatoarea: capacitatea de învăţare a unei reţele este dată în primul rând de dimensiunea matricii de ponderi. Este clar că pentru un număr relativ mic de propoziţii care generează vectori binari (cu puţine elemente setate pe 1) matricea de ponderi este arhisuficientă dacă are **3125·3125=9,765,625** elemente (aproximativ 80 MBytes pentru o reprezentare în virgulă flotantă). Însă din motive de timp de rulare, o reţea cu mii de unităţi, este prea mare. A trebuit deci să reduc dimensiunea reţelei la ordinul sutelor pentru a putea rula antrenări fezabile şi care se conţină un număr suficient de epoci.

4.4.2. Timpul necesar antrenării şi paralelizarea algoritmului

Am observat că rulând algoritmul de învăţare pentru un număr de 3125 unităţi şi o matrice corespunzătoare de 80 Mbytes pe o staţie SUN Sparc într-o reţea cu serverul de fişiere la distanţă, timpul de rulare pentru o singură epocă ar atinge aproximativ 900 de zile!. Timpul scade la aproximativ 300 de zile dacă matricea este memorată pe discul local, iar staţia este folosită exclusiv pentru acest scop. Dacă am avea un sistem cu peste 80 MBytes memorie RAM (nu am avut la dispoziţie astfel de resurse în prima fază), timpul s-ar reduce cu un ordin de mărime (sau chiar mai mult). Totuşi, chiar şi în această situaţie, pentru o singură epocă, timpul de rulare rămâne de ordinul zilelor. Singura soluţie pentru obţinerea unor timpi de rulare acceptabili este procesarea paralelă. Implementarea algoritmului de învăţare în PVM [Szirbik&Somlo 95] şi rularea pe 4 staţii în paralel, a relevat posibilitatea unei accelerări foarte apropiate de factorul maxim de "speed-up" (4 în acest caz). Spre deosebire de restul algoritmilor de învăţare (un exemplu clasic este backpropagation-ul aplicat la reţele feedforward), algoritmul lui Williams şi Zipser (a cărui variantă modificată prezentată în capitolul 2, am folosit-o şi eu) este unul care se pretează perfect la procesarea paralel-distribuită.

Mai multe rezultate experimentale relativ la paralelizare (care nu este subiectul acestei teze) pot fi găsite în [Szirbik95b]. Aceste rezultate au fost obţinute prin rularea programului pe un sistem de calcul cu 80 de procesoare Sparc, fiecare cu 32 de MBytes memorie RAM locală şi o memorie "comună" RAM de 128 MBytes (accesată ceva mai lent decât memoria locală). Procesoare comunicau prin mesaje, printr-o reţea

Ethernet specială, cu o rată de 100Mbs. Acest sistem de calcul, numit ZOO (al Facultăţii de Matematică şi Informatică de la Universitatea Liberă din Amsterdam), dispunea de un sistem de operare artizanal, AMOEBA [Tannenbaum&Rennese89] care permitea multitasking-ul, dar şi rezervarea întregului sistem pentru un singur program, în perioadele în care sistemul nu era folosit (8 p.m.-8 a.m. sau vineri 8 p.m. - luni 8 a.m.). Cele mai bune rezultate erau obţinute în acest mod de rulare "rezervat". Partea de programare era foarte mult simplificată prin existenţa unui compilator (dedicat) pentru limbajul de programare cu facilităţi paralel-distribuite Orca [Bal89]. Limbajul fiind foarte elegant şi eficient, testarea de aplicaţii ca aceasta era relativ facilă, rămânând în sarcina utilizatorului numai problemele aplicaţiei cu care acesta se confruntă.

Utilizând aceste resurse de calcul, pentru reţeaua de 3125 de unităţi şi un set restrâns de tipare (3-5 tipare) formate din secvenţe de 10 (t=0,...,9) grafuri conceptuale de dimensiuni reduse care erau traduse în vectori, am obţinut că durata unei epoci era de 3 ore si 20 de minute. Având în vedere că ulterior am remarcat că sunt necesare un număr de aproximativ 500-1500 epoci pentru astfel de task-uri de învăţare, acest timp de rulare nu era corespunzător. Singura soluţie pe care am găsit-o a fost reducerea dimensiunii reţelei, astfel încât sa putem încadra aproximativ 500 de epoci într-un interval de 12 ore, deci aproximativ 1 minut/epocă. Aceste concluzii au fost obţinute prin experimente succesive, ajungând la următoarele rezultate:

nr. unităţi	timp pentru 1 tipar (sec)	timp pentru 2 tipare (sec)	timp pentru 3 tipare (sec)	timp pentru 4 tipare (sec)
3125	3210	6100	9050	12020
2000	1120	2080	3050	4030
1500	550	1004	1537	1940
1000	101	191	288	375
800	30	57	85	112
700	15	28	43	54
600	9	17	25	31

în condiţiile în care nimeni nu mai foloseşte sistemul. Deci, experimentele se puteau desfăşura doar într-un anumit interval de mărime al reţelei, de [700,800] unităţi de calcul. Utilizarea unui număr atât de mic de unităţi, face dificilă aplicarea reprezentării tensoriale. Fiind vorba de un tensor de gradul 5, dimensionalităţile spaţiilor vectoriale V_F, V_R şi V_L trebuie să fie de maxim 3 sau 4 (pentru că 4^5=**1024** şi 3^5=**243**). Acest lucru înseamnă că putem avea 8-16 individualităţi, 8-16 clase şi 8-16 relaţii posibile. Am ales pentru experimentare o variantă cu 16 individualităţi, 14 clase (reprezentarea "left-right" generează 2^4-2 combinaţii de codificare posibile) şi 8 relaţii codificate binar complet (000, 001,..., 111). Rezultă exact 768 de unităţi şi ne încadrăm foarte bine în

intervalul dorit:

nr. unităţi	timp pentru 1 tipar (sec)	timp pentru 2 tipare (sec)	timp pentru 3 tipare (sec)	timp pentru 4 tipare (sec)
768	22	41	60	79

Dar având un număr atât de mic de cuvinte în universul discursului, este imposibil de construit un set de tipare care să depăşească un nivel semantic acceptabil. Din acest motiv am folosit doar simboluri literale:

- individualităţi: **F = { A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P }**
- clase **R = { α,β,γ,δ,ε,ζ,η,θ,σ,χ,λ,μ,ν,ξ }**
- relaţii **L = { r, s, t, u, w, x, y, z }**

cu care am construit grafurile conceptuale şi ierarhia de clase. Într-o fază în care încercam să testez doar modalităţile de antrenare, nu era necesar ca aceste simboluri să aibă o semantică asociată.

Un tipar era format din 10 grafuri conceptuale la intrare şi 10 la ieşire. Au fost făcute experimente de antrenare a unei reţele cu 786 unităţi pentru 1, 2, 3 şi 4 tipare. Scopul urmărit a fost nu de a învăţa tiparele ci de a cerceta puterea de generalizare a reţelei. Aceasta capacitate poate fi determinată prin următorul experiment:

- se învaţă reţeaua câteva tipare, care au cam acelaşi "pattern" în timp, dar la "nivele" diferite
- după ce eroarea a scăzut sub 3% (în astfel de aplicaţii 5-1% este intervalul cel mai folosit [Hilario96]), se dă reţelei o componentă care respectă "pattern"-ul temporal al intrărilor învăţate (indiferent de "nivel")
- dacă vom obţine la ieşire o componentă care respectă "pattern"-ul temporal şi semantic al ieşirilor învăţate, atunci scopul a fost atins.

Nu era necesar în acest moment testarea unui model cu două reţele în buclă închisă, un sistem cu o singură reţea fiind suficient petru testarea puterii de generalizare a unei reţele recurente neurosimbolice.

4.4.3. Testarea generalizării

Primele tentative de construire a unei baze de învăţare pentru un model cu o singură reţea a plecat de la ideea unui posibil *dialog* om-sistem cognitiv artificial. Utilizatorul uman furnizează propoziţii, într-o secvenţă care implică la fiecare pas, receptarea şi interpretarea răspunsului dat de sistemul automat. Utilizatorul introduce

o propoziţie i(t) şi primeşte imediat răspunsul o(t+1). Pe baza acestui răspuns şi a propoziţiilor vehiculate între om şi sistem în momentele anterioare, el va continua dialogul cu o nouă propoziţie i(t+1), care se află într-o relaţie de asociere semantică cu cele furnizate de utilizator sau generate de sistem. Din punct de vedere al plauzibilităţii unui astfel de sistem, putem consemna apropierea în spaţiul euclidian a punctelor care reprezintă propoziţii cu sens apropiat şi corespunzător, crearea unei traiectorii în spaţiul stărilor reţelei; un alt aspect este că o reţea recurentă are memorie, în sensul că păstrează informaţie despre punctele prin care a trecut.

Asocierea semantică între două sau mai multe propoziţii se referă la vocabularul folosit, la clasele care apar în grafurile conceptuale rezultate şi un anumit pattern temporal al "acţiunilor" (relaţiilor între concepte) care rezultă din dialog, în sensul că anumite propoziţii apar în mod obligatoriu după alte propoziţii. Folosirea unui vocabular cu un număr foarte mic de cuvinte (vezi subcapitolul anterior) a făcut imposibilă construirea unor dialoguri în limbaj natural care să aibă sens. *Dar putem considera ca un limbaj inteligent şi inteligibil de comunicare poate fi format şi dintr-un număr foarte mic de simboluri.* O serie de regularităţi pot fi observate în dialogurile în limbaj natural, iar pe baza acestora se pot construi dialoguri care aparent nu au sens, dar pot testa un astfel de sistem minimal. Am construit o bază de învăţare formată din patru dialoguri.

• primul tipar:

```
i1(0): (α(A),r,β(B))  →  o1(1): (γ(C),s,δ(D))
i1(1): (β(B),t,ε(E))  →  o1(2): (δ(D),u,ζ(F))
i1(2): (ε(E),w,η(G))  →  o1(3): (ζ(F),x,θ(H))
i1(3): (η(G),y,σ(I))  →  o1(4): (θ(H),z,χ(J))
i1(4): (σ(I),r,λ(K))  →  o1(5): (χ(J),s,μ(M))
i1(5): (λ(K),t,ν(N))  →  o1(6): (μ(M),u,ξ(O))
i1(6): (ν(N),w,α(P))  →  o1(7): (ξ(O),x,β(A))
i1(7): (α(P),y,γ(B))  →  o1(8): (β(A),z,δ(C))
i1(8): (γ(B),r,ε(D))  →  o1(9): (δ(C),s,ζ(E))
i1(9): (ε(D),t,η(F))  →  o1(10): (ζ(E),u,η(F))
```

• al doile tipar:

```
i2(0): (α(P),y,γ(B))  →  o2(1): (γ(C),z,ε(D))
i2(1): (γ(B),r,ε(D))  →  o2(2): (γ(D),s,δ(B))
i2(2): (ε(D),t,η(F))  →  o2(3): (δ(B),u,ζ(E))
i2(3): (η(F),w,σ(I))  →  o2(4): (ζ(E),x,θ(I))
i2(4): (σ(I),y,ξ(A))  →  o2(5): (β(A),z,δ(B))
i2(5): (α(A),r,β(B))  →  o2(6): (δ(B),s,ζ(F))
i2(6): (β(B),t,ε(E))  →  o2(7): (ζ(F),u,η(H))
i2(7): (ε(E),w,η(H))  →  o2(8): (β(B),x,δ(D))
i2(8): (η(H),y,σ(J))  →  o2(9): (θ(I),z,χ(M))
i2(9): (σ(J),r,λ(K))  →  o2(10): (χ(M),s,μ(O))
```

• al treilea tipar:

```
i3(0): (ν(M),w,α(O))  →  o3(1): (ξ(B),x,β(P))
i3(1): (α(O),y,γ(D))  →  o3(2): (β(P),z,δ(D))
i3(2): (γ(D),r,ε(F))  →  o3(3): (δ(D),s,ζ(F))
i3(3): (ε(F),t,η(G))  →  o3(4): (ζ(F),u,η(G))
i3(4): (δ(G),w,σ(K))  →  o3(5): (θ(H),x,γ(C))
i3(5): (σ(K),y,ξ(O))  →  o3(6): (γ(C),z,δ(G))
i3(6): (α(A),r,β(B))  →  o3(7): (γ(D),s,δ(G))
i3(7): (β(B),t,ε(D))  →  o3(8): (δ(G),u,ζ(E))
i3(8): (ε(D),w,η(F))  →  o3(9): (ζ(E),x,θ(I))
i3(9): (η(F),y,σ(I))  →  o3(10): (θ(I),z,χ(J))
```

• al patrulea tipar:

```
i4(0): (λ(L),t,ν(M))  →  o4(1): (μ(M),u,ξ(O))
i4(1): (ν(M),w,α(O))  →  o4(2): (ξ(O),x,β(P))
i4(2): (α(O),y,γ(B))  →  o4(3): (β(P),z,δ(B))
i4(3): (α(P),r,β(A))  →  o4(4): (δ(B),s,ζ(F))
i4(4): (β(A),t,ε(E))  →  o4(5): (δ(C),u,ζ(G))
i4(5): (ε(E),w,η(F))  →  o4(6): (ζ(G),x,θ(I))
i4(6): (η(F),y,σ(I))  →  o4(7): (θ(I),z,χ(J))
i4(7): (σ(I),r,λ(L))  →  o4(8): (χ(J),s,μ(M))
i4(8): (ε(E),t,η(G))  →  o4(9): (ζ(F),u,μ(O))
i4(9): (δ(D),w,σ(K))  →  o4(10): (θ(H),x,γ(B))
```

Se pot observa o serie de regularităţi (care nu sunt respectate întotdeauna, tocmai pentru a sublinia aspectul de "common-sense reasoning"), o parte impuse de structura modelului şi o parte definite de structura unor dialoguri în limbaj natural care au fost studiate pentru găsirea unor "pattern"-uri. Structura modelului a impus folosirea unor propoziţii simple, cu structuri sintactice identice, formate din două concepte şi o relaţie, pentru a avea o schemă cât mai simplă de decodificare din vector în graf conceptual (vezi capitolul 3). Lungimea dialogurilor trebuie să fie aceiaşi, adică 10 paşi ceea ce este impus de natura algoritmului Williams&Zipser. Individualităţile trebuie să aparţină unor clase, care au fost definite după cum urmează:

$\alpha \supset$ {A,O,P}	$\beta \supset$ {A,B,P}	
$\delta \supset$ {C,D,E}	$\gamma \supset$ {B,C,D}	$\varepsilon \supset$ {E,F,D}
$\zeta \supset$ {E,F,G}	$\eta \supset$ {F,G,H}	$\theta \supset$ {H,I,K}
$\sigma \supset$ {I,J,K}	$\chi \supset$ {J,M,N}	$\lambda \supset$ {K,L,M}
$\mu \supset$ {M,O,P}	$\nu \supset$ {N,M,O}	$\xi \supset$ {A,B,O}

Regularităţile induse de mine (nu cele impuse de structura modelului) au fost:

• la intrare pot apare jumătate din relaţii, iar la ieşire cealaltă jumătate.
• relaţiile apar în aceiaşi secvenţă, şi la intrare şi la ieşire:
intrare : (r,t,w,y) ieşire : (s,u,x,z)

şi sunt sincronizate unele cu celelalte, chiar dacă prima relaţie diferă de la un dialog

la celălalt.
• conceptul al doilea dintr-o propoziţie i/o(t), apare ca prim concept in propoziţia următoare i/o(t+1).
• anumite concepte nu pot apărea decât împreună cu anumite clase.
(Observaţie: aceste reguli nu sunt respectate în toate circumstanţele).

Aceste regularităţi impun ca dialogurile "artificiale" construite să aibă un aspect structural foarte apropiat de dialogurile din limbaj natural.

Antrenarea reţelei s-a făcut prima oară cu un singur tipar. Cu toate că această antrenare a fost dificilă (vezi subcapitolul următor) ea a putut fi adusă în marja de eroare de 3%. Dar cu toate că furnizând la intrare secvenţe de propoziţii asemănătoare cu {i1(k):k=0,9}, obţineam rezultate foarte apropiate (în majoritatea cazurilor, identice) cu secvenţa {o1(k):k=1,10}, este prea puţin pentru a argumenta o generalizare. Am antrenat reţeaua cu 2, 3 şi 4 tipare, observând că învăţarea devine foarte dificilă cu cât numărul de tipare creşte şi cu cât tiparele sunt mai diferite de "pattern"-ul comun, după cum se vede şi din graficul 4-4 (a,b,c,d,). Dificultatea nu s-a datorat numai creşterii numărului de epoci necesare, ci mai ales unui fenomen numit bifurcaţie, care apărea cu atât mai des cu cât numărul de tipare era mai mare, iar diversitatea lor creştea. Trebuie accentuat mai ales faptul că efortul de antrenare al celor patru tipare mai sus prezentate până la o eroare de sub 5% a fost enorm, de aproximativ 125 de ore de rulare pe sistemul ZOO în mod "unic-utilizator" (luând în considerare şi reprizele de antrenare nereuşită).

Sistemul cognitiv rezultat a fost testat în felul următor: utilizatorul furniza o propoziţie, sistemul răspundea cu altă propoziţie, iar utilizatorul continua dialogul, ţinând cont de propoziţiile sale anterioare şi de propoziţia dată de sistem. Este indicat ca utilizatorul să respecte (mai mult sau mai puţin) configuraţia structurală a tiparelor învăţate. Două tipuri de rezultate au ieşit în evidenţă:
• "conforme"
• cu bifurcaţii
Voi da în continuare câte un exemplu concret pentru ambele situaţii.

Dialogul "conform" respectă "pattern"-ul tiparelor învăţate, chiar dacă există mici abateri de la acestea. Dacă am început un dialog cu:

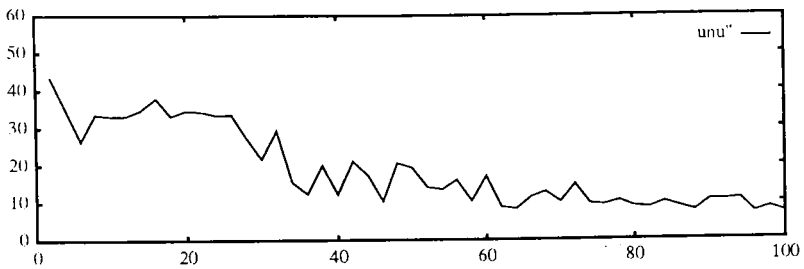
```
i_test1(0): (ν(N),w,α(P))
```

am respectat structura primei propoziţii din tiparul 3, modificând însă individualităţile M şi O cu N şi P (respectând apartenenţa la clase). Răspunsul a fost:

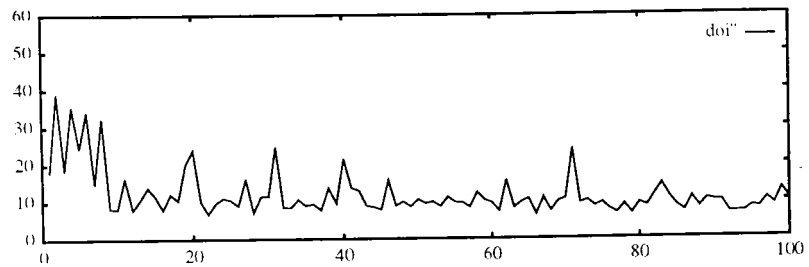
```
o_test1(1): (ξ(B),x,β(P))
```

iar continuarea:

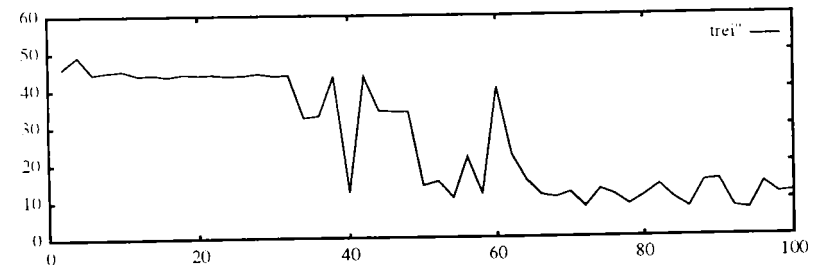
```
i_test1(1): (β(B),y,γ(C))  → o_test1(2): (β(P),z,δ(C))
i_test1(2): (γ(C),r,ε(D))  → o_test1(3): (δ(C),s,ζ(E))
```



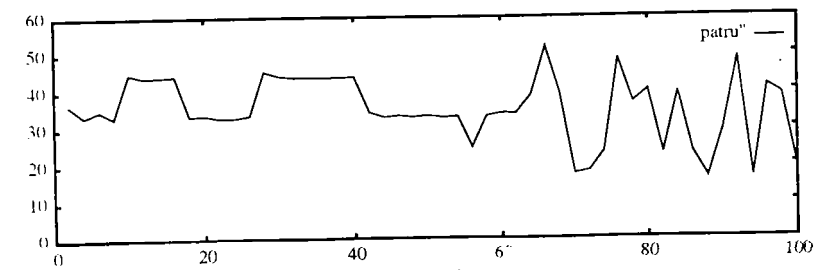
a. un singur tipar



b. doua tipare



c. trei tipare



d. patru tipare

figura 4-4.

i_test1(3): (ε(D),t,η(F)) → o_test1(4): (ζ(E),u,η(F))

Pentru a avea o traiectorie totuşi uşor schimbată, am modificat tripletul (clasă, relaţie, clasă), dar doar în măsura în care exista altul cu aceiaşi relaţie în baza de antrenare (în cazul următor, din tiparul 2):

i_test1(4): (η(F),w,δ(I)) →o_test1(5): (ζ(F),x,8,θ(I))

Conform tiparului învăţat, sistemul ar fi trebuit să răspundă cu tripletul (θ(H),x,γ(C)). Interesant este că a răspuns tot cu relaţia x, care urma în secvenţa impusă (s,u,x,z) şi cu individualităţile F şi I care apăreau în momentul anterior şi în intrare. Dacă am întrerupt regularitatea dată de succesiunea circulară a relaţiilor din intrare (r,t,w,y), sărind peste y:

i_test1(5): (σ(I),r,λ(L)) → o_test1(6): (θ(I),z,χ(J))

sistemul rămâne la ieşire pe traiectoria ştiută din tiparul 3 şi răspunzând cu un triplet care îl conţine pe z (din tiparul 2) şi cu individualitatea I, care apare şi în intrare. La pasul următor se produce resincronizarea între relaţii:

i_test1(6): (λ(L),t,ν(N)) → o_test1(7): (ζ(E),u,μ(M))

Daca am continuat să revin în "pattern"-ul tiparului 3:

i_test1(7): (ε(D),w,η(F)) → o_test1(8): (ζ(E),x,θ(I))
i_test1(8): (η(F),y,σ(I)) → o_test1(9): (θ(I),z,χ(J))

iar răspunsurile au fost cele aşteptate. Pentru că se poate continua şi pe traiectorii de dialog de mai mult de 10 paşi, am introdus propoziţii din tiparul 1:

i_test1(9): (σ(I),r,λ(K)) →o_test1(10): (χ(M),s,μ(O))
i_test1(10): (λ(K),t,ν(N)) →o_test1(11): (μ(M),u,ξ(O))
i_test1(11): (ν(N),w,α(P)) →o_test1(12): (ξ(O),x,β(A))
i_test1(12): (α(P),y,γ(B)) →o_test1(13): (β(A),Z,δ(C))

obţinând rezultatele aşteptate. Sistemul avea tendinţa să se înscrie pe o traiectorie care urma traiectoria anterioară şi era influenţată de traiectoria dată de utilizator la intrare, dar numai în cazul în care acesta respecta cât de cât traiectoriile antrenate. Sistemul era robust, reuşind în majoritatea cazurilor să revină la traiectorii învăţate, chiar dacă apăreau paşi care erau "perturbatori".

Dialogurile cu bifurcaţii apăreau atunci când utilizatorul depăşea o anumită limită în ceea ce priveşte abaterile de la traiectoriile de învăţare antrenate. Salturi neaşteptate în spaţiul de intrare, duceau la salturi şi ciclări la ieşire. Există posibilitatea ca într-un univers al discursului cu semantică, un dialog de acest tip să aibă un posibil sens, dar relativ la structurile traiectoriilor învăţate, aceste ieşiri nu pot fi asociate cu ele sub nici o formă. Unul din exemplele cele mai tipice este următorul:

```
i_test2(0): (α(P),r,β(A))  → o_test2(1): (δ(B),s,ζ(F))
i_test2(1): (δ(C),s,ζ(E))  → o_test2(2): (δ(C),s,ζ(D))
i_test2(2): (ε(E),t,η(G))  → o_test2(3): (γ(C),s,δ(D))
i_test2(3): (ζ(F),u,η(H))  → o_test2(4): (ζ(E),x,θ(F))
i_test2(4): (η(G),w,σ(I))  → o_test2(5): (α(A),w,γ(B))
i_test2(5): (ζ(G),x,θ(H))  → o_test2(6): (α(A),w,γ(B))
i_test2(6): (α(A),y,γ(B))  → o_test2(7): (δ(E),s,ζ(F))
i_test2(7): (β(B),z,δ(C))  → o_test2(8): (ζ(E),u,μ(O))
i_test2(8): (δ(D),w,σ(J))  → o_test2(9): (ζ(E),u,μ(O))
i_test2(9): (λ(L),t,ν(N))  → o_test2(10): (δ(E),u,ζ(F))
i_test2(10): (σ(I),r,λ(L)) → o_test2(11): (δ(E),u,ζ(F))
```

şi aşa mai departe, fără ca ieşirea să se fie asemănătoare cu cea învăţată sau să se sincronizeze cu intrarea, chiar dacă reveneam la o traiectorie cunoscută (reţeaua are memorie, şi odată trimisă pe o traiectorie perturbată şi menţinută mai mulţi paşi pe aceasta, nu mai revine la traiectoriile ştiute).

Dialogurile conforme arată că sistemul poate să reproducă în mod robust ceea ce a învăţat. Chiar şi dialogurile cu bifurcaţii sunt utile, pentru că ele sunt un indicator că sistemul nu acceptă intrări care nu au fost antrenate. Se poate argumenta că demonstrarea experimentală a puterii de generalizare a reţelei are nevoie de un număr mai mare de teste, recte un număr mai mare de baze de antrenare învăţate (de o certă varietate). Faptul că antrenarea este atât de costisitoare a făcut imposibilă reluarea învăţării pentru un nou set de tipare. In capitolele 5 şi 6, se va arăta cum se pot realiza sisteme hibride cu un efort mai redus, modificând arhitectura modelului.

4.5. Fenomenul de bifurcaţie la antrenare - căi de rezolvare

Spre deosebire de reţelele feedforward, cele recurente prezintă atât la funcţionare, cât şi la antrenare, un comportament nedorit, dar caracteristic sistemelor dinamice neliniare, datorat fenomenului de bifurcaţie. În termeni laici, aceasta este o modificare bruscă a stării unui sistem dinamic, datorată modificării lente şi continue a unui parametru (de regulă, un parametru de control, sau chiar o perturbaţie externă cu gradient redus). Subiectul bifurcaţiilor la antrenare este tratat în [Pineda88], iar cel al bifurcaţiilor la antrenare apare pentru prima oară in [Pearlmutter90]. Există o serie de metode propuse pentru evitarea lor la antrenare [Doya93], dar concluzia generală din literatura la zi, este că această problemă a comportamentului haotic la antrenare (un alt termen preferat pentru această problemă) este pe departe cea mai dificilă pentru cei care dezvoltă reţele recurente [Fahlman97].

4.5.1. Analiza fenomenului şi corecţiile posibile

Pentru a explica ce reprezintă bifurcaţiile, în continuare, se prezintă un exemplu. Dacă vom considera că funcţionarea unui sistem dinamic este descrisă de ecuaţia:

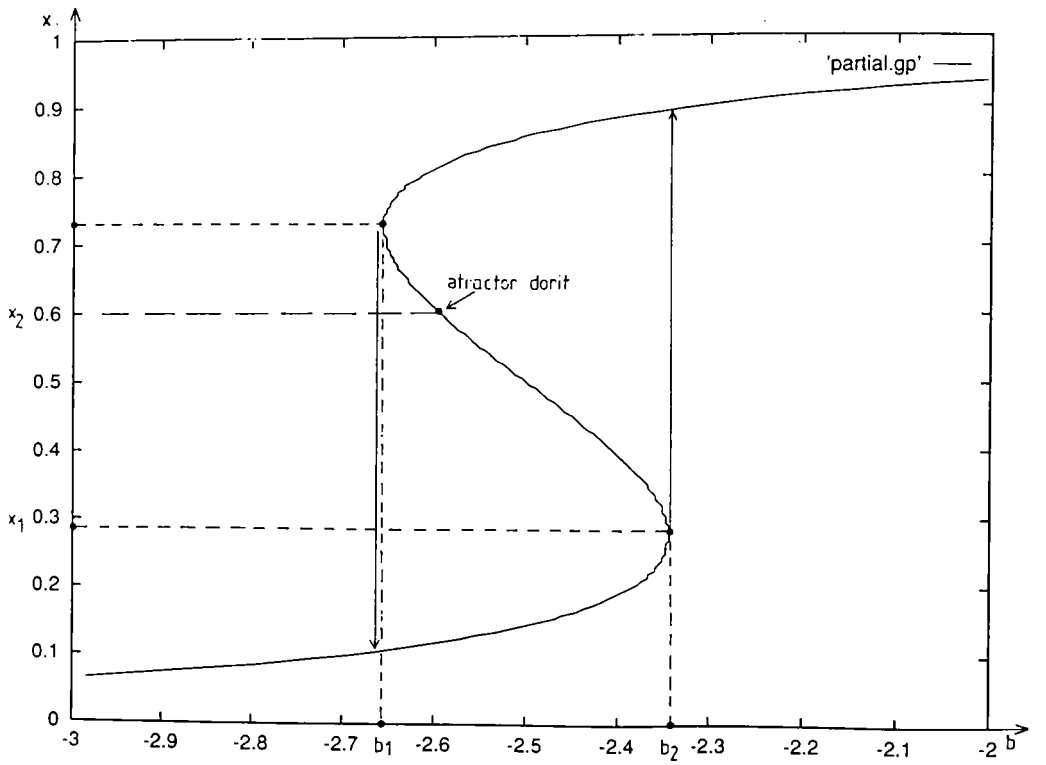


figura 4-5.

$$\sigma \cdot \frac{d\boldsymbol{x}(t)}{dt} = -\boldsymbol{x}(t) + \Gamma(\boldsymbol{W} \times \boldsymbol{x}(t)) \qquad (4\text{-}4)$$

unde:

$$\Gamma(\boldsymbol{x})_i = \frac{1}{1+e^{-x_i}} \qquad (4\text{-}5)$$

La limită se poate considera că sistemul poate fi şi discret:

$$\boldsymbol{x}(t+\Delta t) = \Gamma(\boldsymbol{W} \times \boldsymbol{x}(t)) \qquad (4\text{-}6)$$

Sa presupunem că avem un sistem cu o singură unitate neuronală:

$$\sigma \cdot \frac{dx(t)}{dt} = -x(t) + \frac{1}{1+e^{-(wx+b)}} \qquad (4\text{-}7)$$

Ieşirea x, în cazul în care b este o valoare constantă, va converge către un punct care satisface ecuaţia:

$$x = \frac{1}{1+e^{-(wx+b)}} \qquad (4\text{-}8)$$

Dacă vom interpreta această ecuaţie ca pe o funcţie $f(b)=x$, pentru valoarea $w=5.0$, graficul funcţiei va arăta ca în figura 4-5 (pentru $b \in [-3,-2]$). Toate punctele de pe curba din figură reprezintă atractori pentru sistemul lăsat liber cu o perturbaţie b constantă ($db/dt=0$). Dacă în timpul relaxării sistemului vom modifica valoarea b, printr-o funcţie de preferinţă liniară $f(t)=b$, ($f^2(t)/dt^2=0$), crescătoare sau descrescătoare, vom obţine următorul fenomen. Să presupunem că atractorul ţintă (final) este în acest caz, $x_d=0.6$, iar valoarea iniţială a perturbaţiei este $b=0$. Descrescând linear valoarea perturbaţiei, starea sistemului se va deplasa către valori mai mici, apropiate de starea dorită. Dar când vom atinge valoarea b_1 din figura 4-5, atractorul imediat următor care este pe ramura de jos a funcţiei şi se produce o modificare abruptă a valorii de stare x. Dacă prin mărirea valorii de intrare b, dorim să ajungem la atractorul ţintă, vom avea o situaţie similară când $b=b_2$ pentru că se va produce un salt pe ramura superioară. Se poate observa că în intervalul (x_1, x_2) este o zonă a spaţiului stărilor care nu poate fi niciodată atinsă pornind din afara ei, în cazul în care $w=5.0$. Este de remarcat faptul că pentru un sistem discret, avem un comportament aproximativ identic.

Din experimentele efectuate cu secvenţele de propoziţii prezentate în subcapitolul anterior, am observat că fenomene care pot fi interpretate ca bifurcaţii apar atunci când intrarea este ca şi "pattern" temporal faţă de intrările antrenate. Dacă la un moment dat se furnizează sistemului o propoziţie la care acesta nu se aştepta, iar această propoziţie este foarte "depărtată" ca şi clase, individualităţi şi mai ales ca relaţie (care trebuie să fie grupate în secvenţele repetitive din baza de antrenare), atunci se va

produce cu siguranţă un salt brusc în spaţiul de ieşire. Acest salt poate sa nu fie o bifurcaţie, dar comportamenul anormal (mai ales ciclic) care urmează, indică clar un astfel de fenomen. Pentru că antrenarea unui sistem neurodinamic (continuu sau discret) presupune evoluţia altui sistem dinamic (vezi ecuaţia 2-55) cu NxN ecuaţii, către un atractor din spaţiul ponderilor unde eroarea să fie minimă, fenomenul de bifurcaţie apare şi în această situaţie. El se manifestă prin salturi bruşte în spaţiul definit de ponderile w_{ij} şi mai ales prin creşteri bruşte ale erorii de la o epocă la alta. Aceste salturi sunt deosebit de dezavantajoase din punct de vedere al unei antrenări rapide, pentru că ele în majoritatea covârşitoare a cazurilor, plasează matricea ponderilor într-o zonă a spaţiului $\mathbf{R}^{NxN}$ care este foarte depărtată de atractorii doriţi, iar revenirea presupune un număr exagerat de epoci suplimentare. De obicei, am observat că în cazul în care apărea o bifurcaţie, task-ul de antrenare respectiv trebuia abandonat. Au fost şi situaţii în care saltul era atât de mare pe unele axe ale spaţiului ponderilor, încât s-a ajuns ca ponderile să fie numere reale nereprezentabile prin magnitudinea lor. Şi în acest caz, antrenarea era compromisă.

Există şi posibilităţi de a detecta dacă s-a produs o bifurcaţie, de a corecta traiectoria compromisă şi de a evita zonele în care s-au produs anterior bifurcaţii. Cea mai simplă metodă de a observa dacă s-a produs o bifurcaţie la antrenare este de a măsura variaţia erorii de la o epoca la alta. Dacă eroarea globală creşte brusc cu 10%, înseamnă că aproape sigur s-a produs un salt nedorit al valorii ponderilor. Dar pot să apară şi situaţii în care creşterea erorii să fie relativ mică (sub 5%), dar să se fi produs o bifurcaţie, care să modifice substanţial matricea ponderilor, fără ca acest lucru să afecteze eroarea. Un alt aspect relevat şi în [Pearlmutter95], este că după ce s-a produs o bifurcaţie care să fie un salt relativ scurt (fără a afecta prea mult eroarea), zona în care am ajuns *după* bifurcaţie, "oferă" posibilitatea imediată a unei alte bifurcaţii, care însă de această dată să modifice catastrofal ponderile şi într-un mod ireversibil.

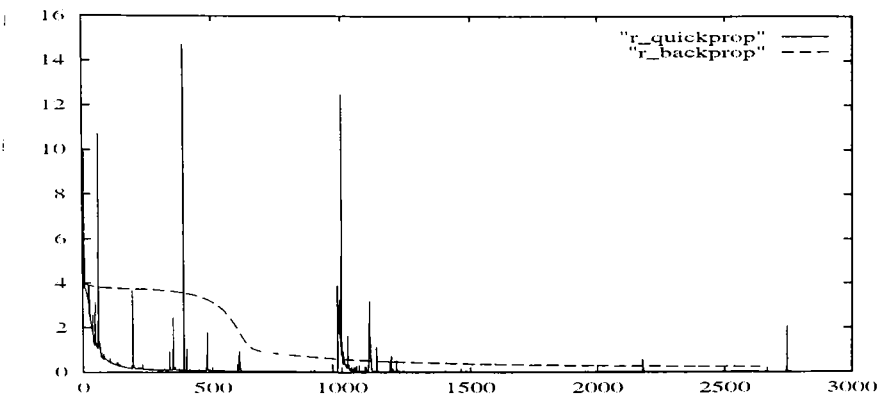
O metodă mai sigură de detecţie este calculul distanţei euclidiene între punctele din spaţiul ponderilor, atinse după două epoci succesive. Dacă vom considera indicele k pe post de contor al epocilor:

$$d = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N} (w_{ij}(k+1) - w_{ij}(k))^2} \qquad (4-9)$$

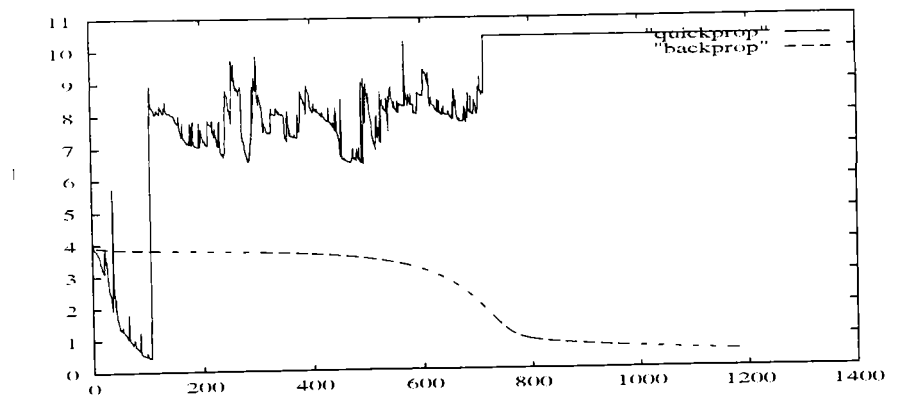
Dacă d este mai mare decât rata de învăţare înmulţită cu media aritmetică a ponderilor în pasul k, atunci în mod sigur s-a produs o bifurcaţie. Însă reciproca acestei aserţiuni nu este adevărată. Saltul poate să se producă pe foarte puţine axe (la limită chiar una), iar distanţa d să rămână mică, pentru că celelalte ponderi nu se modifică decât neglijabil. În consecinţă, trebuie verificat pentru fiecare pondere:

$$(w_{ij}(k+1) - w_{ij}(k)) < \eta \cdot w_{ij}(k) \qquad (4-10)$$

Cel mai acut dezavantaj al apariţiei bifurcaţiei, este necesitatea de a relua antrenarea (de preferinţă din alt punct iniţial setat aleator). Dacă punctul care a fost atins corespundea unei erori relativ mici, reluarea din start este deosebit de costisitoare.



a. fara bifurcatii



b. aparitia unei bifurcatii

figura 4-6

Unii autori, ca de exemplu, [Fahlman97] şi [Pearlmutter95] recomandă reluarea antrenării dintr-un punct apropiat de locul unde s-a produs bifurcaţia, modificând rata de antrenare. Pentru a realiza acest stil de reluare, este necesară menţinerea unei matrici cu ponderile anterioare, numită pivot (poate fi cu 1, 2, 3 sau mai mulţi paşi de antrenare înainte de momentul bifurcaţiei). Repornind din punctul pivot, cu o rată de antrenare schimbată (de obicei micşorată) se observă [Doya93] [Szirbik95a] [Szirbik96] că mai devreme sau mai târziu bifurcaţia se produce oricum.

O metodă folosită pe larg în căutarea euristică a soluţiilor într-un spaţiu de mari dimensiuni, este metoda de a genera aleator vecini apropiaţi ai punctului pivot. Acesta este memorat, iar traiectoria se reia dintr-o astfel de vecinătate. Dacă după un număr relativ mic de paşi de antrenare, se produce o bifurcaţie, se revine în punctul pivot şi se generează un nou vecin din care se reia antrenarea. Esenţial este aici de stabilit ce înseamnă "număr relativ mic de paşi". Dacă traiectoria s-a depărtat de punctul unde se putea produce o bifurcaţie şi se produce o nouă bifurcaţie dar în alt punct, este inutil să revenim în pivotul stabilit înaintea primei bifurcaţii, fiind necesară menţinerea unui pivot "glisant", care urmează traiectoria curentă cu un număr de paşi în întârziere, iar în cazul în care se produce bifurcaţia, se consideră ca pivot, punctul în care este pivotul "glisant" în acel moment. Necesară este în acest context şi memorarea punctului de bifurcaţie anterior, pentru a vedea dacă ne bifurcaţia următoare se produce tot în acelaşi punct. Bifurcaţii repetate în acelaşi punct, înseamnă necesitatea reluării traiectoriei dintr-un punct aflat la distanţă de acest punct "critic".

O altă metodă de folosi saltul aleator este de a genera în acelaşi episod de bifurcaţie, mai multe puncte în jurul pivotului. Pentru toate aceste puncte, se calculează eroarea globală a sistemului cu matricea de ponderi asociată punctului respectiv şi se alege ca punct de repornire a traiectoriei cel care are eroarea minimă. Pentru a evita în traiectoriile de antrenare punctele de bifurcaţii care au fost atinse deja, este recomandat ca ele să fie memorate (deşi acest lucru este foarte costisitor, în cazul meu, memoram doar ultimul punct detectat). Dacă traiectoria punctului curent intră într-o hipersferă de rază r (stabilirea valorii se face prin încercări, în experimentele mele, valoarea preferată a fost $r=0.3$), atunci se va declanşa un proces de generare aleatoare a unor puncte de reluare a traiectoriei, care să fie în afara sferei.

O metodă de accelerare a convergenţei către atractorii cu eroare globală mică a fost utilizarea regulii de antrenare quickpropagation [Fahlman89], care poate fi aplicată atât la reţelele feedforward cât şi la cele recurente. Utilizarea acesteia a dus la o scădere a erorii mult mai rapidă (vezi figura 4-6a), în raport cu metoda clasică, reprezentată cu linie întreruptă), dar şi la o creştere a frecvenţei bifurcaţiilor, care făcea antrenarea foarte dificilă (vezi figura 4-6b). O metodă de accelerare distribuită eficientă a quickpropagationului recurent, care pornea în paralel pe mai multe traiectorii şi le abandona pe cele care ajungeau la o bifurcaţie, balansând continuu efortul de calcul între procesoare, este prezentată pe larg în [Somlo96].

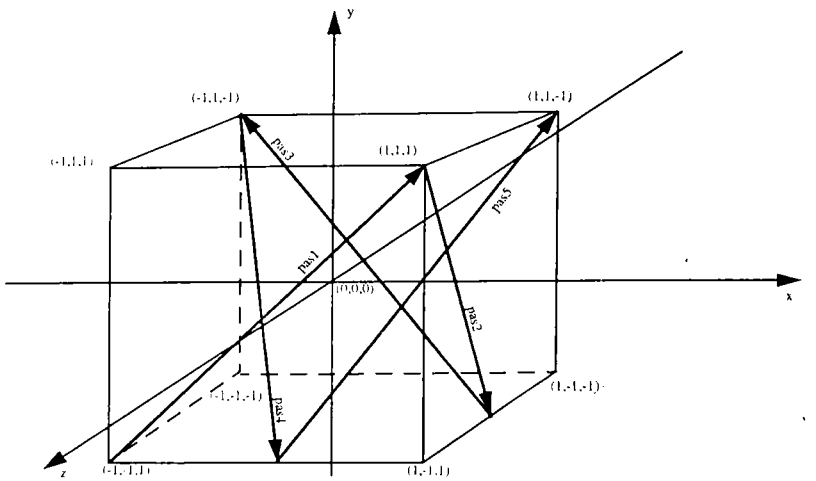
4.5.2. Influenţa formei tiparelor de antrenare

Studiind lucrările care tratau în special incidenţa fenomenului de bifurcaţie la antrenare [Tsung&Cottrell93] [Nagayama&Akamatsu94], am observat că în cazul experimentelor mele, fenomenul apărea mult mai des decât în alte genuri de experimente. Pentru că algoritmii de antrenare erau aceiaşi, cauza putea fi numai baza de antrenare. Pentru a fi sigur de aceasta, am construit o nouă bază de antrenare, care trebuia să fie cât mai diferită de de prima. Oarecum întâmplător, am ales o metodă simplă şi rapidă prin trasarea automată a unor curbe discrete într-un hipercub cu latura 2, centrat în origine. Pur şi simplu, îmi alegeam aleator un punct de start şi generam aleator alte 9 puncte succesive, ca vecinătăţi ale punctului anterior generat. Pentru ca traiectoria generată aleator să aibă totuşi o formă, am folosit regula empirică: "direcţia traiectoriei să fie oarecum aceiaşi". Aceasta se poate obţine, modificând substanţial coordonatele doar pe un număr mic de axe, pe celelalte modificarea fiind mică (dar nu şi neglijabilă). Pentru calibrarea parametrilor generatorului, am folosit un instrument de lucru în 3D, care genera rezultate de tipul celor ilustrate în figura 4-7b. Caracteristic pentru traiectoriile generate este sunt pseudo-continue (o interpolare spline de exemplu, duce la o curbă netedă). Pentru că alegeam puncte iniţiale cu valori mici, curbele erau aproape de origine, adică de centrul cubului. Pentru a evita aglomerarea tiparelor într-o singură zonă, am construit intenţionat şi curbe care erau apropiate de frontiera hipercubului.

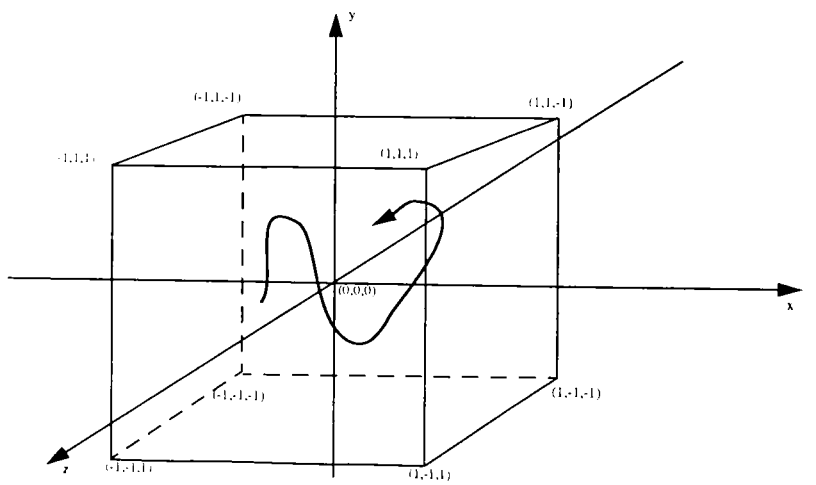
Am comparat rezultatele antrenării, relativ la numărul de bifurcaţii rezultate. Experimentele au fost făcute cu cu grupe diferite de tipare, pentru un număr fixat de epoci (450), fără ca eroarea să fie monitorizată, doar numărul de bifurcaţii, care era notat după 150 de epoci. Tiparele provenite din grafuri conceptuale sunt notate GC, iar cele generate aleator AL. Pentru că observasem din experimentele de până atunci că la o rată de antrenare mică, bifurcaţiile apar mai des, în toate experimentele am folosit o rată de învăţare constantă de valoare foarte mică (0.05). Rezultatele au fost următoarele:

TIPARE	număr de bifurcaţii la 150 de epoci	număr de bifurcaţii la 300 de epoci	număr de bifurcaţii la 450 de epoci
2 tipare GC	8	11	25
2 tipare AL apropiate de origine	0	0	0
2 tipare AL depărtate de origine	1	1	2
4 tipare GC (toate)	12	22	##
4 tipare AL	1	1	2
4 tipare AL apropiate de origine	0	0	1
4 tipare AL depărtate de origine	2	2	5

(## - matricea de ponderi a ajuns în mod repetat să conţină valori nereprezentabile)



a. o traiectorie CG



b. o traiectorie NC

figura 4-7.

Chiar şi din acest unic set de date experimentale se poate deduce că rata de apariţie a bifurcaţiilor pentru tiparele GC este foarte mare, pe când tiparele AL au performanţe excelente relativ la stabilitatea antrenării.

Explicaţia este dată şi în [Pearlmutter95], ca rezultat al unor experimente asemănătoare. Dacă tiparele de antrenare sunt curbe netede, şi pseudo-continue (fără salturi semnificative între două puncte succesive) plasate într-o zonă centrală a hipercubului de stare al reţelei, antrenarea se desfăşoară de regulă fără bifurcaţii. Matematic, explicaţia este dată de liniaritatea funcţiei de transfer sigmoidale în jurul originii. Astfel, sistemul dinamic reprezentat de mecanismul de antrenare poate fi considerat cvasi-liniar, iar un astfel de sistem are un comportament instabil doar în situaţii bine determinate.

În contrast cu tiparele netede şi centrate, cele rezultate din grafuri conceptuale (pe baza translaţiei tensoriale), sunt formate din secvenţe de tensori formaţi în majoritate din valori binare (în cazul tangentei hiperbolice 1 şi -1), care mai pot conţine doar valorile 0.5, 0.75, -0.75 şi -0.75, induse de formula (3-23). Pentru o schemă tensorială simplificată (cu rezultate de gradul 3), o ilustrare a unei astfel de traiectorii este dată în figura 4-7a. Tiparele de tip GC sunt succesiuni de puncte care formează traiectorii extrem de discontinue în spaţiul stărilor reţelei, având chiar şi salturi între "colţurile", hipercubului, ambele caracteristici (discontinuitatea şi depărtarea de zona liniară din jurul originii) favorizând într-un mod exagerat bifurcaţiile la antrenare.

Observaţia pe care am făcut-o după aceste experimente, a fost că dificultatea antrenării unor astfel de sisteme rezidă nu atât în duratele necesare pentru atingerea erorii impuse, cât în evitarea fenomenului de bifurcaţie, care măreşte cu un ordin de mărime aceste durate.

4.5.3. O schemă extinsă de translatare

O soluţie pentru evitarea bifurcaţiilor, care pare cea mai simplă şi directă, este de a folosi un alt gen de tipare decât tiparele GC şi anume tipare care prin interpolare spline reprezintă curbe netede şi centrate în hipercubul stărilor reţelei (care sunt similare cu tiparele AL). Dificultatea este că aceste noi tipare trebuie sa conţină informaţia din grafurile conceptuale iniţiale. Am numit această clasă de tipare, NC (netede şi centrate).

Forma tiparelor GC este datorată metodei de translaţie tensorială. Pentru a obţine tipare NC, ar fi necesar să modificăm sau să completăm metoda de translaţie. Deşi am căutat o metodă automată de transformare, nu am reuşit să o dezvolt. Am folosit o metodă semi-manuală, în care alocam puncte în centrul hipercubului pentru propoziţiile din baza de antrenare, astfel încât propoziţiile cu semantică apropiată să fie apropiate, iar cele cu semantică diferită, să fie depărtate. Pentru a avea tipare care să reprezinte curbe netede, am alocat punctele astfel încât secvenţele de propoziţii sa genereze astfel

de traiectorii. Dacă două secvenţe conţineau propoziţii comune, cele două curbe de antrenare se intersectau.

La început am folosit instrumentul 3D, pe care l-am folosit pentru generarea tiparelor AL. Am păstrat dimensiunea reţelei tot la 768 unităţi, dar am redus drastic dimensiunea intrării şi a ieşirii, din motivul simplu ca erau dificil de generat puncte într-un spaţiu cu dimensiune atât de mare. Experimental, am ajuns la concluzia că sistemul este cel mai simplu de antrenat, utilizat (iar tiparele NC de generat) atunci când dimensiunea intrării şi ieşirii sunt de 80 adică aproximativ 10% din totalul unităţilor, valoare ideală dată şi în [Pineda88]. O primă metodă ad-hoc pe care am folosit-o a fost divizarea componentelor vectorilor de natură binară cu 2.0, 2.5 şi 3.0, pentru a aduce valorile acestora in zona centrală a hipercubului. Deşi numărul de bifurcaţii a scazut (la 15 bifurcaţii pe primele 300 de epoci), acest rezultat nu era cel scontat, dorind o reducere cu un ordin de mărime. Obţinusem tipare relativ centrale dar nu şi netede, iar dimensiunea nu a fost redusă de 10 ori, aşa cum mi-am propus.

Pentru transformarea vectorilor obţinuţi prin translatare tensorială din grafuri conceptuale în vectori ai tiparelor NC dorite, este necesară o schemă de mapare, care să transforme vectorii de natură binară în vectori reali, cu componente cuprinse între -0.5 şi 0.5 şi de o dimensiune 10 ori mai mică. Din moment ce maparea se face propoziţie cu propoziţie, se poate constitui un tabel de mapare, unde pentru fiecare propoziţie există o intrare conţinând propoziţia în forma sa simbolică şi vectorul real ales pentru a face parte din tiparele NC. Nu ar mai fi necesară translatarea tensorială, pentru că pur şi simplu prin "pattern-matching", propoziţia ar putea fi identificată în tabel şi s-ar furniza reţelei vectorul real asociat. La ieşire, s-ar putea calcula distanţa între punctul reprezentat de vectorul de ieşire în spaţiul 80 dimensional şi toate punctele descrise de vectorii din tabel, iar cel mai apropiat este cel asociat propoziţiei care va fi selectată ca şi ieşire. La funcţionarea sistemului, apar însă propoziţii care nu fac parte din baza de antrenare şi care nu aparţin tabloului static descris mai sus. Schema de mapare propusă este în consecinţă inutilizabilă.

Soluţia găsită de mine a fost extinderea schemei de translaţie tensoriale prin două reţele neuronale suplimentare, menită să transforme vectorii eminamente binari în vectori reali şi invers. La intrare, după aplicarea translaţiei tensoriale şi obţinerea vectorului cu componente binare, acesta din urmă este furnizat unei reţele neuronale feedforward cu un număr de intrări egal cu dimensiunea de ieşire a formulei tensoriale de translaţie şi cu 80 de ieşiri. Ieşirile se normalizează pe intervalul (-1.0,1.0) de pe intervalul specific al tiparelor NC, (-0.5,0.5). Reţeaua feedforward este antrenată cu perechi (propoziţie, vector-real) din tabloul static prezentat anterior. Pentru translaţia inversă, se foloseşte o altă reţea feedforward cu 80 de intrări şi un număr de ieşiri egal cu intrarile primei reţele feedforward. Această reţea se antrenează cu perechi (vector-real, propoziţie) construite din acelaşi tablou static. Dacă o nouă propoziţie, care nu apare în tabel, este furnizată sistemului în faza lui de funcţionare, aceste două reţele suplimentare vor genera punctele potrivite în spaţiul redus dimensional al tiparelor NC şi în spaţiul tiparelor GC.

Apare un avantaj legat de generarea tiparelor NC. Pentru că aceste două reţele sunt capabile de generalizare, nu este necesar să le antrenăm cu toate perechile de date din tabloul static. Corespunzător, nu este necesar să generăm toate punctele care formează tiparele NC. Am generat doar jumătate din acestea, restul fiind obţinut prin generalizare, obţinându-se chiar o netezime mai bună în acest caz (pentru curbele NC), decât în cazul în care erau generate semi-manual. Alegerea jumătăţii potrivite din totalul de propoziţii am făcut-o astfel încât să acopere toate simbolurile din vocabularul folosit. Alt avantaj major este posibilitatea folosirii unei scheme de translaţie care să lucreze cu reprezentări cât mai puţin "înghesuite", pentru simbolurile din V_F, V_R şi V_L (vezi capitolul 3). Aceasta va îmbunătăţi considerabil performanţa translatării inverse, din vector de reprezentare cu componente binare în graf conceptual. Dimensiunea vectorilor din tiparele GC va creşte foarte mult, dar nu va afecta dimensiunea reţelei recurente. Relativ la vocabularul folosit şi la noile reprezentări binare, dimensiunea acestor vectori devine egală cu 12634. Schema sistemului îmbunătăţit este dată în figura 4-8.

Structura reţelei feedforward FFI am stabilit-o la 12634-100-80 unităţi, cu o bază de antrenare de 20 de tipare. Resursele necesare pentru o astfel de reţea sunt relativ mari (aproximativ 14 MBytes pentru matricile de ponderi), antrenarea necesitând peste 8000 de epoci, adică aproximativ 59 de ore de rulare pe singur procesor SPARC - algoritmul fiind neparalelizat - folosind un program de antrenare clasic "quickpropagation", o variantă a algoritmului backprop (importat de pe "site"-ul Neurosoft Repository de la Carnegie-Mellon University, Pittsburgh). Rezultatul a fost foarte bun, eroarea putând fi redusă sub 3% din prima încercare. Stabilirea dimensiunii stratului ascuns s-a facut anterior, la antrenarea reţelei FFO (vezi figura 4-8), care a fost dezvoltată înaintea reţelei FFI.

Reţeaua FFO are o particularitate care mi-a atras atenţia şi din acest motiv m-am gândit la utilizarea unui alt tip de reţea. La antrenarea reţelelor feedforward, pierderea de timp se datorează în principal găsirii unui număr potrivit de neuroni pentru stratul ascuns. De obicei se începe cu un număr mic de unităţi, se testează dacă eroarea impusă poate fi atinsă într-un număr rezonabil de epoci, iar dacă nu se poate, se măreşte numărul de unităţi de pe stratul ascuns cu o fracţiune şi se porneşte procesul de antrenare din start. Aceste cicluri de reglare a dimensiunii reţelei pot fi deosebit de costisitoare ca şi timp de calcul irosit. Ele sunt necesare pentru a stabili un necesar de memorie optim pentru reţea. Relativ la dimensiunea stratului ascuns, necesarul de memorie pentru ponderi poate să difere foarte mult. De exemplu, regula empirică folosită de obicei pentru stabilirea primei valori alese este:

[1/2*(nr_neuroni_intrare + nr_neuroni_ieşire) + nr_tipare]/2

Pentru reţeaua mea specifică (FFO) dimensiunea stratului ascuns dată de formula de mai sus, ar fi fost de 3189 de unităţi. Ceea ce înseamnă o primă matrice de ponderi de 12634*3189 de valori în virgulă flotantă, adică un necesar de aproximativ 330 de MBytes, iar pentru a doua matrice 3189*80 de ponderi (aprox. 2 MBytes). Comparate cu valorile pe care le-am utilizat (aprox. 14 MBytes pentru ambele matrici), ac

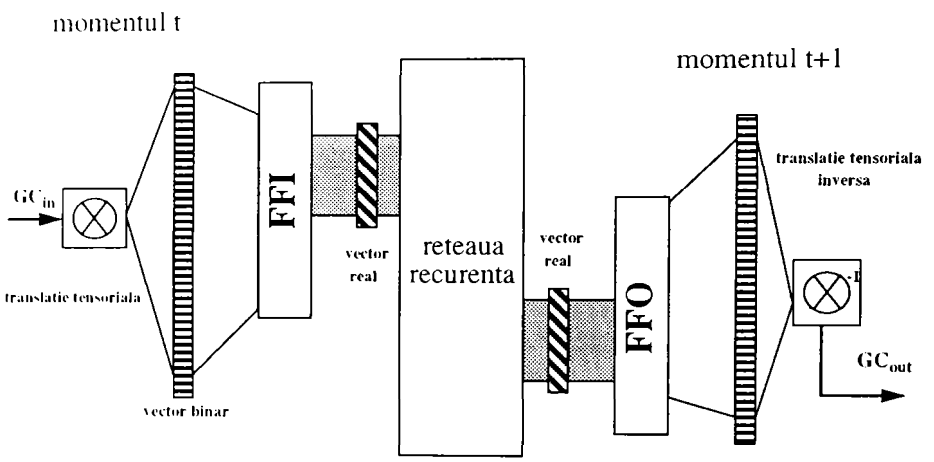


figura 4-8.

este valori sunt mult prea mari. Dimensiunea optimă (aproximativ 100 de unităţi pe stratul ascuns) a fost detectată cu o variantă a algoritmului de antrenare "cascade-correlation" [Fahlman&Lebiere91], care îşi ajustează din mers numărul de neuroni de pe stratul ascuns. Dezavantajul este că o arhitectură în cascadă funcţionează mult mai încet (ca şi când ar fi o reţea cu un număr de straturi egal cu numărul de neuroni ascunşi), dar antrenarea este relativ rapidă (m-a costat aproximativ 80 de ore pe o maşină SPARC). Ideea de a folosi "cascade-correlation" mi-a venit datorită faptului că acest algoritm se comportă execelent în situaţia când intrările sunt reale, iar ieşirile binare, vezi [Fahlman&Lebiere91].

4.5.4. Concluzii

Repetând antrenarea reţelei recurente cu tiparele NC (pe maşina ZOO), timpul de antrenare s-a redus cu aproximativ un ordin de mărime. Acest fapt s-a datorat reducerii la jumătate a numărului de epoci necesar pentru atingerea erorii impuse (5%), dar mai ales reducerii drastice a numărului de bifurcaţii la antrenare. În 735 de epoci s-au produs doar de trei ori bifurcaţii, corectarea traiectoriei fiind eficientă, fără a se mai obţine valori exagerat de mari în matricea de ponderi şi fără a fi necesară reluarea din start a procesului de antrenare. Verificând capacitatea de generalizare a reţelei, cu cele două tipare de test prezentate anterior am obţinut aproximativ acelaşi gen de rezultate. Ceea ce validează această structură îmbunătăţită a sistemului.

Bineînţeles, există un preţ pentru această evitare forţată a fenomenului de bifurcaţie la antrenare. Este necesară introducerea a două noi reţele, care ocupă un spaţiu important de memorie şi care cer şi ele un timp destul de lung de antrenare, chiar mai lung decât al reţelei recurente - datorită variantei neparalelizate. Însă în termeni absoluţi acest timp este mult mai mic decât timpul necesar pentru antrenarea reţelei recurente (aproximativ cu două ordine de mărime). Posibilitatea de a folosi un spaţiu cu foarte multe dimensiuni pentru tiparele GC duce la tentaţia de a folosi un vocabular mult mai bogat, dar aceasta ar conduce la o reprezentare "înghesuită" a simbolurilor din vocabular şi din o serie de experimente, am observat că antrenarea reţelelor FFI şi FFO (mai ales), devine extrem de costisitoare.

Putem rezuma metodologia propusă la următorii paşi:

- alegerea vocabularului (clase, individualităţi şi relaţii)
- alegerea a două reprezentări pentru $\mathbf{V_F}$, $\mathbf{V_R}$ şi $\mathbf{V_L}$ (una "largă" şi alta "înghesuită").
- stabilirea propoziţiilor folosite şi a tiparelor de tip secvenţe de propoziţii.
- determinarea dimensiunii vectorilor GC obţinuţi prin translatare tensorială pentru ambele reprezentări, notate D pentru reprezentarea "largă" (dimensiune3125 poziţii) şi d pentru cea "înghesuită" (dimensiune 768).
- stabilirea dimensiunii reţelei recurente egală ca număr de unităţi cu d (intrarea şi ieşirea sunt $n=d/10$).
- jumătate din propoziţii (în mod obligatoriu trebuie să acopere tot vocabularul folosit) sunt asociate cu puncte din spaţiul n dimensional, astfel încât tiparele să devină în acest

spaţiu netede şi centrate în zona originii.
• stabilirea dimensiunii reţelelor FFI (*D* intrări şi *n* ieşiri) şi FFO (*n* intrări şi *D* ieşiri).
• stabilirea prin "cascade-correlation" a numărului de unităţi ascunse din FFO (notat *h*), prin antrenarea până la atingerea unui nivel propus de eroare (recomand 3%).
• antrenarea reţelei FFI cu "quickpropagation", cu aceleaşi perechi ca şi pentru FFO, pe o structură fixă cu *D-h-n* unităţi până la atingerea nivelului propus de eroare (tot 3%).
• antrenarea reţelei recurente, până la un nivel de eroare de 5%.

Cele mai costisitoare activităţi sunt antrenarea reţelei recurente, care cere un enorm efort computaţional (96% din total), şi asocierea propoziţiilor la vectori în spaţiul *n* dimensional, care cere un efort enorm din partea celui care dezvoltă sistemul (90% din total), în situaţia când această operaţiune se face semi-manual. Automatizarea asocierii structurilor simbolice la un spaţiu euclidan, astfel încât să se păstreze cerinţele tiparelor NC este o problemă care ar putea reprezenta o direcţie de cercetare separată. Există rezultate în acest sens (vezi de exemplu [Crucianu&Memmi92]), dar un excelent tutorial recent publicat pe această temă [Pfeifer96] nu indică nici o metodă pe care aş fi putut-o folosi în contextul specific al modelului dezvoltat de mine.

5. Modelul multimodular omogen

Sistemul prezentat în capitolele precedente, se bazează pe o singură reţea recurentă de mari dimensiuni, un vocabular de o dimensiune relativ scăzută şi o schemă de translatare bidirecţională. Se folosesc două tipuri distincte de date pentru reprezentarea cunoştinţelor: vectorii de activare neuronali şi grafurile conceptulale. Este un sistem omogen, pentru că procesarea propriu-zisă - raţionamentul - se desfăşoară prin calcule specific conexioniste. Natura parţial hibridă este dată de restaurarea de tip "translaţie-retranslaţie", care este aplicabil doar modelul în buclă închisă şi de de verificările posibile de consistenţă logică (dar acestea nu sunt neaparat necesare).

Complexitatea antrenării acestei reţele este cu aproximaţie N^4P, iar în cazul în care lungimea tiparelor temporale discrete este aproximativ egala cu N, complexitatea este de gradul 5 înmulţită cu P - numărul total de tipare de antrenare. Timpul necesar pentru antrenarea celor două reţele feedforward este nesemnificativ în raport cu timpul necesar reţelei recurente. Pentru că nu există soluţii prea bune pentru reducerea complexităţii algoritmilor de antrenare recurenţi [Pearlmutter 95], singura cale de a îmbunătăţi sistemul este să reducem N, numărul total de neuroni ai reţelei recurente. Pare o soluţie paradoxală, pentru că scade capacitatea de antrenare, odată cu dimensiunea vocabularului. Dar există şi alternativa *distribuirii*.

Idea mea a fost să folosesc mai multe reţele recurente mici [Szirbik 95b], care au fiecare, un vocabular propriu, foarte redus. Dar dacă fiecare reţea mică lucrează cu câteva cuvinte pe care numai ea le poate procesa, există şi cuvinte care pot fi acceptate doar de câteva reţele şi există un fond comun de simboluri utilizabile de toate reţelele, atunci numărul total de simboluri poate fi foarte mare. Fiecare reţea recurentă este însoţită de cele două reţele feedforward (FFI şi FFO) necesare pentru completarea translaţiei simbolic-neuronale şi neuronal-simbolice, astfel încât tiparele de antrenare să nu faciliteze apariţia bifurcaţiilor la antrenare. În prezentarea care urmează, am făcut abstracţie de aceste două reţele, fără să intru în amănunte legate de implementarea şi performanţele lor. Aceasta pentru că antrenarea lor este banală, singura problemă fiind

asocierea unor puncte din spaţiul N dimensional la propoziţiile folosite. Dimensiunea reţelei recurente care efecuează procesarea cognitiv-neuronală este întotdeauna în funcţie de vocabularul asociat ei, mai exact rezultă din formula de translaţie tensorială care foloseşte cea mai compactă reprezentare pentru simboluri (vezi capitolul 3). Pentru că toate reţelele folosite sunt identice ca structură, am numit această arhitectură, *modelul multimodular omogen*. Este important de menţionat că acesta este un contra-exemplu relativ la metodologia propusă în teză, iar dezavantajele sale sunt analizate în profunzime, pentru a constitui argumentarea necesară pentru modelul prezentat în capitolul următor.

5.1. Consideraţii dimensionale

Cel mai evident avantaj al unei arhitecturi cu mai multe reţele recurente este posibilitatea utilizării unui vocabular de mari dimensiuni. Vom numi în continuare reţelele folosite (împreuna cu partea aferentă de translaţie) *module conexioniste* - MC. Putem considera că fiecare modul poate avea la intrare sau genera la ieşire grafuri conceptuale care sunt compuse dintr-un vocabular limitat la maximum 20 de cuvinte. Am observat că în general, pentru un set relativ mare de propoziţii (314), distribuţia morfologică optimă, pentru a putea construi propoziţii cu sens normal (ne-metaforic sau prea factual), este următoarea: 30% verbe, 35% substantive, 35% adjective. Avem în cazul vocabularului de 20 de cuvinte: 6 verbe, 7 substantive şi 7 adjective. Aplicând formula tensorială prezentată în cap.3.1. vom obţine o reţea cu aproximativ 200 de unităţi (40000 ponderi = 320kBytes), mult inferioară ca dimensiuni decât una cu 1000 de unităţi (1 milion de ponderi = 8MBytes) care poate utiliza un vocabular de maximum 50 de cuvinte. În acelaşi spaţiu de 8MBytes pot fi memorate ponderile a aproximativ 25 de module "mici". Să presupunem următoarea situaţie: Avem 3 MC, iar distribuirea vocabularului este următoarea:

- 5 cuvinte "locale" fiecărui modul
- 10 cuvinte "universale", care pot fi utilizate de toate modulele
- 5 cuvinte "comune" două câte două (vezi figura 5-1)

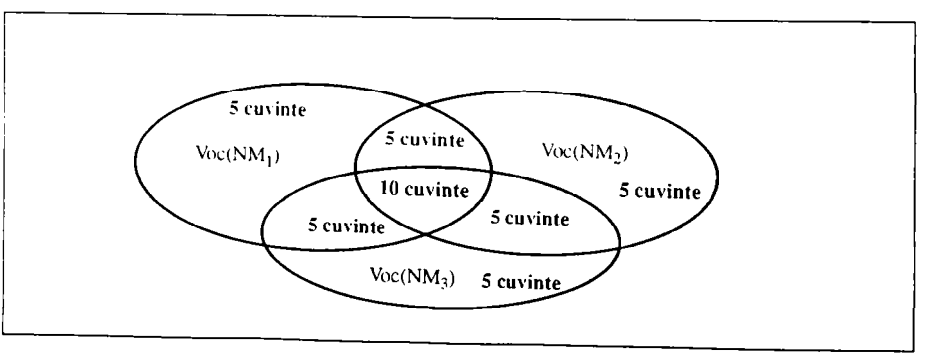


figura 5-1

Se poate observa că avem un total de 40 de cuvinte. Dacă numărul de module este crescut la 25, şi folosim acelaşi stil de distribuţie, vom avea un total de:

25 5 + 12 5 + 10 = 200 de cuvinte

Folosind 400 de module (necesar de memorie = 128 MBytes) şi considerând că cuvintele comune (nu cele "universale") sunt distribuite pe grupe mai mari (aproximativ de 50 de module), atunci vom avea un total de:

400 5 + 50 5 + 10 = 2260 de cuvinte

care este un vocabular foarte mare. Însă vom vedea în continuare, din analiza modului de funcţionare că nu este avantajos să avem un număr atât de mare de cuvinte locale (individuale unui singur modul) - acestea reprezentând majoritatea din cele 2260 de cuvinte.

5.2. Comunicarea între module

Comunicarea se desfăşoară prin intermediul unei zone comune, numită "blackboard" (BB). Toate modulele au acces la această zonă putând citi şi scrie propoziţii din/pe BB. Din punct de vedere temporal, se pot alege două variante constructive:

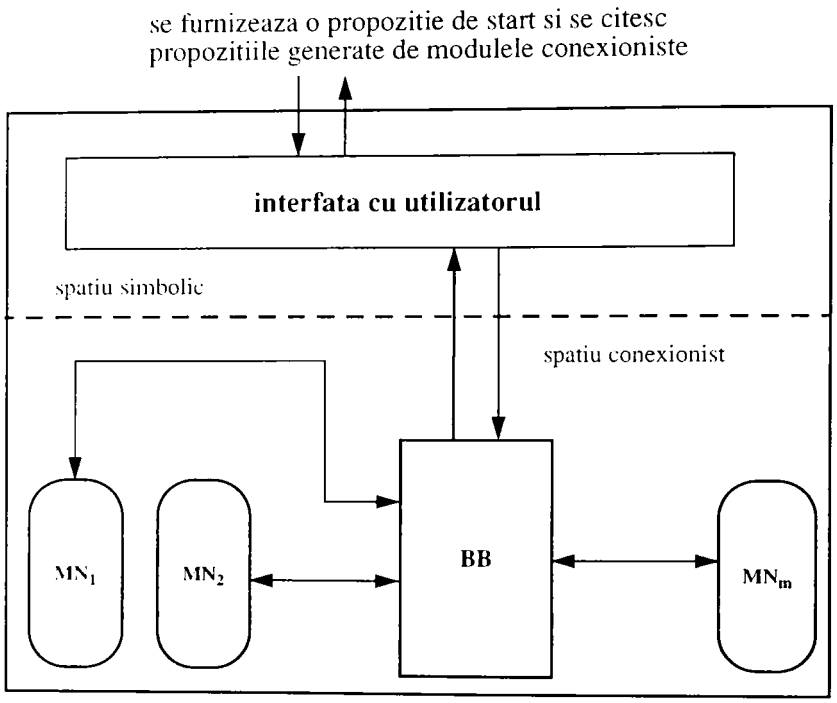
- varianta sincronă
- varianta asincronă.

Din motive de simplitate am ales modelul sincron, cel asincron neoferind nici un avantaj din moment ce informaţia era transmisă în pachete delimitate de conţinutul propoziţiei. Schema de funcţionare a unui astfel de set de module este dată în figura 5-2. Intuitiv, funcţionarea este următoarea: se scriu pe BB informaţiile (propoziţiile) de start, apoi modulele care pot citi aceste propoziţii (adică toate cuvintele din respectiva propoziţie fac parte din vocabularul modulului respectiv) le acceptă la intrare şi evoluează din starea iniţială într-o altă stare staţionară. Se poate întâmpla ca starea finală să fie identică cu starea iniţială. În acest caz se consideră că modulul nu a generat nici o ieşire (vom numi acest modul, *staţionar*). Propoziţiile care au fost citite de pe BB de către module nestaţionare, se consideră "consumate" şi sunt şterse.

Modulele care sunt nestaţionare şi au consumat câte o propoziţie din BB vor genera o propoziţie de ieşire. Setul de propoziţii astfel obţinut va fi scris pe BB şi va reprezenta pachetul de informaţie necesar un nou ciclu ca cel descris, pe care îl vom numi pas de raţionament. În momentul citirii propoziţiilor de pe blackboard, apar o serie de probleme legate de prioritate (ordinea în care sunt citite propoziţiile).

i. o propoziţie poate conţine un set de cuvinte care poate fi acceptat de mai multe module conexioniste. În acest caz, am decis ca toate modulele care pot accepta propoziţia să o şi citească.

ii. pot apărea la un moment dat pe BB mai multe propoziţii care pot fi acceptate de un singur modul. În acest caz vom avea o coadă de aştepare, pentru că un modul nu poate



MN_i = module neuronale
BB = blackboard

figura 5-2.

accepta decât o singură propoziţie într-un pas de raţionament. Se poate imagina şi o schemă în care fiecare modul îşi consumă într-un singur pas toate propoziţile din coada de aşteptare, dar am observat că în acest caz, modulele tind să se distanţeze la nivel semantic, propoziţiile de pe BB find foarte diferite şi neducând la un fir de raţionament unitar, ci la generarea unor propoziţii foarte diferite între ele. Ca raţionamentul să poate evolua, e necesar ca toate modulele să fie grupate la un moment dat în aceiaşi zonă semantică.

Dacă folosim schema cu o_propoziţie/un_pas_de_raţionament, se pot imagina mai multe scheme de rezolvare a priorităţii propoziţiilor. O soluţie "brută" este aplicarea unei operaţii de "join" grafurilor conceptuale din coada de aşteptare. Există însă cazuri în care această operaţie să nu ducă la un singur graf final, pentru că între anumite grafuri din coada de aşteptare, operaţia de "join" nu este posibilă. Evident, această metodă este posibilă numai dacă reprezentarea propoziţiilor se face pe BB în formă de grafuri conceptuale. Când operaţia de "join" ducea la un singur graf final, se producea o "aglomerare" a informaţiei (majoritatea elementelor vectorului GC erau 1) la intrarea reţelei neuronale FFI, şi de obicei reţeaua recurentă evolua într-o stare stabilă de unde nu mai putea fi scoasă de nici o altă intrare (ca un fel de saturare), adică devenea tot timpul staţionară.

Altă soluţie ar fi superpoziţia vectorilor conexionişti care reprezintă propoziţii. În acest caz, nu mai e necesară reprezentarea la nivel de graf a propoziţiilor pe BB. Superpoziţia se face prin însumarea vectorilor de intrare. Dar comparând rezultatul obţinut prin translatarea dupa o operaţie de "join", cu vectorul obţinut prin superpoziţie (plecând de la aceleaşi propoziţii iniţiale), se poate observa o mare diferenţă. Acest lucru este normal, pentru că metoda superpoziţiei directe este o simplificare a metodei de translatare prezentată în capitolul 3. Efectuând experimentele cu ambele variante, am ajuns la concluzia că nici una din ele nu este potrivită. Prima este aplicabilă doar când propoziţiile sunt în formă de graf conceptual şi operaţia de "join" este posibilă între toate propoziţiile din şirul de aşteptare al unui modul (rezultă un singur graf). A doua metodă simplifică în mod grosolan procesul de translatare al unui text într-un vector, pierzând foarte multă informaţie.

Alternativa era ordonarea propoziţiilor care pot fi acceptate de un modul (sau mai multe module simultan) în coada de aşteptare, ele fiind consumate una câte una. A fost deci necesară găsirea unor criterii de ordonare. Primul criteriu de ordonare este cât se poate de evident, ordinea în coada de aşteptare fiind dată de ordinea venirii (FIFO sau LIFO). Am ales FIFO (first-in, first-out). Dar în acest caz, se poate întampla ca mai multe propoziţii care aparţin aceleiaşi cozi de aşteptare să apară în acelaşi pas de raţionament (ca ieşire simultană a mai multor module). În acest caz, am adoptat o stabilire conform unei distribuţii aleatoare uniforme a ordinii de intrare în coada de aşteptare, pentru că nu am găsit nici un criteriu viabil pentru stabilirea ordinii.

Mai apare un fenomen de care a trebuit să ţin cont. De multe ori, o propoziţie generată într-un pas de raţionament este deja pe BB (undeva într-o coadă de aşteptare). Cea mai simplă metodă de evitare a includerii multiple a unei propoziţii din coada de

aşteptare era verificarea dacă propoziţia există deja şi în acest caz se ignora noua apariţie. Am ajuns la concluzia că o astfel de abordare pierde din informaţie, pentru că apariţia succesivă a aceleiaşi propoziţii (informaţii) are şi ea semnificaţia ei. Am decis ca în astfel de cazuri, propoziţia să fie "avansată" în coada de aşteptare, până pe primul loc. Şi în această situaţie poate să apară o situaţie de conflict. De exemplu, se poate întampla ca într-o coadă de aşteptare să avem trei propoziţii pe poziţii diferite şi acestea apar din nou într-un pas de raţionament:

$$....,P_i,....,P_j,...,P_k,...,P_{first} \ (\rightarrow \textit{sensul cozii})$$

În algoritmul de căutare al cozii potrivite pentru o propoziţie nou generată şi constatarea existenţei anterioare a propoziţiei respective, aceste operaţii se vor face secvenţial (de regulă în ordinea de numerotarea a modulelor care au generat propoziţii). Să presupunem că ordinea în care tratăm propoziţiile este P_k, P_j, P_i. În acest caz, aplicând metoda descrisă mai sus, coada va avea următorul aspect:

$$....,P_{first},P_k,P_j,P_i$$

pentru că tratând propoziţia P_k prima dată, am pus-o pe prima poziţie, care apoi a devenit a doua şi apoi a treia, ajungând la o ordine total diferită de cea dată de paşii de raţionament. Acest lucru se datorează numai şi numai faptului că ordinea de tratarea fost cea descrisă. În alta ordine s-ar fi obţinut o prioritate dată de această nouă ordine. Ca să păstrez totuşi ordinea iniţială, am grupat propoziţiile deja existente într-o nouă coadă provizorie, care păstra ordinea iniţială:

$$P_i, P_j, P_k$$

indiferent de ordinea tratării (în acelaşi pas) acestor propoziţii şi am concatenat această coadă provizorie cu coada iniţială, din care s-au şters propoziţiile existente din coada provizorie:

$$....,P_{first},P_i,P_j,P_k \ (\rightarrow)$$

Se poate observa că rezolvând toate aceste probleme, comunicarea între module şi suportul ei au devenit o parte importantă a sistemului.

În ceea ce priveşte nivelul de reprezentare, putem alege între doua strategii fundamental diferite pentru a realiza comunicarea între module, şi anume:
- comunicarea la nivel conexionist
- comunicarea la nivel simbolic (pe care am folosit-o).

În primul caz, informaţie care circulă între module este reprezentată prin vectori de activare neuronali. Apar două probleme: propagarea erorilor (conform regulii divergenţei - vezi subcapitolul 4.2) dar mai ales devine foarte dificilă *identificarea modulelor potrivite care pot accepta o anumita propoziţie*. La nivel simbolic, am reprezentat propoziţiile ca şi grafuri conceptuale (sau într-o variantă mai simplă, ca şi

structuri predicative).

În cazul în care restaurăm ieşirea de tip vectorial a unui modul în forma de graf conceptual (şi chiar facem şi o verificare aplicând principiul rezoluţiei dacă propoziţia nou creată nu duce la o inconsistenţă), eliminăm micile erori care pot apare în structura unui vector de ieşire. Bineînţeles, această structură simbolică trebuie transformată din nou în vector pentru a fi aplicată ca intrare la o altă reţea neuronală. Sistemul acesta este mult mai complicat, dar elimină posibilitatea apariţiei fenomenului de acumulare a erorilor şi a eşuării complete a procesului de raţionament. Dezavantajul este mai ales din cauza funcţionării mai lente, şi nu atât din cauza complicării implementării. Practic, fiecare modul va apela la *aceiaşi* schema de translaţie, directă şi inversă, dar cu vocabularul aferent.

Problema cea mai importantă care apare în cazul comunicării la nivel vectorial este identificarea modulelor potrivite care pot accepta la intrare vectorii respectivi. Schema de funcţionare pe care mi-am propus-o se bazează pe ideea că propoziţia nou creată va fi acceptată de unul sau mai multe module care au un vocabular care include toate cuvintele din propoziţie. Acest lucru nu este cunoscut numai dacă se descifrează vectorul care reprezintă propoziţia (şi timpul necesar pentru această operaţie este cu un ordin de mărime mai mare decât timpul necesar pentru operaţia inversă). Aceasta a fost prima metodă pe care am aplicat-o, furnizând vectorul iniţial modulelor care puteau să accepte propoziţia, fără să mai translatez graful conceptual refăcut într-un nou vector din care am eliminat practic micile erori introduse de procesarea conexionistă. Numai că aplicând această idee, am observat acumularea unor erori, până la momentul în care translaţia din vectorii de ieşire nu se mai putea produce şi sistemul se bloca. Abia atunci am definit principiul divergenţei.

Ca şi concluzie putem spune că avem o colecţie de module, care pot fi privite ca nişte subsisteme de sine stătătoare, de acelaşi tip ca şi întregul sistem prezentat în capitolul 4. Fiecare modul este format dintr-un translator din graf in vector şi un translator invers, din vector in graf, si o reţea neuronală recurentă, care primeşte ca intrare vectori şi generează tot vectori (prin perturbare dintr-un punct stabil şi relaxare în alt punct stabil). Comunicarea se realizează prin intermediul unei structuri de date de tip blackboard (BB) care poate conţine numai grafuri conceptuale, care sunt menţinute în mai multe şiruri de aşteptare aferente fiecărui modul. Trebuie să mai existe şi un subsistem de administrare a BB, care să aleagă şirul (sau şirurile) potrivit pentru fiecare propoziţie (graf generat) şi să stabilească ordinea în şirurile de aşteptare.

Înainte de a face primele experimente, mi-am dat seama că ideea de a folosi cuvinte individuale pentru un modul era restrictivă. Este destul de simplu de observat (deja din stadiul experimentelor facute cu creionul pe hartie, ca o pre-simulare a sistemului) că aceste cuvinte individuale pot crea probleme în funcţionarea sistemului. Ideea principală a sistemului propus mai sus este ca raţionamentul să fie efectuat mai ales prin comunicarea mai multor module care pot efectua paşi limitaţi de raţionament. Existenţa unor cuvinte individuale, nu va permite transmiterea informaţiei la alte

module, din simplul motiv că acestea nu pot accepta propoziţii cu cuvinte pe care nu le cunosc. În mod practic, ceea ce se întâmplă este că aceleaşi module care încep procesul de raţionament îl şi continuă pentru că numai ele cunosc vocabularul cumulat al propoziţiilor care sunt generate. Restul modulelor ar putea să fie folosite în raţionament, doar dacă se întâmplă ca o propoziţie generată să fie compusă din cuvinte care nu sunt individuale acelui modul, dar sunt cunoscute şi de alte module, care vor putea accepta respectiva propoziţie.

Această problemă rămâne şi pentru cazul că nu mai folosim cuvinte individuale, dar gradul de acoperire al vocabularelor modulelor este mic. Cu cât vom creşte gradul de acoperire, cu atât va creşte şi rata de implicare a modulelor în raţionament. La limită, în cazul (nedorit) în care toate modulele au acelaşi vocabular (adică ele cunosc toate cuvintele din sistem), toate modulele vor fi implicate în fiecare pas de raţionament, fapt care nu este productiv şi măreşte dimensiunea reţelelor constituente ale modulelor. La cealaltă extremă avem situaţia în care un singur modul efectuează toţi paşii de raţionament, restul ramânând nefolosiţi. Se poate observa că problema alegerii vocabularului pentru fiecare modul şi a gradului de acoperire al vocabularelor între module este foarte importantă în funcţionarea sistemului.

5.3. Alegerea vocabularelor modulelor, experimente

5.3.1. Scopul unui astfel de sistem

Dacă scopul sistemului monomodular era demonstrarea generalizării unor texte şi a capacităţii reţelelor recurente de a fi folosite în astfel de aplicaţii, scopul pe care mi l-am propus pentru utilizarea sistemului multimodular a fost realizarea unor raţionamente automate. Practic, un astfel de sistem expert are acelaşi rol ca şi un sistem expert implementat prin metode pur simbolice, şi din punct de vedere al comportamentului văzut din exterior, sistemul este acelaşi cu un sistem pur simbolic. Nu am dorit să realizez un sistem care să funcţioneze mai bine sau să aibă capacităţi pe care un sistem simbolic nu le are. Dorinţa mea a fost să realizez un sistem care să funcţioneze la fel ca şi un sistem expert, să demonstrez că se pot implementa sisteme bazate pe reţele neuronale recurente şi să explorez problemele care apar la dezvoltarea şi implementarea sistemului şi eventual să identific avantaje pe care sistemele simbolice nu le au.

Este evident că un astfel de sistem este extrem de costisitor în comparaţie cu sistemul simbolic care face acelaşi lucru. Necesarul de memorie este cu două ordine de mărime mai mare (la acelaşi volum de cunoştinţe), iar viteza de lucru este cu un ordin de mărime mai mic. Deşi există posibilitatea de a realiza aceleaşi funcţii (raţionamente) şi într-un caz şi în celălat, pentru sistemul multimodular conexionst, avem nevoie de resurse de calcul care nu sunt justificate. În mod normal, un sistem de raţionament automat care are aceste capabilităţi, va fi întotdeauna implementat prin metodele "clasice". Singurul avantaj al unei implementări conexioniste este de a arăta că o versiune bazată pe reţele recurente care comunică la nivel simbolic este

funcţională.

Aceste dezavantaje care ţin de intensitatea computaţională a sistemului conexionist, pot fi evitate prin utilizarea unui hardware adecvat (pentru că implementarea se bazează pe ideea modularităţii extensibile) şi anume a unor procesoare neuronale [Ienne97] care pot implementa reţele recurente şi pot comunica prin intermediul unui procesor obişnuit, care ar realiza secvenţierea operaţiunilor de raţionament. Dar marele dezavantaj rezidă în dezvoltarea sistemului. Dacă în cazul sistemului pur simbolic, extensia cunoştinţelor s-ar face doar prin simpla adăugare la baza de cunoştinţe a unor noi propoziţii, în cazul sistemului conexionist multimodular, adaugarea unui nou set de propoziţii implică adaugarea unor noi module în sistem. Aceasta implică antrenarea reţelelor recurente şi acest lucru, făcut de fiecare dată, este extrem de costisitor. Repet însă, demonstrarea, chiar şi experimentală, a funcţionalităţii unui sistem expert neurodinamic este un succes în sine, pentru că nimeni nu realizat aşa ceva până la nivelul anului 1997.

5.3.2. Stabilirea vocabularului

Am construit vocabularul plecând de la un sistem de raţionament automat bazat pe reguli, funcţionând la nivel predicativ, şi care folosea modus ponens pentru găsirea unor propoziţii adevărate, plecând de la un set de propoziţii de start. Propoziţile puteau fi exprimate fie ca şi clauze Horn, fie ca şi grafuri conceptuale foarte simple. Toate cuvintele care apăreau în baza de cunoştinţe a acestui sistem expert, au format vocabularul viitorului sistem conexionist. Cuvintele din vocabular au fost împărţite ca şi în cazul sistemului monomodular în trei categorii: relaţii (predicate), individualităţi (constante) şi clase (variabile). Acest lucru este necesar pentru a aplica formulele de translatare tensorială prezentate în capitolele anterioare.

Nu am folosit cuvinte din limbajul natural, ci am folosit simboluri fără semnificaţie semantică, pentru a avea un mai bun control asupra proceselor de raţionament şi a nu fi influenţat în primă fază de sensul propoziţiilor generate în sistemul conexionist. Pentru că vocabularul era destul de mare la nivelul întregului sistem, l-am distribuit în aşa fel, încât toate modulele să "cunoască" un set minim de cuvinte comune (cele care sunt cele mai des întâlnite) din toate cele trei categorii de cuvinte posibile iar anumite cuvinte să apara la un număr mai mic de module (la jumatate, la un sfert, ş.a.m.d.). Nu am folosit cuvinte individuale, chiar şi cele mai rar folosite cuvinte fiind cunoscute de cel puţin două module. Un alt motiv pentru care am evitat cuvinte explicite din limbajul natural, cu o semantică bine definită, a fost acelaşi pe care l-am mai invocat în capitolul 4 şi anume imposibilitatea definirii unui univers al discursului cu un vocabular relativ mic (deşi în cazul de faţă, vocabularul a fost mai mare).

Pentru a împărţi cuvintele care nu sunt general cunoscute de toate modulele (să le numim cuvinte individuale) am încercat două tehnici: una bazată pe structura propoziţiilor din sistem (în acest caz cel care dezvoltă sistemul trebuie sa aleagă el

modulul potrivit pentru fiecare cuvânt) sau metoda pur aleatoare. Metoda pur aleatoare, nu este însă potrivită, pentru că în faza ulterioară de antrenare a reţelelor recurente, se poate întampla ca o propoziţie care este necesară pentru un anumit modul, să nu fie posibil de reprezentat pentru modulul respectiv, pentru că un cuvânt din această propoziţie nu aparţine vocabularului aferent modulului. Singura metodă sigură este identificare propoziţiilor de intrare şi de ieşire pentru fiecare modul şi stabilirea vocabularului care formează propoziţiile necesare antrenării unui modul.

Dar cum stabilim care sunt aceste propoziţii? Aici am identificat cea mai mare slăbiciune a unui astfel de sistem, şi anume că *trebuie prefigurate liniile posibile de raţionament* şi pe baza acestora, făcute seturi de propoziţii care vor conduce la configuraţia modulelor. Din păcate, această metodă este bazată pe prezumţia că noi trebuie să cunoaştem toate cazurile în care sistemul va ajunge de la premise la concluziile finale. Într-un sistem clasic, nu trebuie decât să adaugăm noi propoziţii în baza de cunoştinţe, iar in funcţie de acestea, putem atinge diverse concluzii finale. În cazul sistemului multimodular conexionist, din cauza necesităţii distribuirii vocabularului, trebuie să cunoaştem apriori modul în care raţionamentul va fi facut în timpul utilizării sistemului.

Pentru că într-un sistem bazat pe modus ponens, propoziţiile pot fi grupate pe nivele într-un graf (de tip arbore) de raţionament, am împărţit propoziţiile care apar în raţionamentele sistemului în funcţie de poziţia lor în acest graf. Un exemplu posibil de mod de construire a sistemului este dat în continuare. Considerăm următoarea colecţie de cunoştinţe:

if s(**A**, α(**B**)) and t(**D**, α(**B**)) then r(**A**, α(**B**)).
if s(**A**, α(**C**)) and t(**D**, α(**C**)) then r(**A**, α(**C**)).
if t(**D**, α(**B**)) and u(**D**, β(**E**)) then r(**D**, β(**E**)).
if t(**D**, α(**C**)) and u(**D**, β(**E**)) then r(**D**, β(**E**)).
if t(**D**, α(**B**)) and u(**D**, β(**F**)) then r(**D**, β(**F**)).
if t(**D**, α(**C**)) and u(**D**, β(**F**)) then r(**D**, β(**F**)).
if u(**D**, β(**E**)) and v(**G**, γ(**H**)) then r(**G**, γ(**H**)).
if u(**D**, β(**F**)) and v(**G**, γ(**H**)) then r(**G**, γ(**H**)).
if u(**D**, β(**E**)) and v(**G**, γ(**I**)) then r(**G**, γ(**I**)).
if u(**D**, β(**F**)) and v(**G**, γ(**I**)) then r(**G**, γ(**I**)).
if w(**A**, **B**) and x(**A**, **C**) then s(**A**, α(**C**)).
if w(**A**, **B**) and x(**D**, **B**) then s(**A**, α(**B**)).
if x(**A**, **C**) and y(**D**, **B**) then t(**D**, α(**B**)).

if x(**A**, **C**) and y(**D**, **E**) then t(**D**, α(**C**)).

if x(**D**, **B**) and y(**D**, **B**) then t(**D**, α(**B**)).

if x(**D**, **B**) and y(**D**, **E**) then t(**D**, α(**C**)).

if y(**D**, **B**) and z(**D**, **F**) then u(**D**, β(**F**)).

if y(**D**, **E**) and z(**D**, **E**) then u(**D**, β(**E**)).

if y(**D**, **B**) and z(**D**, **F**) then u(**D**, β(**F**)).

if y(**D**, **E**) and z(**D**, **E**) then u(**D**, β(**E**)).

if z(**D**, **F**) and w(**G**, **I**) then v(**G**, γ(**I**)).

if z(**G**, **H**) and w(**G**, **I**) then v(**G**, γ(**H**)).

Se poate observa că aceste propoziţii sunt formate cu ajutorul unui vocabular de 21 de cuvinte, împărţite în felul următor:

9 relaţii: **r, s, t, u, v, w, x, y, z**
3 clase: α, β, γ
9 individualităţi: **A, B, C, D, E, F, G, H, I**

Putem figura acest sistem expert bazat pe modus ponens ca pe un graf cu 12 noduri (vezi figura 5-3), în care nodurile 8, 9, 10, 11, 12 sunt noduri de intrare, iar 1, 2, 3 sunt noduri de ieşire. Putem scrie că nodurile au următoarele propoziţii asociate:

1. r(A, α(B or C))
2. r(D, β(E or F))
3. r(G, γ(H or I))

4. s(A, α(B or C))
5. t(A, α(B or C))
6. u(D, β(E or F))
7. v(G, γ(H or I))

8. w (A, B)
9. x (A, C) or x(D, B)
10. y(D, B) or y(D, E)
11. z(D, F) or z(G, H)
12. w(G, I)

Vom aloca noduri la module conexioniste, se poate observa cum graful a fost partiţionat în patru zone (I, II, III şi IV), corespunzătoare funcţionalităţii celor patru module care vor implementa sistemul multimodular. Fiecare modul va fi antrenat să accepte propoziţiile posibile din intrare, generând conform regulilor sistemului expert aferente zonei respective de graf, propoziţiile corespunzătoare de ieşire. Din punct de vedere al vocabularului se poate observa că avem următoarea distribuţie:

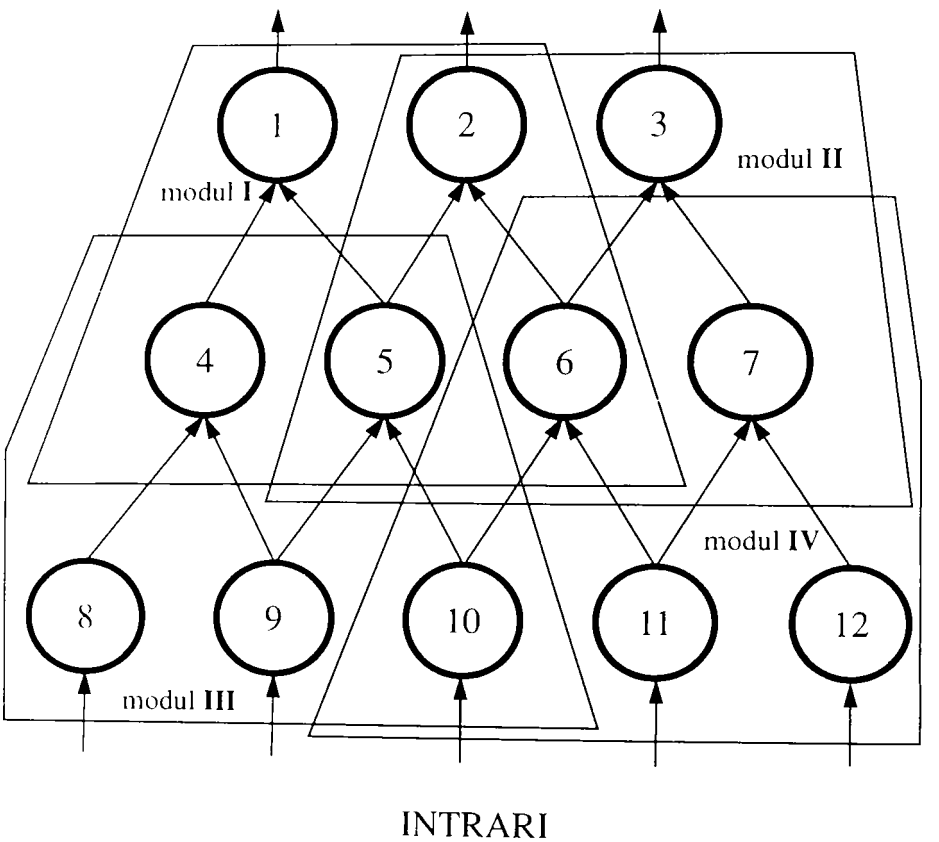


figura 4-3.

nr. modul	relaţii	roluri	individualităţi	total cuvinte/modul
I	r,s,t,u	α, β	A,B,C,D,E,F	**12**
II	r,t,u,v	α, β, γ	B,C,D,E,F,G,H,I	**15**
III	s,t,w,x,y	α	A,B,C,D,E	**11**
IV	u,v,w,y,z	β, γ	B,D,E,F,G,H,I	**14**

Putem optimiza numărul de cuvinte/modul, pentru că acesta ne va da în final numărul de unităţi al reţelei neuronale, astfel încât numărul de cuvinte/modul să fie cât mai mic în raport cu numărul total de cuvinte din sistem. Aici, numărul de cuvinte a scăzut de la 21 la 15 (pentru cel mai mare modul), ceea ce pare o scădere destul de mică, dar trebuie ţinut cont că dimensiunea reţelei neuronale este dată de formula de translatare tensorială. Pentru cazul cel mai avantajos, cel al modulului III, vom avea un vector de lungime 125. Pentru cazul cel mai dezavantajos (II) vom avea un vector de lungime 1252 (destul de mult, luând în consideraţie rezultatele prezentate în capitolul 4.) Bineînţeles, numai în cazul în care folosim o reprezentare locală pentru clase, relaţii şi individialităţi (fillers). Am arătat însă în capitolul anterior că folosirea reţelelor suplimentare FFI şi FFO reduce la cât dorim noi dimensiunea reţelei recurente (în limita scopului propus - cel de a putea învăţa tiparele necesare).

Se poate observa că alegerea acestor vocabulare este un proces deosebit de laborios şi care este destul de dificil de a fi făcut în mod automat. Pentru fiecare mecanism de raţionament (principiul rezoluţiei, maşină abstractă Warren, abducţie, etc) trebuie ca acest proces de distribuire al vocabularului general să fie făcut de către proiectantul sistemului, care în plus, trebuie să identifice toate perechile de propoziţii care sunt intrări/ieşiri ale unui modul, astfel încât să pregătească modulele pentru antrenarea prin algoritmul de backpropagation recurent.

5.3.3. Experimentele efectuate

Din cauza faptului că ar fi fost o investiţie foarte mare de timp să realizez un sistem cu propoziţii cu semantică bine definită, am ales o cale de experimentare mai simplă, deşi mai puţin spectaculoasă, fără a asocia simbolurilor folosite o anumită semantică (cuvinte din limbajul natural). Şi calea pe care am urmat-o pentru realizarea modulelor a fost diferită de cea prezentată in subcapitolul anterior, pentru că am plecat în primul pas de la vocabular şi nu de la reguli. Apoi am construit regulile (fiind constrâns de vocabularul strict definit) în aşa fel încât modulele care rezultau să aibă cam acelaşi număr de cuvinte şi aceiaşi distribuţie a claselor, relaţiilor şi individualităţilor. Chiar am forţat acest lucru, astfel încât în final am obţinut ca toate modulele să aibă acelaşi număr de cuvinte în vocabularul propriu şi aceiaşi distribuţie pe categorii de cuvinte. Mecanismul de raţionament simbolic pe care am incercat sa-l simulez cu modulele conexioniste era cel de modus-ponens, acesta fiind după părerea

mea foarte potrivit pentru implementarea neuronală. Vocabularul de la care am plecat a fost următorul:

F = { A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, A1, B1, C1, D1, E1, F1, G1, H1, I1, J1, K1, L1, M1, N1, O1, P1 }
card(F) = 32

R = { α,β,γ,δ,ε,ζ,η,θ,ι,κ,λ,μ,ν,ξ }
card(R) = 14

L = { q, r, s, t, u, w, x, y, z, q1, r1, s1, t1, u1, w1, x1, y1, z1 }
card(L) = 18

adică un total de 64 de cuvinte. Distribuţia pe individualităţi (32), clase (14) şi relaţii-predicate (18), este conformă cu distribuţia acestor categorii sintactice în propoziţii de tip "if-then" folosite uzual în limba engleză [Towell91].

Împărţind vocabularul pe module, am ales o distribuţie uniformă, în care toate modulele aveau acelaşi număr de cuvinte şi acelaşi număr de cuvinte în fiecare categorie. În urma mai multor încercări de distribuire cea mai echilibrată (din punct de vedere al resurselor necesare) a reieşit cea care avea 9 individualităţi, 5 clase şi 5 predicate. O restricţie majoră poate să apară în cazul în care ataşam individualităţile la clase şi construim o ierarhie de clase din mulţimea R (o latice). În acest caz, individualităţile care sunt ataşate unui modul, trebuie ca în mod necesar să poată aparţine doare claselor ataşate acelui modul. Dar din moment ce nu lucram la nivel semantic, acest lucru nu a creat nici o problemă, pentru că puteam să ataşez din start oricare individualitate oricărei clase.

Construirea unei ierarhii de clase a simplificat problema codificării binare a cuvintelor folosite. Pentru că cele mai multe cuvinte erau din categoria individualităţilor, am optat pentru codificarea acestora la o schemă left-right, care a redus lungimea vectorului rezultat la 5. Aplicând formula tensorială dezvoltată în capitolul 4, lungimea unui vector de reprezentare a unei propoziţii de tip intrare/ieşire este de numai 125, pentru că într-o astfel de propoziţie poate fi doar un singur predicat, iar produsele tensoriale clasa/individualitate se suprapun prin adunare. Se poate observa o reducere substanţială a dimensiunii vectorilor cu care lucram, de la 12634 în cazul prezentat în capitolul 4, la 125 (doua ordine de mărime). Am ales să folosesc un număr total de 250 de unităţi_recurente/modul, cu un număr de intrări şi ieşiri egal cu 125. Acelaşi număr de intrări şi ieşiri l-am folosit pentru reţelele FFI şi FFO (pentru fiecare modul - a rezultat o structură fixă de 125-125-125 unităţi). Rezulta un necesar de memorie numai pentru reţeaua recurentă de:

250[ieşiri] x (**250**[intrări recurente] + **125**[intrări]) = **93750**[ponderi]

adică un necesar de memorie pentru fiecare modul de aproximativ de 1MByte (dat fiind şi un surplus nesemnificativ pentru FFI şi FFO).

Numărul de module este astfel resticţionat de memoria ocupată de un modul. Pentru a nu fi blocat de această restricţie şi de memoria limitată a sistemului folosit de mine (8MBytes) am decis ca numai un singur modul să fie la un moment dat activ în memorie, restul de module (mai exact, matricile de ponderi) să fie menţinute pe disc (unde dispuneam de în medie de 130 până la 300MBytes) iar atunci când era nevoie de un anumit modul, să încarc ponderile acestuia de pe disc. Evident că acest stil de lucru este foarte lent, dar nu are nici o importanţă daca un proces de raţionament durează câteva minute. Numarul de maxim de module care puteau fi folosite într-o astfel de configuraţie era în jur de 100, fără ca să fiu în pericol să nu am spaţiu disponibil. La limită, în condiţii ideale, puteam să folosesc chiar şi 300 de module.

Configuraţia finală a modulelor a fost stabilită în felul următor: am împărţit vocabularul de 64 de cuvinte, în mulţimi de 19 cuvinte, conform schemei prezentate, astfel încât să asigur un grad de acoperire cât mai mare, şi să am un număr cât mai mai mare de module. În final, am obţinut un număr de 19 module, pe care le-am grupat într-o structură ca în figura 5-4. Suprapunerea trapezelor care reprezintă module în figura de mai sus, indică şi suprapunerea vocabularelor. Fiecare modul poate primi o propoziţie ca şi intrare, şi poate genera la ieşire o propoziţie prin simularea inferenţei prin modus-ponens. Relativ la schema de funcţionare prezentată în subcapitolul dedicat comunicării între module (vezi 5.1.2) aici apare o problemă legată de conectorii logici. În exemplul dat în subcapitolul anterior, se putea observa că la intrarea unui modul puteau apărea mai multe propoziţii deodată, iar acest lucru era dat de folosirea conectivei "and" (conjuncţie logică) care apărea în regulile "if-then" ale bazei de cunoştinţe iniţiale. În acest caz, ar fi fost necesară superpoziţia celor doi vectori care reprezentau cele două propoziţii conectate prin "and". Pentru a nu complica schema de funcţionare a sistemului multimodular, am preferat să folosesc schema de secvenţire a propoziţiilor rezultate şi să admit la intrarea unui modul doar o singură propoziţie la un moment dat. Acest lucru face ca să dispară conectivele din graful de decizie al unui sistem pur simbolic izomorf cu cel conexionist, pentru că nu putem asocia nici un fel de conective logice cu operaţiile de secvenţiere din şirurile de aşteptare de pe BB descrise în 5.2.1.

Ceea ce am dorit să fac era o verificare a funcţionalităţii, în sensul că atunci când furnizăm sistemului un set de propoziţii care sunt premise de lansare a unui raţionament, să se obţină propoziţii pe nivelul final de ieşire. Nu era important dacă rezultatul era corect din punct de vedere al logicii predicatelor de gradul 1, ci dacă pur şi simplu apărea un rezultat (o propoziţie) care fusese antrenat anterior în cadrul modulelor care asigurau ieşirea finală a sistemului. Dacă apăreau propoziţii care nu fuseseră antrenate, însemna că sistemul nu funcţiona corespunzător. Fiecare modul a fost antrenat cu câteva perechi de propoziţii generate cu ajutorul vocabularului redus al modulului, fiecare propoziţie reprezentând un fel de regulă "if-then". Pentru că reţelele recurente au o componentă temporală, era foarte importantă ordonarea tiparelor de antrenarea formate de perechile de propoziţii, astfel încât punctele în spaţiul stărilor reţelelor să fie aştezate pe o traiectorie formate din puncte apropiate. Acest lucru se putea uşor realiza, studiind vectorii asociaţi propoziţiilor (prin translaţia tensorială şi transformarea vectorului cu FFI). Se defineşte în acest caz o normă a vectorilor de

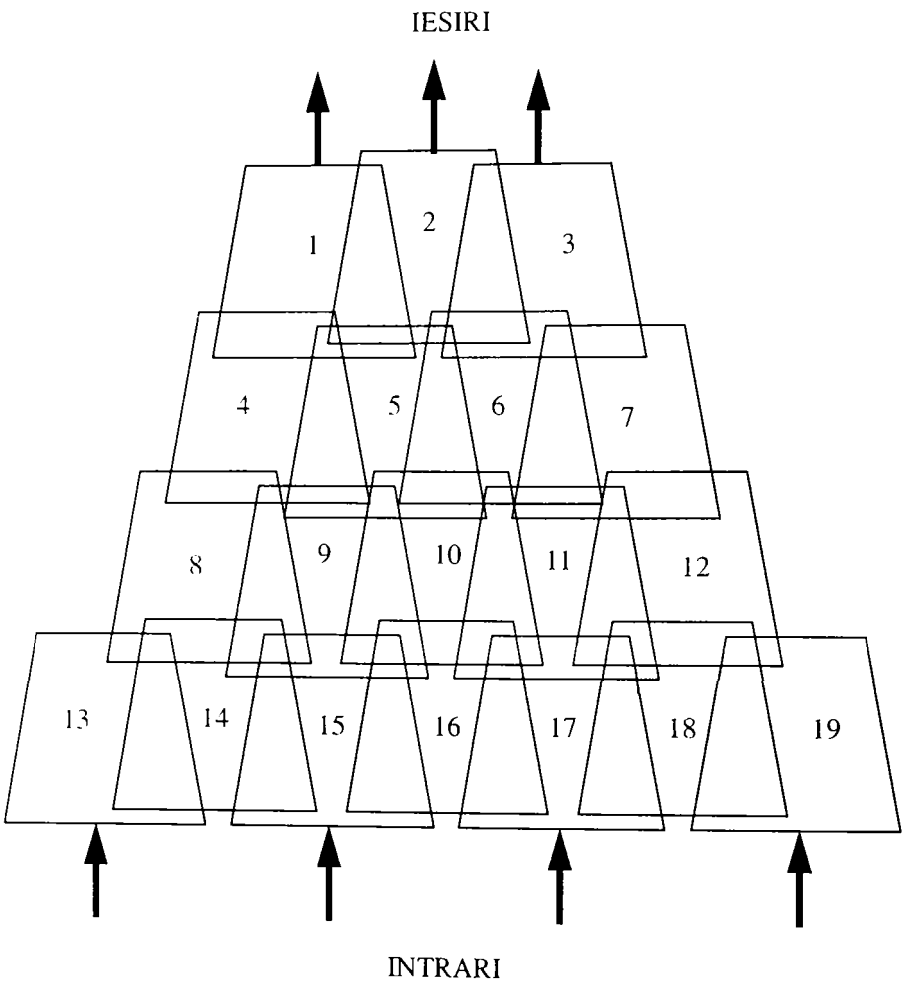


figura 5-4.

intrare/ieşire, dată de numărul de poziţii din vector setate pe valoarea 1, şi se ordonează crescător vectorii, astfel încât să formeze o traiectorie discretă de o formă cat mai puţin sinuoasă în hipercubul de dimensiune 125.

Experimentul care testa funcţionarea sistemului, a decurs în felul următor:
i. se poziţionează reţele din toate modulele (starea modulelor - un vector real 250 dimensional - este menţinută în RAM) pe un punct din spaţiul 250 dimensional dat de prima propoziţie de intrare şi prima propoziţie de ieşire de pe traiectoria descrisă de toate propoziţiile învăţate de modulul respectiv. Mai exact, se iau cele 125 de valori din vectorul de intrare şi se setează cu aceste valori cele 125 de unităţi ascunse ale reţelei recurente. La fel, se iau cele 125 de valori din vectorul de ieşire şi se setează cu aceste valori unităţile de ieşire ale reţelei. Această setare a stării tuturor reţelelor este o operaţiune deosebit de "artificială", dar nici un raţionament nu poate fi pornit dacă starea iniţială este setată aleator (sau este pusă pe 0). Practic, sistemul este pus într-o stare care deja generează un răspuns plecând de la un set de premise (care sunt propoziţiile de intrare de pe prima poziţie a traiectoriilor de antrenare a modulelor 13-19) şi ajungând la un set de concluzii finale.
ii. Se furnizează sistemului un set de premise (care în mod obligatoriu, trebuie să fie propoziţii învăţate de modulele 13-19) şi se porneşte schema de comunicare între module, prin intermediul blackboard-ului BB, până se obţin propoziţii de ieşire din modulele 1-3. În acest moment, procesarea se consideră încheiată. Se poate observa asemănarea între acest sistem şi un sistem bazat pe forward-chaining, bazat pe reguli predicative.

O serie de aspecte funcţionale neaşteptate (unele nedorite) au fost observate în decursul exprimentării acestui sistem. Primul efect l-am denumit "ciclare simplă". Pentru că există un singur blackboard care deserveşte toate modulele, propoziţiile care sunt generate de un modul, vor fi "consumate" imediat de acelaşi modul, pentru că vocabularul propoziţiilor de intrare, este identic cu vocabularul propoziţiilor de ieşire. Pentru a elimina acest fenomen, am introdus pentru fiecare modul, un pas de acceptare suplimentar, care verifica dacă propoziţia de intrare nu face parte din setul de propoziţii de ieşire învăţate. În caz afirmativ propoziţia de intrare respectivă era eliminată din şirul de aşteptare al modului respectiv. Efectul nedorit al ciclării simple era acela că starea reţelei era dusă de obicei într-un punct unde rămânea staţionară, indiferent de propoziţia pe care o primea în intrare. Alt fenomen, mai rar întâlnit, dar care ducea tot la stări staţionare nedorite ale unor module, era ciclarea prin două module. Se putea întampla în unele situaţii, ca propoziţia generată de un modul (de exemplu 10) care primea intrare o ieşire a unui alt modul (de exemplu 15) să genereze o ieşire, care din punct de vedere al vocabularului să fie acceptată din nou modulul care a generat prima ieşire (în cazul exemplului, tot 15). Evident că propoziţia nou acceptată nu face parte din setul de propoziţii de intrare învăţate, dar nici din setul de propoziţii de ieşire învăţate (conform noii reguli, ar fi ştearsă). De obicei, acceptarea unei astfel de propoziţii duce tot la o stare staţionară. Se poate imagina o schemă de evitare a acestui fenomen, dar nu am mai implementat aşa ceva, pentru că fenomenul era relativ rar.

Există un caz ideal de funcţionare a sistemului, care dă şi rezultatele dorite.

Dacă se iau propoziţiile de intrare care sunt pe a doua poziţie a traiectoriilor de antrenare ale modulelor 13-19 (după ce sistemul a fost pus în aşteptare conform pasului 1 de lucru descris anterior) şi se aplică aceste propoziţii ca şi premise de intrare, vom obţine la ieşire propoziţiile învăţate de pe a doua poziţie a traiectoriilor de ieşire a modulelor 1-3. Lasând modulele în starea în care au fost aduse de acest raţionament (fiecare a facut un pas şi am observat că nici unul nu a rămas staţionar - adică a făcut pasul dar s-a întors în starea anterioară) şi aplicând ca premise propoziţiile de intrare care urmează pe traiectoriile de antrenare ale modulelor 13-19, vom obţine la ieşire tot propoziţiile de ieşire care urmează pe cele de la procesul anterior de raţionament (pe traiectoriile de antrenare ale modulelor 1-3). Procesul continuă în acelaşi mod până se epuizează toate propoziţiile de pe traiectoriile de intrare care sunt premise.

Acest caz ideal presupune ca toate traiectoriile de antrenare să fie formate din acelaşi număr de puncte (propoziţii), iar la intrare (ca premise) să fie date tuturor modulelor de intrare (13-19) câte o propoziţie şi numai una. Se poate observa că această funcţionare este foarte restrictivă, fiind din punct de vedere al performanţelor cognitive (capacitate de inferenţă) sub nivelul unui sistem similar pur simbolic. Dar partea cea mai interesantă a funcţionării acestui sistem, cea care mi-a dat noi idei pentru continuarea aplicării reţelelor recurente în raţionamentul automat au fost experimentele care nu au dus la o funcţionare ideală, ba chiar au dus la blocare sistemului sau la obţinerea de rezultate care nu au fost prevăzute prin antrenare.

Cel mai plauzibil scenariu de funcţionare al unui astfel de sistem este că el primeşte câteva premise la intrare, care sunt aplicabile doar unui număr mic de module care au rol de a prelua premise (în experimentul meu, modulele 13-19) şi acest lucru duce în final la obţinerea unei propoziţii (sau mai multe) de la modulele care au rol de furniza ieşirea utilă sistemului (în cazul prezentat, 1-3). În cel mai bun caz, dacă numărul de propoziţii de intrare (premise iniţiale) este mai mic decât numărul de module de intrare, atunci raţionamentul va implica în continuare doar propoziţii care sunt generate de aceste module activate. Dacă aceste propoziţiie de start au fost cele de pe a doua poziţie a traiectoriilor de antrenare, raţionamentul se va desfăşura normal, până se va genera un răspuns la la modulele care asigură ieşirea. Chiar şi în cazul în care am dat o singură propoziţie ca şi premisă iniţială.

În cazul în care în procesul următor de raţionament apar premise care sunt şi ele pe poziţia a doua a traiectoriilor (şi nu au mai fost prezentate anterior sistemului) funcţionarea dorită nu mai este asigurată în continuare pentru că acestea pot să ducă la activarea unor module care au fost deja activate şi au facut un pas pe traiectoriile de antrenare. Aceste module, care sunt în "avans" faţă de propoziţiile pe care le primesc la intrare, datorită faptului că sunt activate la de către o propoziţie care este "trecută" pentru ele, vor fi module staţionare. Acest fenomen duce la o "blocare prin lipsă de informaţie" (starvation) cunoscut din sistemele multi-agent, şi la imposibilitatea obţinerii unui rezultat.

Se poate observa că pentru funcţionare normală a sistemului este nevoie de "sincronicitate", adică toate modulele să fie din punctul de vedere al traiectoriei discrete

al starii interne în acelaşi punct. Orice întârziere a unui modul, prin nefolosirea sa într-un proces de raţionament, duce la fenomenul de starvation prin atingerea de către aceste module a unor stări staţionare. Spre deosebire de acest model de sistem, un sistem pur simbolic, bazat pe reguli propoziţionale de tip "if-then" şi pe inferenţă de tip modus ponens, funcţionează şi fără ca această sincronicitate să fie necesară.

5.4. Concluzii

Deşi rezultatele nu sunt deosebit de încurajatoare, sistemul experimentat are importanţa lui practică. El demonstrează că se poate (cu o serie de restricţii - destul de serioase) realiza un raţionament automat cu ajutorul unor reţele recurente. Dacă acest lucru nu era posibil cu ajutorul unei reţele de mari dimensiuni (din considerente computaţionale), prin divizarea acestei reţele în mai multe reţele de mici dimensiuni, care necesită fiecare în jur de 1 - 4 de ore pentru antrenarea cu 4-5 perechi de propoziţii (pe o staţie obişnită), se poate implementa un astfel de mecanism de raţionament.

Modelul prezentat ar fi trebuit să genereze prin natura lui conexionistă şi distribuită, trei însuşiri care l-ar fi făcut superior unui sistem pur simbolic şi unuia bazat pe o structură cu un singur bloc. Aceste însuşiri sunt:
i. generalizarea;
ii. robusteţea la intrări perturbate (şi într-o mai mică masură generalizarea tipic conexionistă, deşi acest deziderat era mai greu de atins, dată fiind natura dual, conexionist simbolică a sistemului);
iii. plasticitatea; posibilitatea de a adăuga noi module, fără a le mai modifica pe cele vechi, şi acest lucru să ducă la o mai bună funcţionare a sistemului, în sensul ca "baza de cunoştinţe" este mai mare.

Din păcate, aceste deziderate pe care le-am căutat în dezvoltarea modelului iniţial nu au putut fi atinse. Generalizarea nu se poate realiza pentru că ar furniza propoziţii care nu au fost învăţate de către modulele dedicate pentru intrare (chiar dacă au un vocabular adecvat) şi aceasta conduce în mod inevitabil la fenomenul de modul staţionar şi la starvation.

Robusteţea ar putea fi obţinută în cazul în care am avea vectori de lungime mai mare, rezultaţi prin translaţia tensorială a unor propoziţii bazate pe un vocabular mare şi am avea situaţii în care propoziţiile ar avea mici deviaţii de la traiectoria antrenată. În acest caz, reţeaua recurentă ar fi suficient de robustă ca să nu se abată de la traiectoria învăţată. Dar în cazul nostru, vocabularul este foarte mic, şi modificarea unui cuvânt duce la modificarea destul de puternică a vectorilor de reprezentare. Lucru care duce la abaterea de pe traiectoria starii interne a reţelei suficient de mult ca să se ajungă în stări care nu mai pot fi modificate prin propoziţiile învăţate. Concluzia este în acest caz, că cu cât mai mică este reţeaua şi mai ales vocabularul aferent, cu atât modulul este mai puţin robust la intrări perturbate sau incomplete.

Plasticitatea implică cunoaşterea modulelor existente şi a vocabularelor lor. Dacă este nevoie de adăugarea de noi cunoştinţe, trebuie ca toate cuvintele adaugate în sistem să fie deja cunoscute măcar de o parte a modulelor deja existente. Adăugarea unui cuvânt necunoscut implică construirea unui modul suplimentar, ca cel puţin două module să cunoască acel cuvânt. Mai mult, modulele nou adăugate trebuie să aibă un rol foarte clar în lanţurile de raţionament, şi să respecte principiile de sincronicitate ca sistemul să poată funcţiona normal. Toate aceste restricţii fac ca adăugarea de noi module să fie un proces foarte laborios, mult mai complicat decât în cazul unui sistem pur simbolic.

Există însă pentru această arhitectură un avantaj foarte important. Resursele necesare (timp, memorie) sunt mult mai mici decât pentru o variantă echivalentă monomodulară. Dacă n ar fi numărul total de unităţi necesare, iar k gradul complexităţii polinomiale care dă necesarul pentru o anumită resursă, este evident că dacă:

$$\sum_{i=1}^{m} c_i = n \qquad (5\text{-}1)$$

(m este numărul de module, c_i numărul de unităţi pentru fiecare modul) atunci:

$$\sum_{i=1}^{m} (c_i)^k < n^k \qquad (5\text{-}2)$$

Am mai remarcat încă un fenomen: **cu cât modulele sunt mai echilibrate ca număr de unităţi, cu atât resursele necesare sunt mai mici**. Dacă, la limită, $c_i=c\ (i=1,m)$, acest fenomen este susţinut de relaţia (demonstraţia ei este imediată):

$$\sum_{i=1}^{m} c^k \le \sum_{i=1}^{m} (c+x_i)^k \qquad (5\text{-}3)$$

unde $x_i \in Z$, $|x_i| < c$ şi:

$$\sum_{i=1}^{m} x_i = 0 \quad ; \quad m \cdot c = n \qquad (5\text{-}4)$$

Aceste observaţii au influenţat modul în care am partajat modulele în modelul prezentat în continuare.

6. Modelul multimodular eterogen

Cea mai importantă concluzie preliminară la care am ajuns după experimentele şi analiza modelului multimodular a fost că un sistem de raţionament automat complex (cu un vocabular şi o complexitate a informaţiei mari) nu poate fi implementat cu ajutorul instrumentelor conexioniste actuale, utilizând mai ales reţele recurente. Motivul nu este neapărat necesarul de resurse computaţionale (şi aceasta este o problemă, dar nu insurmontabilă). O reţea neuronală, sau un grup care cooperează pentru rezolvarea unei probleme au nevoie de *supervizare*, pentru control şi deblocare.

În sprijinul afirmaţiei că procesarea pur neuronală nu este o soluţie practică, putem face o analogie cu modelul uman de rezolvare a problemelor. Dacă problema este simplă, iar omul care o rezolvă are experienţă în domeniu, răspunsul "îi vine" imediat în "minte", prin analogie cu alte probleme mai mult sau mai puţin asemănătoare. Dacă problema este dificilă şi presupune utilizarea unui formalism şi a unor paşi concreţi de rezolvare, aflaţi într-o secvenţă suficient de lungă, omul se vede în situaţia de a folosi o reprezentare grafică exterioară. El manipulează simboluri, aplicând reguli stricte de combinare / procesare / extragere. Doar la un nivel al paşilor intermediari între diversele operaţii, intervin decizii "imediate", rezultate din experienţă şi "intuiţie". Este adevărat că în mintea umană, formalismele legate de reprezentările simbolice sunt memorate şi utilizate la nivel pur neuronal, dar analizând această schemă de lucru, putem spune că într-un proces de raţionament, efectuat pentru rezolvarea unei probleme, exprimarea şi procesarea simbolică, nu numai că sunt necesare, dar ca şi importanţă, ele se află la un nivel superior celui analogic-neuronal. Mai ales în sensul că ele sunt cele care "conduc" raţionamentul.

La nivelul tehnicii şi ingineriei actuale, din punct de vedere practic, realizarea unui sistem inteligent pur neuronal este inutilă şi extrem de costisitoare. Majoritatea task-urilor intelectuale care presupun cunoştinţe aprofundate şi un volum mare de lucru, pot fi rezolvate prin programarea obişnuită, pe suportul computaţional existent (fireşte există şi excepţii - generarea discursului de exemplu). Însa toate aceste task-uri

presupun fiecare, realizarea unor operaţii care sunt foarte asemănătoare (omogenitate). Problema apare când taskul propus pentru rezolvare presupune apelarea la operaţii care sunt foarte diferite, sau reprezentarea informaţiei este extrem de eterogenă (sau cu accente de incertitudine). În astfel de cazuri, se foloseşte strategia "divide et impera", task-ul fiind împărţit în mai multe sub-task-uri specializate, care sunt rezolvate într-o anumită ordine, utilizând la nevoie, arhitecturi computaţionale foarte diferite. Supervizarea realizării task-urilor este o problemă care presupune "inteligenţă", pentru că agentul (uman sau automat) care se desfăşoară o astfel de activitate, trebuie să aibă cunoştinţe extrem de eterogene.

6.1. Modelul compoziţional aplicat în inteligenţa artificială

În domeniul inteligenţei artificiale clasice, răspunsul la problema eterogenităţii task-urilor a fost **modelul compoziţional** [Chang&Keisler73] [Cohen83&85] [Treur88]. El este aplicat şi în alte domenii, cum ar fi sistemele de operare distribuite sau bazele de date. După cercetări şi experimentări care implicau sisteme uriaşe (vezi ca exemple [Giunchiglia&Weyhrauch88] sau [Vilain&McAllester89]), înglobând un volum foarte mare de cunoştinţe, asupra cărora se aplicau mecanisme foarte complicate de inferenţă, s-a observat ca o specializare şi modularizare în cadrul acestor sisteme este necesară. Cunoştinţele au fost grupate pe diverse grade de similaritate, iar mecanismele de inferenţă au fost simplificate, ele fiind aplicabile în acest caz asupra unei singure clase de cunoştinţe. Practic, un sistem mare, a fost împărţit în mai multe module mici, mai uşor de construit, administrat şi adăugat. Noi module, conţinând cunoştinţe într-un format nou, pot fi adăugate la sistem. Din acest motiv modelul se numeşte compoziţional.

Ideea sistemului multimodular omogen, mi-a venit după încercările nereuşite cu sistemul monomodular, pentru că la Vrije Universiteit exista un framework pentru sisteme compoziţionale, numit DESIRE (**DE**sign and Specification of **I**nteracting **RE**asoning modules) [Treur90] [Treur92]. Prima idee pe care am împrumutat-o din acest framework a fost aceea de a comunica prin blackboard, într-un limbaj universal (grafuri conceptuale, în cazul meu) cunoscut de toate modulele. Spre deosebire de modelul meu, DESIRE este eterogen, modulele sale nefiind identice ca funcţionalitate interioară. Doar limbajul de comunicare (un gen de propoziţii asemănătoare logicii predicatelor de gradul întâi, trivalorică şi cu operatori modali, fără cuantificatorul existenţial) era acelaşi pentru toate modulele, fiecare modul având în componenţa sa, translatoare din/în acest limbaj. De exemplu, pentru implementarea unui joc, se pot construi mai multe module care rezolvă problema "mutării" următoare (prin metode diferite), altele care supervizează "strategia" pe termen lung, astfel încât, în final, o variantă de optim (de obicei local) să fie atinsă într-un pas de joc. Stilul acesta de lucru, pe lângă avantajul simplificării prin modularizare, mai are şi avantajul execuţiei paralele, prin alocarea diverselor module la procesoare mai mult sau mai puţin specializate.

Ideea de bază a colectivului de la VU a fost implementarea unor module foarte simple, capabile să facă operaţii banale de raţionament, care să fie utile în cât mai multe posibile aplicaţii. Dezvoltând aceste aplicaţii, sistemul este lărgit treptat, module care servesc numai unor aplicaţii, urmând să devină utile şi noilor aplicaţii (dacă este cazul). Sistemul se dezvoltă gradual, devenind din ce în ce mai mare (nici un modul construit nu este abandonat), pentru rezolvarea unor probleme noi, utilizându-se cât mai mult din vechile module. Sub "acoperişul" inteligenţei artificiale, se refolosea o idee mai veche, din ingineria programării, cea a reutilizării codului. Însă aici, aplicarea acestui principiu are o semnificaţie mai subtilă. Refolosirea unui modul, înseamnă refolosirea unor cunoştinţe "dobândite" (prin programare) anterior pentru rezolvarea unei probleme diferite dar care are o parte oarecum analoagă cu noua aplicaţie.

Dezideratul "creşterii" inteligenţei unui sistem prin simpla adăugare a unor module nu a fost încă atins. În acest model, creşteri cantitative nu duc la salturi calitative. Cercetătorii de la VU cred că punctul important este protocolul de comunicare între module. Diverse tehnici de comunicare au fost folosite, evitându-se construirea unui modul de supervizare, care să preia problemele, să le dividă, să le secvenţieze şi să le aloce modulelor. Din păcate, un protocol "natural" de comunicare nu a fost încă găsit, fiind necesară pentru fiecare tip de aplicaţie, construirea unui modul dedicat de secvenţiere.

Din discuţiile săptămânale avute cu colectivul de AI de la VU (în timpul şederii mele la Amsterdam), am observat o reticienţă în a folosi module bazate pe calcule analogice (mai ales datorită tradiţiei pur simbolice a grupului respectiv). Pornind de la această observaţie şi de la concluzia că un sistem de raţionament automat **trebuie** să fie dirijat de o componentă simbolică, am încercat inserarea unor module conexioniste în acest model. Prima soluţie, era pur şi simplu adăugarea unor module care funcţionau la acelaşi nivel ca şi cele simbolice, comunicând prin blackboard (transformând grafurile conceptuale în propoziţii de gen DESIRE şi invers). Piedica principală a fost vocabularul foarte bogat al părţii simbolice a sistemului. Un alt motiv pentru care nu am folosit această metodă, a fost că tot ce puteau face modulele conexioniste recurente dezvoltate de mine, puteau face şi module simbolice cu acelaşi rol, pentru că la nivelul problemelor care puteau fi rezolvate de DESIRE nu era nevoie de generalizare şi de robusteţe.

Analizând funcţionarea unei aplicaţii de generare a discursului utilizând un sistem bazat pe framework-ul DESIRE, s-a constatat [vanLangevelde92] că problema principală care apărea la funcţionare şi care apărea şi în alte aplicaţii, era *starvation* (problemă apărută şi în cazul modelului multimodular omogen - vezi capitolul 5). Modulele nu îşi puteau continua linia de raţionament din cauza unei propoziţii lipsă, care trebuia în acest caz, furnizată pe baza contextului în care se afla sistemul, context dat de informaţia de pe blackboard şi eventual starea internă a unor module active. Interesant că pentru probleme oarecum asemănătoare, în speţă, generarea unor discursuri cu semantică şi temă aproape identică, dar cu amănunte diferite, era nevoie de *aceleaşi* propoziţii. Metoda de lucru prin care utilizatorul furniza propoziţiile

necesare continuării raţionamentului era denumita la VU "offering hints". Proiectanţii sistemului puteau face o analiză experimentală a unei anumite aplicaţii şi puteau adăuga un modul care furniza în mod automat aceste propoziţii în funcţie de context. Acest stil de lucru nu asigură însă o facilitate automată de Machine Learning. Soluţia la care m-am gândit a fost adăugarea şi antrenarea unor module conexioniste pentru "offering hints".

6.2. Schema pe doua nivele

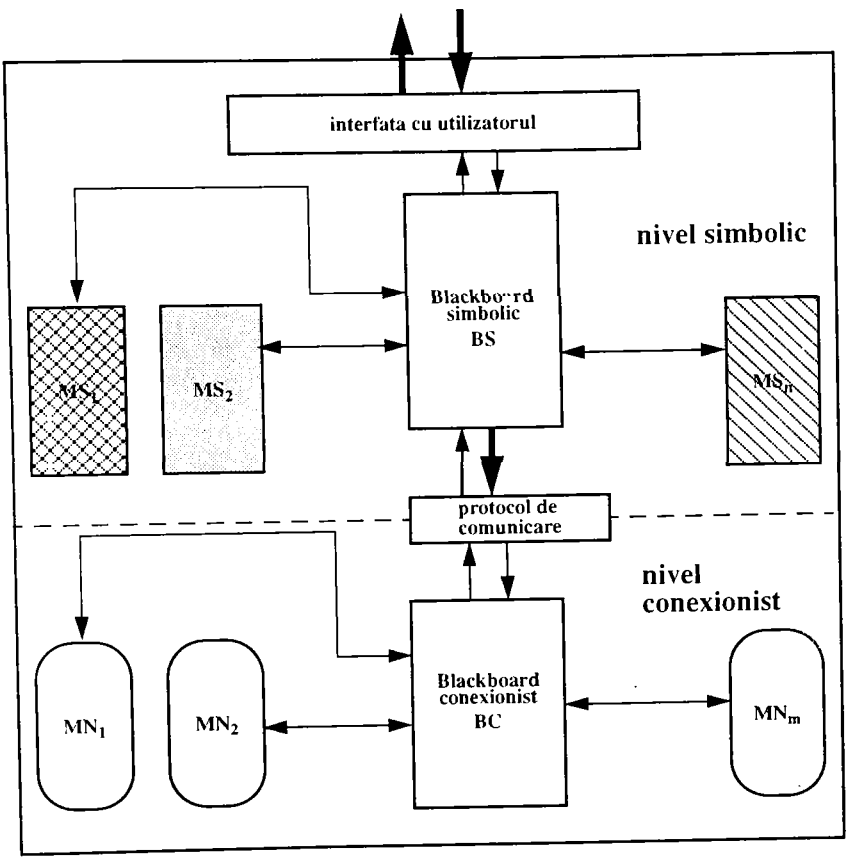
Din discuţia de mai sus, rezultă că partea simbolică a sistemului este componenta de bază, cea care preia informaţia de start, execută linia de raţionament, şi furnizează informaţia. Partea neuronală este folosită doar pentru deblocarea sistemului, atunci când acesta întâmpină evenimente de tip "starvation-points". Am considerat că cel mai potrivit mod de a realiza hibridizarea este utilizarea unei "gâtuiri" (bottleneck) a informaţiei între două nivele distincte. Un nivel va fi pur simbolic, compus din module care cooperează prin intermediul unui blackboard, şi un alt nivel compus din module conexioniste, care cooperează prin intermediul unui alt blackboard. Singura legătură între cele două nivele este asigurată prin scrierea unor informaţii de pe un blackboard pe celălalt. Schema unui sistem de acest tip este dată in figura 6-1. Analizând viabilitatea unei astfel de scheme bazate pe bottleneck, comparând-o cu alte idei care erau aplicabile în această situaţie şi cu alte metode propuse în literatură (de exemplu: [VanGelder95], [Hilario96], [Khosla&Dillon96]), am concluzionat că este cea mai plauzibilă arhitectură. Deşi nu era explicit enunţat, se poate observa şi în arhitectura sistemelor propuse acest principiu al gâtuirii informaţiei (precum şi al separaţiei funcţionale între cele două stiluri diferite de procesare).

6.2.1. Funcţionarea părţii simbolice a sistemului

Pentru că sistemul propus era unul pur experimental, menit să demonstreze utilitatea unui nivel conexionist, am folosit un număr relativ mic de module pur simbolice, toate bazate pe mecanisme de raţionament cu forward-chaining şi backward-chaining, iar limbajul de comunicare a fost redus la structuri predicative cu maximum două predicate. Am încercat două tipuri diferite de protocoale de comunicare. În principiu, aceste module simbolice funcţioneaza sincron, citind propoziţii de pe blackboard la un moment dat, şi generând în pasul următor propoziţii care se scriu pe blackboard. Trei strategii diferite de citire/scriere pot fi folosite:

i.- după ce s-au citit propoziţiile, cele care au fost citite se şterg.
ii.- după ce s-au citit propoziţiile, toate sunt şterse.
iii.- toate propoziţiile rămân.

Primele două strategii sunt utile în cazul în care modulele nu au o stare internă, care



MS_i = module simbolice (eterogene)
MN_i = module conexioniste (omogene)

figura 6-1.

se modifică dinamic, iar alocarea propoziţiilor la module se face static (doar pe baza vocabularului). În cazul în care alocarea unei propoziţii la un modul depinde şi de starea internă a acestuia (punctul unde a ajuns el în raţionamentul local pe care îl efectuează) [Treur92], atunci toate propoziţiile pot să ramână pe blackboard.

Pentru simplitate, am ales a doua strategie de citire/scriere. Aparent rudimentară, ea este însă mult mai potrivită pentru funcţionarea sistemului fără ca acesta să aibă module dedicate pentru supervizarea executării anumitor task-uri particulare. Strategiile care implică lăsarea tuturor propoziţiilor pe blackboard, implică o puternică supervizare, atât din punctul de vedere al alocării propoziţiilor (sub-task-urilor) la module cât şi din punct de vedere al secvenţierii operaţiilor (unele module trebuie să execute neapărat operaţii înaintea altora, deşi acestea pot fi "pornite", în caz contrar, apărând situaţii de deadlock).

Din punct de vedere al modului de lucru, am ales varianta nesupervizată (numită şi "naturală"), în care modulele recunosc care sunt propoziţiile de care au nevoie la un moment dat (acestea fiind prezente pe blackboard) şi doar prin simpla comunicare/citire/generare de propoziţii, task-ul general este rezolvat. La nivel teoretic, funcţionarea unui astfel de sistem este demonstrată [Treur90], în felul următor: se construieşte o bază de cunoştinţe formată din reguli de aceaşi formă (omogenă) şi care poate fi folosită într-un raţionament prin un singur mecanism de inferenţă, iar aceasta bază KB, este divizată pe module care pot executa raţionamente parţiale pe grupuri mici de reguli (kb_i). Formalizând blackboard-ul sub forma unui modul dinamic, care îşi schimbă setul de reguli in timp, şi alocând fiecărui modul o stare, care se schimbă în funcţie de starea curentă a blackboard-ului şi a operaţiilor de raţionament locale unui modul, se poate efectua cu sistemul compoziţional orice raţionament care se putea executa şi cu sistemul iniţial (monomodular - KB). Există mult mai multe variante de sisteme compoziţionale [Treur92], dar acesta este singurul care funcţionează fără modul de supervizare. Limitarea majoră a acestui sistem este fireşte omogenitatea, toate modulele fiind de fapt la fel. Singurul avantaj este uşoara mentenanţă, noi cunoştinţe putând fi adăugate prin construirea unor noi module (mici şi în special lizibile) şi prin funcţionarea eficientă (doar strictul necesar de module participă în raţionament, reducând timpul de procesare).

Problema apare atunci când adăugăm module care nu sunt identice ca şi formă de reprezentare internă a cunoştinţelor şi/sau mecanism de raţionament. Formalismul care defineşte starea internă a modulelor trebuie să fie singular (plecând de la modulul dinamic reprezentat de blackboard), şi el se bazează pe unicitatea acestor structuri per întregul sistem. Dacă două module sunt diferite nu se mai poate descrie o stare a modulelor (numai empiric). Până la nivelul anului 1996 (Decembrie) nu s-a realizat nici măcar unificarea pentru două module bazate pe backward-chaining, respectiv forward-chaining în logica bivalentă, care ar fi primul pas [Treur97]. Mai mult, eforturile s-au concentrat în direcţia cercetării unui formalism adecvat pentru modulele de supervizare (secvenţiere prin grafuri de tranziţie, alocarea nodurilor, găsirea unui model cât mai general pentru aceste module).

Adăugarea unui modul cu structură internă diferită, transformă sistemul într-un sistem eterogen. Atâta timp cât modulul nou adăugat se supune regulilor compoziţionalităţii (adică poate citi/scrie propoziţii de pe blackboard) nu există nici un impediment ca sistemul să fie viabil din punct de vedere funcţional. Experimentele cu astfel de sisteme eterogene nesupervizate, au relevat apariţia unor puncte de starvation. Se pare că problema este insuficienţa informaţiei care asigură legătura între module (adică informaţia din întregul sistem nu mai poate fi refăcută ca un set de reguli KB, care să ducă la construirea unui graf de decizie conectat). Problema poate să apară şi în cazul în care sistemul este omogen, şi se adaugă un nou modul. În acest caz însă, se poate face o verificare a faptului că graful derivat din KB-ul general este conectat sau nu. În cazul în care sistemul este eterogen, această verificare automată nu se mai poate face [Treur90]. Singura soluţie este o analiză funcţională a sistemului pe un număr cât mai mare de exemple de procesare, şi adăugarea informaţiei necesare pentru evitarea punctelor de starvation. Metoda este evident empirică, laborioasă, şi nu exclude lăsarea unor potenţiale puncte de starvation nerezolvate. Din acest motiv, sistemle compoziţionale eterogene (chiar şi cele cu module de supervizare) permit interacţiunea utilizatorului pentru furnizarea unor informaţii necesare pentru deblocarea sistemului.

Observaţie: starea internă a modulelor, într-un sistem omogen, nu este relevantă pentru utilizatorul care urmăreşte linia de raţionament. Această stare este folosită pentru alocarea propoziţiilor la module, şi pentru demostrarea teoretică a viabilităţii sistemului. În cazul sistemului eterogen fără module de supervizare, utilizatorul trebuie să vadă o "stare" a sistemului. Această pseudo-stare este de obicei dată de configuraţia blackboard-ului, si propoziţiile care au fost citite anterior.

6.2.2. Funcţionarea nivelului conexionist al sistemului eterogen

Dezavantajul principal al metodei de lucru prin "offering hints" este că sistemul nu "reţine" (învaţă) din experienţă, astfel încât, în aceleaşi situaţii, utilizatorul trebuie să ofere aceiaşi informaţie de fiecare dată. Metoda de corectare este doar studiul acestor cazuri, şi adăugarea de noi reguli unor module, sau chiar a unor noi module. Ideea mea a fost să automatizez acest proces de eliminare al punctelor de intervenţie utilizator, prin adăugarea unui nivel conexionist, compus dintr-o populaţie de module neuronale, ca cel prezentat în capitolul anterior. Am denumit noua arhitectură *modelul eterogen multimodular hibrid pe două nivele* [Szirbik97] (vezi figura 6-1). Natura eterogenă este dată de nivelul simbolic, care are în componenţă module de tipologie internă diferită, iar natura hibridă este dată de existenţa a două nivele, unul simbolic şi unul conexionist. Spre deosebire de sistemul prezentat în capitolul 5, în care modulele rezultau ca subcomponente ale unei graf de decizie iniţial, şi în care distribuirea vocabularului general trebuia facută cu foarte multă stricteţe, altfel sistemul neavând o funcţionalitate utilă, aici lucrurile sunt mai simple. Pur şi simplu, nivelul conexionist trebuie să "urmărească" linia de raţionament care se desfăşoară la nivel simbolic, şi în momentul în care apare un punct de starvation, să poată oferi informaţia

necesară.

Primul pas pentru realizarea nivelului conexionist este stabilirea configuraţiei modulelor. Aceasta se face ca şi în cazul precedent (capitolul 5) prin împărţirea vocabularului general în subseturi care se acoperă (intersecţia lor nu este vidă). Este natural ca modulele să aibă un vocabular care să poată genera propoziţiile necesare pentru rezolvarea de puncte de starvation. Se memorează aceste propoziţii, se grupează în seturi care au acelaşi vocabular, iar aceste seturi vor fi alocate fiecare unui modul conexionist. În funcţie de dimensiunea fiecărui set de cuvinte, se poate stabili dimensiunea reţelei recurente interne. Modul în care modulele sunt antrenate, va fi prezentat in subcapitolul 6.3. Ideal ar fi ca nivelul conexionist să fie compus dintr-un singur modul care să poată oferi "hints", pentru că în acest caz antrenarea ar fi mult mai simplă, iar problma dificilă a distribuţiei dicţionarului ar dispărea. Pentru că vocabularul necesar ar fi foarte mare, din considerente prezentate în capitolul 4, este imposibil de realizat un astfel de modul. Se poate argumenta că vocabularul necesar pentru "hints" este mai redus decât vocabularul întregului sistem, dar vom vedea că este nevoie de toate cuvintele care sunt cunoscute în cadrul părţii simbolice a sistemului.

Modulele conexioniste execută şi ele "paşi" de lucru, care sunt deplasări pe traiectoriile interne posibile (traiectorii discrete). Ideea de bază a funcţionării sistemului este că modulele simbolice şi modulele conexioniste execută paşi în mod *sincron*. Dacă un număr de module de pe nivelul simbolic execută un pas în linia de raţionament, modulele conexioniste (cele care sunt activate conform regulii vocabularului - vezi capitolul 5) execută şi ele un pas pe traiectoriile din spaţiul stărilor reţelelor recurente. Problemele care sunt imediat următoare sunt stabilirea unui protocol de lucru între modulele conexioniste şi stabilirea unui protocol de comunicare între cele două nivele. Deşi în principiu, schema de funcţionare a nivelului conexionist era aceiaşi ca şi la sistemul multimodular omogen, aceasta era puternic afectată de modul în care se efectua comunicarea între cele două nivele. Pentru că raţionamentul era dirijat de nivelul simbolic, nivelul conexionist fiind oarecum "auxiliar", nu mai este nevoie de o schemă foarte complicată, care să poată efectua un raţionament de sine stătător, ci este suficientă şi o schemă mai simplă, care să poată oferi informaţia necesară deblocării sistemului la momentul potrivit.

6.2.3. Posibile moduri de comunicare între cele două nivele

Principiul pe baza căruia am încercat să stabilesc un stil de comunicare între cele două nivele a fost principiul bottleneck-ului informaţional. Modulele simbolice nu pot comunica direct cu cele conexioniste (şi invers). Acest lucru are şi o motivaţie psihologică, în cursul unui raţionament, omul îşi poate concentra atenţia numai asupra unor aspecte de moment, un set restrâns de atomi de informaţie (propoziţii, formule, părţi de desen) care sunt în faţa lui, reprezentate simbolic. Mintea lui este concentrată în acel moment, doar la câteva concepte (experimentele cu subiecţi umani [Quillian&

Collins69] au relevat ca nu pot fi mai mult de 7, cu o medie de 3-5), acestea fiind în general ultimele asupra cărora şi-a concentrat atenţia. În modelul meu, comunicarea între nivele se face exclusiv între cele două blackboard-uri. Am notat blackboard-ul nivelului simbolic cu BS, iar blackboard-ul nivelului conexionist cu BC.

Cele două nivele funcţionează sincron, la un pas pe linia de raţionament simbolic îi corespunde un pas pe traiectoriile neuronale. Între aceste două evenimente există un interval, ele neavând loc simultan. Prima dată se execută pasul pe nivelul simbolic, ceea ce duce la scrierea unor propoziţii pe BS. Întotdeauna, aceste propoziţii sunt copiate pe BC. După aceasta copiere, se execută pasul la nivel conexionist, care duce la generarea unor propoziţii noi, care sunt scrise pe BC. Am identificat patru metode posibile (mai sunt şi altele, dar nu am considerat că au sens) de gestionare a informaţiei de pe BC:

I. Dacă nu există în pasul următor de pe nivelul simbolic un starvation point, atunci, înainte de citirea noilor propoziţii de pe BS pe BC, toate propoziţiile de pe BC sunt şterse. Dacă apare un punct de întrerupere la nivel simbolic, propoziţiile nou generate pe BC sunt scrise pe BS şi se încearcă un nou pas la nivel simbolic. Dacă propoziţia lipsă a fost generată corect la nivel conexionist, noul pas la nivel simbolic va reuşi şi conform cu această metodă, înainte de a scrie noile propoziţii de pe BS pe BC, BC este ştearsă în totalitate.

II. Doar propoziţiile consumate de pe BC sunt şterse (indiferent dacă este vorba de starvation point sau nu). În acest caz, pe BC pot rămâne propoziţiile neconsumate care au fost citite de pe BS, sau propoziţii care au fost generate de nivelul conexionist, toate fiind grupate în cozi de aşteptare, în modul descris în capitolul anterior. Se poate întampla ca propoziţii generate la nivelul simbolic să nu poată fi citite la nivel conexionist, din cauza vocabularului limitat al modulelor (acestea cunosc toate cuvintele, dar pot exista propoziţii care să fie formate din cuvinte care aparţin unor module diferite). Aceste propoziţii nu trebuie scrise pe BC, iar pentru aceasta am folosit un "filtru" care analiza compoziţia propoziţiei din punct de vedere al vocabularelor posibile.

III. Propoziţiile generate la nivel conexionist, sunt scrise pe BS, indiferent dacă a apărut un punct de întrerupere sau nu, înaintea pasului executat la nivel simbolic (şi sunt folosite în acest pas de raţionament, dacă apare starvation point). Conform strategiei iniţiale, după un pas de raţionament simbolic, toate propoziţiile de start sunt şterse de pe BS, înainte de a scrie noile propoziţii generate. Relativ la ce se întâmplă pe BC, putem avea două subcazuri:

IIIa. Toate propoziţiile de pe BC sunt şterse.

IIIb. Numai propoziţiile consumate (la nivel simbolic, pentru deblocare, sau la nivel conexionist) sunt şterse. Această ultimă variantă, poate fi rafinată, în sensul că doar propoziţiile consumate la nivel simbolic să fie şterse, cele consumate la nivel

conexionist, urmând să fie păstrate - în ideea că eventual, o astfel de propoziţie generată la nivel conexionist, va fi utilă în unul din paşii următori.

Aceste patru metode au fost prezentate în ordinea complexităţii date de implementarea lor. Din experimente, am constatat că cea mai eficientă metodă era prima, care este şi cel mai simplu de implementat. Asta nu înseamnă că celelalte nu sunt bune. În special IIIb, este o metodă foarte interesantă, care duce la rezolvarea unor probleme care apar aplicând metodele I şi II. Se mai poate observa că aceste metode implică gradual (de la prima la a patra) o creştere a schimbului de informaţie între cele două nivele (referindu-ne numai la direcţia BC către BS, cealaltă direcţie fiind constantă). În primul caz comunicarea este limitată doar la momentele de starvation, în al doilea caz se observă o aglomerare a informaţiei pe BC, iar în al treilea caz, fiecare pas conexionist contribuie la paşii de pe nivelul simbolic.

6.3. Antrenarea modulelor nivelului conexionist

Primul lucru care trebuie cunoscut la antrenarea nivelului conexionist este numărul de module. Acest număr poate fi constant sau variabil. Am ales varianta mai simplă (statică) în care numărul de module rămâne neschimbat. Mai mulţi factori influenţează numărul de module create. Primul este numărul total de cuvinte folosit la nivelul simbolic. Din experienţele efectuate cu sistemul multimodular omogen, am observat că numărul optim de cuvinte/modul (lucrând cu staţii de lucru obişnuite este de 15-22 cuvinte. Am concluzionat că numărul maxim de module este dat de formula empirică:

nr_module = (nr_total_cuvinte / 22) * grad_de_acoperire_global

Tot din experienţele anterioare, am constatat ca valoarea uzuală pentru *grad-de-acoperire-global* este în jur 1.5 (exprimarea matematică a gradului de acoperire global este dată în 6.4.3).

6.2.1. Vocabularul folosit la nivel conexionist

Factorul cel mai important în stabilirea numărului de module este însă numărul de starvation points, adică numărul total de propoziţii care trebuie furnizate nivelului simbolic. Acestea pot fi identificate manual, prin analiza funcţională a sistemului (compoziţional-eterogen-pur simbolic), sau automat, în timp ce sistemul este folosit de utilizator, care în momentele de blocare, oferă "hints". S-a observat (vezi [Treur90], [Treu92]), că pentru mai multe linii de raţionament asemănătoare, punctele de întrerupere apar în general în acelaşi moment (măsurat prin numărul de paşi efectuat de la începutul raţionamentului). Mai mult, propoziţiile care sunt necesare pentru deblocarea raţionamentului sunt foarte asemănătoare din punct de vedere al

vocabularului folosit. Această observaţie m-a condus la decizia de a construi un modul separat pentru fiecare grup de propoziţii asemănătoare care apar aproximativ în acelaşi moment.

Definiţie

Un Punct General de Blocare (PGB) este un moment t (variabil) din linia de raţionament a mai multor linii de raţionament $\{L_i\}_{i=1,n}$ *(n = min(m)) unde apare un punct de întrerupere datorită fenomenului de starvation. Valoarea t nu este fixă, ea fiind calculată faţă de o valoare fixă* t_f, *ca* t_f+/-*k, definindu-se astfel un interval de apariţie.*

Observaţie: Valorile m şi k se stabilesc de către proiectantul sistemului, în funcţie de dimensiunile vocabularelor modulelor care rezultă.

Am lucrat cu valorile $m = 4$ şi $k = 1$. Pentru detectarea automată a acestor PGB-uri, sistemul trebuie încercat de utilizator de un număr cât mai mare de ori, acesta oferind informaţia lipsă în momentele de blocare. Doar în cazul în care informaţia oferită este utilă (duce la continuarea şi terminarea raţionamentului), propoziţia este reţinută ca făcând parte din grupul de propoziţii legat de PGB. Acest grup poate avea un număr mai mare de propoziţii decât n şi în acest caz ar trebui împărţit în grupe mai mici de propoziţii. Împărţirea se face pe baza vocabularului, astfel încât pentru subgrupele de propoziţii să avem un număr cât mai mic de cuvinte. Pentru că am făcut această împarţire manual (deşi se poate construi un program care să facă această aplicaţie automat), nu am fost foarte riguros în păstrarea regulii ca exact patru propoziţii să facă parte dintr-un subgrup (uneori erau şi 5 sau 6, uneori doar 2 sau 3). Relativ la exemplele pe care le-am folosit, împărţirea unui grup în subgrupe s-a întâmplat destul de rar, de obicei un PGB având asociat un singur modul conexionist. Mai mult, în experimentele pe care le-am făcut, am observat ca o tendinţă că sunt puţine starvation points care să poată fi grupate conform definiţiei PGB-urilor. Pot să apară şi starvation points care să nu îndeplinească condiţiile impuse de valoarea n. Am luat decizia ca şi pentru acestea să construiesc module, chiar dacă a fost cazul de a forma un modul pentru o singură propoziţie (deşi astfel de situaţii individuale sunt uşor de rezolvat, prin intervenţia proiectantului la nivelul simbolic - printr-o simplă adăugare a unor reguli într-unul din modulele simbolice). Avantajul acestei metode de rezolvare conexionistă s-a observat mai târziu.

Pentru a putea furniza o propoziţie, un modul conexionist bazat pe o reţea recurentă, trebuie să parcurgă o traiectorie în spaţiul stărilor acestei reţele. Traiectoria este discretă, iar fiecare punct reprezintă o propoziţie translatată din spaţiul structurilor simbolice în spaţiul vectorial real N-dimensional. Se poate observa de aici, că vocabularul folosit de un modul, care primeşte la intrare propoziţii provenite de la nivelul simbolic, prin scrierea de pe BS pe BD, nu poate fi limitat doar la vocabularul propoziţiilor asociate unui PGB. *Pentru a putea urmări raţionamentul desfaşurat la nivel simbolic, este bine ca modulele conexioniste, să poată recunoaşte TOATE*

cuvintele folosite în sistem. După cum se va vedea, datorita schemei de antrenare pe care am folosit-o, metoda mea necesită acest lucru.

Ceea ce am încercat să evit la stabilirea configuraţiei modulelor conexioniste a fost o legătură directă inter-nivele, între anumite module simbolice şi anumite module conexioniste. Am fost tentat să fac la început acest lucru, pentru ca PGB-urile puteau fi asociate şi cu perechi de module simbolice (starvation points-urile apăreau cu frecvenţă ridicată "intre" două module simbolice care puteau fi identificate). Astfel s-ar fi putut crea o legătură directă între trei module, două simbolice şi unul conexionist, care s-ar fi susţinut reciproc, prin schimb de informaţie, pentru evitarea blocării. Însă folosind o astfel de comunicare, aş fi renunţat la principul de "bottleneck" informaţional şi chiar la natural compoziţională a sistemului (modulele simbolice nu ar mai fi comunicat doar prin BS, ci şi prin intermediul modulelor conexioniste). S-ar fi renunţat şi la BC, de care nu era nevoie în această schemă (modulele simbolice comunicând direct cu modulele conexionste), astfel încât ar fi dispărut comunicarea între modulele conexioniste. Această comunicare s-a dovedit a fi foarte importantă, pentru că ea a fost motivul pentru un comportament avantajos, neaşteptat în timpul experimentelor (pe larg prezentat în subcapitolul 6.4). O tentaţie asemănătoare poate fi şi asocierea intenţionată (la nivel de proiectare iniţială) a vocabularelor unor module simbolice cu vocabulare ale modulelor conexioniste, mapări ale unor cuvinte din zona simbolică cu reprezentarea lor în zona neuronală fiind întalnite în literatură (vezi [Sun92]). Această legare am considerat-o ca nefiind necesară, pentru că oricum, în timpul construirii modulelor prin schema MILD (prezentată în subcapitolele următoare), o astfel de mapare se realizează automat.

6.3.2. Traiectoriile de antrenare şi alocarea lor la module

Datorita comunicării permanente între BS şi BC (la fiecare pas, conţinutul BS este copiat pe BC), modulele conexioniste pot avea acces la informaţia care este folosită pentru efectuarea raţionamentului simbolic. Pentru a putea oferi informţie utilă la momentul oportun, ele trebuie să decidă între posibilele linii de raţionament simbolic care pot sa existe. Numărul de linii de raţionament posibile este foarte mare (dat de o formulă combinatorială în raport cu numărul de propoziţii care apar în sistem). Dar "hint"-ul corespunzător este întotdeauna asociabil cu linia de raţionament din care face parte. Dacă cunoaştem ce propoziţii s-au scris pe BS până la un anumit moment, putem predicta cu uşurinţă ce propoziţie ar urma (şi lipseşte). Ideea mea a fost de a lega linia de raţionament care duce la un starvation point cu o traiectorie neuronală care se termină într-un punct care reprezintă propoziţia căutată. Pentru că o reţea neuronală recurentă poate fi antrenată cu mai multe traiectorii diferite, un modul conexionist poate rezolva mai multe starvation points (grupate într-un PBG). Dacă o reţea neuronală primeşte ca intrare o serie de propoziţii (translatate în vectori) pe care le-a învăţat anterior, ea va evolua pe o traiectorie predictibilă, la capătul căreia putem plasa propoziţia care rezolvă punctul de întrerupere care a blocat raţionamentul simbolic.

Aceste propoziţii care formează traiectoriile de antrenare, pot fi identificate, urmărind interacţiunea utilizatorului cu sistemul simbolic. De la startul raţionamentului, toate propoziţiile care au fost plasate pe BS, sunt memorate separat pentru fiecare pas executat, şi în ordinea apariţiei lor. La un moment dat, pot să apară mai multe propoziţii pe BS, dar va trebui ca numai una per pas să fie reţinută (modalitatea de eliminare a restului de propoziţii se face pe bază de vocabular). Numai atunci când apare un punct de întrerupere, propoziţiile memorate până în acel moment vor fi folosite pentru construirea bazei de antrenare pentru reţeaua neuronală. Mai mult, punctul de întrerupere *trebuie* rezolvat cu succes de utilizator; informaţia oferită de el trebuind să ducă la finalizarea raţionamentului dorit. În acest caz favorabil, propoziţiile liniei de raţionament, de la început, până la punctul de întrerupere, trebuie memorate într-o listă ordonată pentru a fi utilizată în procesul de învăţare a reţelelor recurente. Ultima propoziţie din listă va fi tocmai cea oferită de utilizator. Am numit aceste propoziţii terminale, PHINT (phrase for hint). Întregul şir de propoziţii care duce la PHINT, l-am denumit WL-PHINT (Whole Line to PHINT).

Dacă extragem toate cuvintele din care sunt compuse propoziţiile dintr-un WL-PHINT obţinem un vocabular de o anumită mărime. Grupând vocabularele mai multor WL-PHINT-uri (prin reuniune), a căror PHINT-uri aparţin aceluiaşi PGB, obţinem de obicei un vocabular cu o dimensiune foarte mare în raport cu numărul total de cuvinte (o treime sau chiar mai mult de jumătate). Pentru construirea modulelor este important (din punct de vedere al resurselor computaţionale) ca vocabularul intern asociat reţelei recurente, să aibă dimensiuni rezonabile (20, maximum 30 de cuvinte - vezi capitolele 4 şi 5). Dacă am asocia fiecarui PGB un modul conexionist şi dimensiunea relativa a vocabularului PGB ar fi mare în raport cu numărul total de cuvinte, am restrânge vocabularul total la un număr prea mic de cuvinte (maximum 60). Reuniunea vocabularelor asociate tuturor PGB-urilor impune vocabularul total al sistemului simbolic folosit experimental. În cazurile practice, vocabularul de la nivel simbolic impune vocabularul folosit la nivel conexionist.

Pentru sisteme cu un vocabular restrâns (sisteme destul de rar întâlnite), soluţia ar fi construirea unui modul pentru fiecare PGB, fiecare WL-PHINT asociat PGB fiind transformat prin translatare tensorială într-o traiectorie de antrenare de intrare pentru reţeaua neurodinamică. Am numit această traiectorie T2P (Trajectory to PHINT). Ieşirea va fi de tip predictiv, propoziţia care se generează, este următoarea propoziţie (asociată următorului pas) de pe traiectoria de intrare. Acest mod de antrenare este cel mai natural şi avantajos pentru reţelele recurente [Williams&Zipser90]; chiar dacă există propoziţii lipsă, sau unele propoziţii sunt incomplete, starea reţelei se menţine pe traiectoria antrenată. Apar însă două probleme.

a. Adăugarea unor noi module simbolice, poate mări vocabularul total al sistemului, lucru care influenţează aproape sigur PGB-urile existente, vocabularul asociat lor fiind mărit de apariţia a noi WL-PHINT-uri. Acest lucru duce la reconstruirea din start a modulului conexionist, cu toate implicaţiile legate de mărirea matricii de ponderi şi a timpilor de antrenare. Desigur, se poate dezvolta sistemul prin

adăugarea de noi module simbolice şi fără a mări vocabularul, dar această restricţie este foarte severă. Din acest motiv nu am folosit idea de asocia fiecarui PGB un modul conexionist, ci am gândit o schemă mai flexibilă, pe care am numit-o "Module Integration and Linkage for Distribution" (MILD).

b. A doua problemă este însă mult mai importantă. O reţea recurentă de tipul celei utilizate prezintă o restricţie importantă, legată de lungimea traiectoriilor discrete care urmează să fie antrenate. **Acestea trebuie să fie formate toate din acelaşi număr de puncte**. Numărul de propoziţii din WL-PHINT-uri este diferit pentru acelaşi PGB şi trebuie ca T2P-urile să aibă acelaşi număr de puncte. Am găsit mai multe metode de a rezolva această problemă. Una ar fi executarea unor operaţii de "join" între propoziţii consecutive, astfel încât numărul final să fie mai fix. Dar nu se poate garanta întotdeauna că aceste operaţii duc la rezultatul dorit (lungimi egale). O metodă mult mai simplă este de a găsi experimental un număr fix de propoziţii pentru un modul (pe care l-am notat NP - Number of Phrases) şi a elimina primele TN_m - NP propoziţii din WL-PHINT$_m$, unde TN_m este numărul total de propoziţii din WL-PHINT-ul de indice *m*. O altă problemă care decurge din această metodă, este că uneori lungimea traiectoriilor este prea mică pentru a putea îndeplini sarcina reţelei de a predicta următoarele propoziţii de pe traiectorie. Şi pentru această problemă am găsit o rezolvare, care se îmbină foarte bine cu schema MILD.

Schema MILD am aplicat-o manual, deşi dacă s-ar defini un formalism mai riguros al ei, s-ar putea implementa şi automat (am renunţat la aceasta pentru că era destul de complicat din punct de vedere al implementării). Schema este următoarea:
- Se alege un număr (de WL-PHINT-uri) care să fie în jur de *n*, numărul minim de propoziţii PHINT pentru construirea unui PGB, şi se aloca unui modul. Vocabularul rezultat nu trebuie să depăşească o valoare maxim impusă (în cazul experimentlor mele, 30). Dacă există şi alte WL-PHINT-uri asociate aceluiaşi PGB rămase neintegrate, se grupează şi se alocă altui modul nou (care va avea un vocabular uşor diferit). Pentru PGB-uri care au număr de WL-PHINT-uri mai mic decât *n*, se va face câte un modul pentru fiecare (chiar dacă e vorba de PGB-uri cu un singur WL-PHINT), dar încercând păstrarea unui număr cât mai mic de module. Voi argumenta mai târziu, în discuţiile legate de experimentele efectuate, de ce este bine ca numărul de module rezultate prin MILD să fie cât mai mic.

După această împărţire, se stabilesc vocabularele, iar în funcţie de acestea structura şi dimensiunea reţelelor, precum şi numarul NP pentru fiecare modul şi se transformă WL-PHINT-urile in T2P-uri. Există o relaţie între numărul de T2P-uri învăţate şi lungimea lor. Relaţia este de proporţionalitate inversă, crescând lungimea şi scăzând numarul de traiectorii cu o raţie egală, timpul de antrenare este acelaşi (pentru o epocă - vezi capitolul 3). Pentru a avea un echilibru între efortul de antrenare al modulelor conexioniste rezultate, este bine ca pentru module care au multe T2P-uri de învăţat, lungimea lor să fie mai mică, iar pentru module cu puţine traiectorii, lungimea să fie mai mare. Aceasta poate duce la traiectorii prea scurte pentru modulele cu multe T2P-uri în baza de antrenare, astfel încât reţeaua să nu poată fi antrenată până

la o eroare suficient de mică ca să poată discrimina PHINT-ul corect. Soluţia ar fi împărţirea WL-PHINT-urilor alese pentru acest modul în două noi grupe, fiecare ducând la un modul mai mic (şi ca număr de traiectorii, dar şi ca vocabular) şi uşor de antrenat, chiar dacă mărim valoarea NP pentru fiecare. Dar repet, este foarte important să păstram numărul de module conexioniste cât mai mic.

6.3.3. Echilibrarea efortului de antrenare al reţelelor neuronale

O soluţie pentru asigurarea reuşitei discriminării între WL-PHINT-urile antrenate, fără a le împărţi excesiv între module este ataşarea unui modul special, pe care l-am denumit **modul de prefaţă**. Acest lucru presupune adăugarea unui modul suplimentar, dar stilul de adăugare nu implică o divizare a vocabularului iniţial al modulului "mare". Voi ilustra ideea cu un exemplu. Să presupunem că avem un modul cu un număr mare de WL-PHINT-uri asociate (n traiectorii), care au lungimea traiectoriilor de antrenare P, obţinută prin scurtarea traiectoriilor iniţiale (să presupunem că acestea aveau aproximativ 2*P). Pentru a obţinem discriminarea corectă, am putea să extindem traiectoriile cu câte un pas (P+1, P+2, ş.a.m.d.) şi să repornim antrenarea din start. Din păcate numărul de paşi suplimentari necesar este destul de mare atingând de obicei 2*P. Experimental am obsevrat că dacă avem m traiectorii de lungime P discriminate corect, şi adăugam înca m traiectorii (în total 2*m) aveam nevoie de aproximativ 2*P paşi. Având un număr dublu de traiectorii şi existând relaţia pătratică între lungimea lor şi timpul de antrenare, efortul de antrenare creşte de 8 ori. Acest calcul nu ia deloc în considerare creşterea absolut necesară a numărului de unităţi necesare pentru ca antrenarea să reuşească cu acelaşi succes ca la reţeaua cu m unităţi (creşterea capacităţii de învăţare a reţelei). Având mai multă informaţie de introdus în reţea prin antrenare, creşterea dimensiunii reţelei este absolut necesară. Timpul de antrenare este în relaţie cubică cu numărul de unităţi (vezi capitolul 3), astfel încât numai dublarea numărului de unităţi duce la mărirea cu un factor de 8 a timpului de antrenare iniţial. Deci, o dublare a numărului de traiectorii, o dublare a lungimii lor, şi o dublare a numărului de unităţi duce la o mărire cu factorul 64 a timpului de antrenare (relativ la o singură epocă).

Este evident că o astfel de soluţie nu este viabilă. Unul din scopurile mele minore în această fază a fost ca toate modulele conexioniste să poată fi antrenate aproximativ cu acelaşi necesar de resurse computaţionale (care să fie relativ mic). Chiar dacă acest deziderat nu este atins, suma totală e timpilor de antrenare este bine să fie cât mai mică. Am arătat că acest timp total este minim atunci când avem acelaşi număr de unităţi distribuite fiecărui modul (vezi subcapitolul 5-4). Altă observaţie este că mărirea numărului de traiectorii care trebuie învăţate nu duce neapărat la o creştere cu aceiaşi rată liniară a numărului de unităţi. O astfel de dependenţă există şi între lungimea discretă a traiectoriilor şi numărul de unităţi. Este normal ca în această situaţie să căutăm o soluţie care reduce lungimea traiectoriilor şi nu numărul de traiectorii. Reducerea numărului de traiectorii duce şi la reducerea vocabularului (care ar duce şi la reducerea numărului de unităţi) dar am afirmat că nu dorim acest lucru,

din motive care vor fi ilustrate ulterior. În plus, reducerea vocabularului în acest caz, poate fi destul de redusă, aproape nesemnificativă, dacă WL-PHINT-urile care se despart în două grupuri, au cam acelaşi vocabular (şi de obicei aşa este).

Dacă avem un grup de *n* traiectorii T2P (să luăm pentru claritate *n* număr par) de lungime 2*P-1, putem împărţi traiectoriile în două grupuri egale de n/2 T2P-uri. Aceasta ar fi o soluţie "paralelă". O soluţie "serială", este să împărţim toate traiectoriile în două segmente contigue, una de la momentul 1 la momentul P, şi alta de la momentul P la momentul 2*P-1. Important este ca punctul final unei traiectorii de la 1 la P să fie identic cu punctul iniţial al unei traiectorii de la P la 2*P-1. Vom obţine două grupuri de n traiectorii, care pot fi conectate spaţial două câte două prin perechile de puncte "ultimul-primul". Vom antrena o reţea recurentă cu primul grup de traiectorii cu paşi de la 1 la P şi altă reţea cu al doilea grup de traiectorii cu paşi de la P la 2*P-1. Punctele din momentele 2*P-1 reprezintă PHINT-uri, iar propoziţiile asociate momentului P le-am denumit LINK-uri. Modulul nou creat, l-am denumit, modul de prefaţă (cel de-al doilea l-am numit **principal**). Dacă un modul conexionist recunoaşte o linie de raţionament care se desfăşoară la nivel simbolic (a cărei propoziţii sunt transmise pe BC) şi este unul din modulele de prefaţă, ea va evolua pe traiectoria internă până la punctul asociat propoziţiei LINK. Această propoziţie este scrisă pe BC, şi va fi citită de modulul (conexionist) principal, iar în cazul în care evoluţia de la nivelul simbolic urmează linia WL-PHINT-ului care a dus la construirea traiectoriei urmată în prima sa jumătate de modulul de prefaţă, modulul principal va urma aceiaşi traiectorie, până la punctul asociat PHINT-ului. Aceasta este partea de "Linkage", din schema MILD, folosită exclusiv din motive computaţionale, pentru a echilibra task-urile de antrenare a reţelelor. Ca avantaj al metodei, se observă că efortul computaţional pentru antrenare este acelaşi pentru cele două reţele (şi nu implică nici o modificare a numărului de unităţi ale unei reţele pentru *n* traiectorii de lungime P).

În aceste moduri, PGB-urile prea "aglomerate", pot fi împărţite pe mai multe module, fără a renunţa la ideea de a avea cât mai puţine module conexioniste. O regulă empirică este de a utiliza metoda împărţirii WL-PHINT-urilor (asociate unui PGB) în mod "paralel" dacă PGB-ul se află la un moment relativ "de început" (în comparaţie cu liniile de raţionament cele mai lungi) sau de aplica metoda "Linkage", cu un modul suplimentar de prefaţă, dacă PGB-ul este "la sfârşit". Pentru că nu am lucrat decât cu sisteme de mică dimensiune, nu a fost necesară metoda "Linkage" desfăşurată pe trei sau mai multe module, deşi nu văd nici un impediment ca aşa ceva să nu funcţioneze la fel de bine ca şi schema cu un singur modul de prefaţă. Sunt şi PHINT-uri care nu pot fi grupate în PGB-uri; pentru acestea am construit module separate, care au fost mult mai uşor de antrenat decât modulele derivate din PGB-uri cu n traiectorii, dar numărul lor a fost în general mic şi nu a influenţat efortul de antrenare. Resursele computaţionale necesare pentru toate aceste "lightweight modules", au fost sub nivelul resurselor cerute de pildă de un modul principal împreună cu modulul lui de prefaţă.

6.4. Experimente

Scopul acestei cercetări, după cum am mai afirmat anterior, este validarea unui model care propune o metodologie diferită pentru construirea unui sistem hibrid (prin integrarea reţelelor neurodinamice). O validare teoretică a acestui tip de model este extrem de dificilă şi după unii autori chiar imposibilă [Touretzky91] [Smolensky92], dată fiind natura analogică, puternic distribuită a reţelelor neuronale, în contrast cu natura simbolică, bine determinată, locală şi unitară a sistemelor logice cu care sunt puse în contact. Un excelent exemplu de model de raţionament automat care nu are un suport demonstraţional teoretic este MYCIN [Shortliffe&Buchanan76], utilizat în foarte multe sisteme opraţionale, care au certe avantaje. Demonstraţia pe care am încercat-o pentru modelul meu, va fi utilitaristă, printr-o implementare de tip "toy system", şi care a încercat pentru început doar să ilustreze că un astfel de sistem funcţionează, iar în plus, să arate că funcţionarea nu este posibilă în lipsa nivelului simbolic (lucru discutat şi în capitolul anterior), prin ilustrarea rolului necesar al acestuia.

Ca pe un fel de demonstraţie prin inducţie, am conceput partea experimentală în două faze, similare demonstrării unei formule care are un parametru k variabil. Dacă presupunem că pentru n formula este corectă, iar pentru $k+1$ putem demonstra că ea rămâne corectă, atunci este corectă pentru orice k. Fazele experimentului meu erau:

i. Construirea unui sistem minimal, care să arate că sistemul funcţionează, după schema dorită.

ii. Extinderea sistemului (cu un "ordin de mărime") şi testarea lui în aceleaşi condiţii ca şi la pasul 1. Dacă sistemul funcţionează, putem presupune că modelul este viabil. Acest tip de abordare se numeşte în literatura dedicată sistemelor de raţionament automat cu incertitudini, "weak demonstration" [Todd91], şi este în general acceptat în comunitatea cercetătorilor în domeniul sistemelor expert cu componente analogice.

Deşi nu a fost unul din scopurile urmărite iniţial, s-a relevat experimental şi rolul util al proprietăţii de generalizare a componentelor conexioniste folosite. La alt nivel şi cu alte metode, aptitudinea de generalizare a unei reţele recurente a fost încercată şi în capitolul 4. Dacă experimentul acela nu a reuşit decât parţial, din motive de intensitate computaţională, prin reducerea complexităţii problemei de rezolvat şi adaptarea ei în cadrul unui sistem hibrid cu o puternică parte simbolică, am obţinut un rezultat asemănător pe care îl consider mai valoros decât simpla funcţionalitate a sistemului de probă. *O observaţie foarte importantă este că punerea în evidenţă a capacităţii de generalizare a întregului nivel conexionist este proporţională cu gradul de acoperire al vocabularelor modulelor conexioniste.* Pentru că am considerat fenomenul interesant am studiat această proprietate de generalizare în o a treia fază demonstrativă, care a dus la eliminarea cuvintelor individuale din aceste vocabulare şi suprapunerea lor la un nivel mai mare decât cerinţele funcţionale ale modelului.

6.4.1. Faza sistemului minimal

Pentru a obţine o simplificare esenţială a experimentelor, am renunţat (ca şi în experimentele descrise la capitolul 4) la o semantică definită prin cuvinte alese din limbajul natural. Am folosit simboluri fără o reprezentare în lumea reală sau abstractă, de tipul c1 pentru clasa 1 ş.a.m.d. Experienţa acumulată în timpul lucrului cu sistemul monomodular mi-a accentuat convingerea că utilizarea unor propoziţii certe din limbajul natural este o frână în de experimente care caută funcţionalitatea sistemului. Folosirea unor propoziţii cu sens bine determinat duce la necesitatea unei plauzabilităţi cu sens psihologic a rezultatelor obţinute. Din cauza nuanţelor şi a semanticii vagi a limbajului natural este foarte greu ca să obţinem rezultate edificatoare. Alt motiv foarte serios este limitarea vocabularului folosit. Este foarte greu de construit o bază de cunoştinţe cu sens, legată de o problematică reală, cu un vocabular constrâns la un număr mic de relaţii, clase şi individualităţi. Realizarea unei aplicaţii bazate pe modelul propus de mine ar fi în primul rând o căutare a aplicaţiei potrivite şi a rezolvării problemei mapării între forma de exprimare a cunoştinţelor şi a structurilor simbolice folosite. Din păcate, distanţa de la cunoşinţele exprimate în limbaj natural până la grafurile conceptuale folosite este destul de mare.

O analiză critică a experimentelor efectuate în continuare (şi aici intră şi cele din capitolele 4 şi 5) ar putea avea ca punct central *lipsa semanticii asociată simbolurilor folosite*. O argumentaţie în favoarea metodei mele ar fi următoarea: omul a inventat de-a lungul istoriei o mulţime de limbaje pentru comunicare şi reprezentare a cunoştinţelor sale, folosind simboluri şi mijloace foarte diverse. Toate limbajele au ca şi caracteristică comună, posibilitatea stabilirii de asocieri între simbolurile (cuvinte sau pictograme) folosite, şi/sau stabilirea unor ierarhii şi categorizări ale acestora. Nu văd de ce nu putem alege o cale inversă pentru definirea unor structuri de test, în sensul ca mai întâi să definim limbajul (vocabular, categorii, ierarhii, asocieri) şi apoi să definim structuri, chiar dacă acestea nu au o interpretare într-un univers real (semantică). Am încercat să aloc simbolurilor cuvinte ale limbajului natural, dar rezultatele au fost neconcludente. Consider ca aceasta se datorează complexităţii foarte mari a limbajului natural şi mai ales a lumii înconjurătoare, descrisă de cuvintele noastre. Putem să ne imaginăm însă o lume mai simplă, artificial creată, descrisă de cunoştinţele folosite de mine în experimente.

Cel mai simplu sistem conform modelului multimodular hibrid eterogen putea fi obţinut pornind de la un sistem compoziţional eterogen cu două module. Punctul de plecare a fost construirea unei baze de cunoştinţe care să poată fi folosită prin forward chaining. Vocabularul pe care l-am conceput a fost minimal, 15 cuvinte care erau:

- 3 clase (c1, c2, c3) {roles}
- 5 relaţii (p1 - p5) {predicates, links}
- 7 individualităţi (f1 - f7) {fillers}

Pentru a obţine puncte de întrerupere, am adăugat un modul compus din reguli care puteau fi folosite prin backward chaining astfel încât cele două module puteau comunica prin blackboardul asociat nivelului simbolic pentru executarea unor linii de

raţionament. Schema de lucru a sistemului era următoarea. Utilizatorul scria o propoziţie pe blackboard-ul nivelului simbolic BC, aceasta operaţie având sensul epistemologic de "este această propoziţie adevărată?; dacă da, atunci în ce condiţii?". Modulul simbolic bazat pe backward-chaining (l-am notat MB) stabilea o propoziţie (ar putea fi mai multe dar pentru simplitate am ales variantă cu lanţuri de raţionament iniţiale formate din noduri cu o singură propoziţie) care avea sensul "propoziţia de la intrare este adevărată, daca şi numai dacă propoziţia nou generată este adevărată. Modulul bazat pe mecanismul de forward chaining (notat MF) prelua această propoziţie şi o interpreta conform regulilor sale, în sensul "dacă se pune întrebarea P atunci trebuie să se pună şi întrebarea Q". Întrebarea Q este pusă pe blackboard-ul BS şi este citită de modului MB, ş.a.m.d.

Notând propoziţiile cu literele mici ale alfabetului, se poate ilustra setul de reguli din modulele simbolice MF şi MB. Pentru modulul MB o schemă iniţială a regulilor poate fi:

$e \rightarrow a$
$f \rightarrow b$
$g \rightarrow c$

$x \rightarrow h$
$y \rightarrow i$
$z \rightarrow j$

iar pentru MF:

$?e \rightarrow ?h$
$?f \rightarrow ?i$
$?g \rightarrow ?j$

Plecând de la întrebarea **?a**, sau **?b**, **?c**, putem avea trei linii de raţionament care se termină cu propoziţiile **x**, **y**, sau respectiv **z**. Nu mai există altă informaţie care să poată continua lanţul de raţionament. În mod intenţionat, pentru a introduce puncte de întrerupere, am modificat regulile din MB în felul următor:

$e \rightarrow a$
$f \rightarrow b$
$g \rightarrow c$

$x \rightarrow (h \wedge m)$
$y \rightarrow (i \wedge n)$
$z \rightarrow (j \wedge o)$

Se poate observa destul de uşor că în pasul al treilea al raţionamentului compoziţional este necesară o propoziţie suplimentară (de la caz la caz, **m**, **n** sau **o**). Este acceptat

faptul că prezenţa mai multor propoziţii pe blackboard înseamnă din punct de vedre logic o conjuncţie. Dacă utilizatorul consideră că propoziţiile **m**, **n** sau **o** sunt întotdeauna adevărate şi le poate furniza ca şi "hint", această situaţie conduce la posibilitatea rezolvării automate a acestui punct de blocare.

Apariţia unui punct de starvation în acelaşi pas, pentru linii diferite de raţionament, face ca să putem identifica un PGB (care conţine doar trei WL-PHINT-uri, faţă de patru câte am impus, dar pentru acest sistem minimal, am acceptat şi o definiţie cu trei). Cele trei WL-PHINT-uri, de lungime 3, vor fi:

(**a**, **e**, **m**)
(**b**, **f**, **n**)
(**c**, **g**, **o**)

Acestea se transformă în traiectorii T2P prin translatare tensorială (atenţie, propoziţiile nu sunt atomice, au o structură predicativă, obţinută prin vocabularul prezentat) şi sunt folosite ca bază de învăţare pentru un singur modulul conexionist (notat MC). Acesta a avut un număr de 250 de unităţi din care 125 erau de ieşire şi un număr de 250*(250+150) de ponderi (necesar de memorie - 0.8MBytes), o dimensiune destul de mică - antrenarea fiind obţinută cu un efort de 1451 de epoci până la o limitare a erorii de 5%.

Un exemplu de funcţionare este următorul: utilizatorul pune întrebarea **a**, aceasta este scrisă pe BS şi pe BC. Modulul MB produce propoziţia **e**, aceiaşi propoziţie fiind predictată şi de modulul MC. Propoziţia **e** este consumată de de MF şi este ştearsă de pe BS şi se produce **h** care este scrisă pe BS. Dar propoziţia **e** este transmisă conform protocolului de comunicare între nivele şi pe BC de unde este consumată de MC, acest modul producând la ieşire propoziţia **m**, conform traiectoriei asociată WL-PHINT-ului (**a**, **e**, **m**). Dat fiind faptul că la nivelul simbolic apare un punct de întrerupere, informaţia de pe BC este scrisă pe BS, acesta conţinând simultan propoziţiile **h** şi **m** care vor fi acceptate în acelaşi pas de MB care va putea genera răspunsul aşteptat **x**. Pentru celelalte trei linii de raţionament, procesul este identic.

Se poate afirma în acest moment că rolul modulului conexionist ar putea fi cu uşurinţă jucat de un modul simbolic mult mai simplu. Se mai poate accentua şi predictibilitatea comportării modulului conexionist. Dar tocmai posibilitatea utilizării reţelei neuronale este probată, nu căutarea unui alte metode de "machine learning". Există metode care se pot aplica unor sisteme compoziţionale [Peng96], dar sunt limitate datorită complexităţii combinatoriale. Acest sistem este funcţional, modulul conexionist funcţionând aşa cum m-am aşteptat, iar antrenarea reţelei a fost un task doar de ordinul orelor pe o staţie de lucru obişnuită. Este adevărat că multe din operaţiile de construire ale modulului conexionist le-am făcut manual, dar automatizarea lor este o problemă destul de simplă (relativ la identificarea punctelor de starvation, a WL-PHINT-urilor, a stabilirii structurii reţelei şi a antrenării). Practic, un sistem complet automat, ar constata punctele unde utilizatorul dă informaţie

sistemului, ar grupa un număr de linii de raţionament care conduc în aceste puncte şi ar antrena o reţea recurentă pentru fiecare din aceste grupe.

Problema cea mai dificilă ar fi distribuirea liniilor de raţionament în funcţie de vocabularele asociate lor, astfel încât grupurile să aibă un vocabular cât mai mic. Aceasta problemă de optimizare s-ar putea să fie foarte complicată, dat fiind faptul că apar multe cerinţe (număr de traiectorii, lungimi, număr de unităţi minim), fiind principalul motiv pentru care nu am încercat să intru in detaliile de implementare ale unui subsistem automat de identificare / construire / antrenare a modulelor conexioniste. Am considerat că efortul merită concentrat în altă direcţie. Ca şi observaţie la sistemul minimal, se impune foarte clar, ca modulul (în acest context, chiar nivelul) conexionist să "cunoască" tot vocabularul sistemului simbolic.

6.4.2. Faza sistemului extins

În timpul experimentului anterior, din start, în mod intenţionat, am introdus în sistem cauzele pentru apariţia momentelor de starvation. În acest al doilea set de experimente, nu am mai prevăzut din etapa de proiectare care sunt punctele de întrerupere. Acesta este şi abordarea naturală, un proiectant de baze de cunoştinţe nefăcând altceva decât să adauge noi cunoştinţe, verificând dacă acestea duc la incosistenţe, sau în cazul sistemelor compoziţionale, la puncte de întrerupere. Folosind acest stil de construire a modulelor simbolice, este dificil de prevăzut toate punctele de întrerupere, mai ales dacă sistemul are un număr relativ mare de reguli. Din acest motiv, găsirea punctelor de întrerupere trebuie făcută exclusiv prin încercări, prin traversarea unor linii de raţionament de către utilizator (care furnizează şi "hint"-urile necesare).

Ar fi fost convenabilă crearea un subsistem automat de identificare a punctelor de întrerupere. Nu am folosit o astfel de strategie, pentru ca ar fi fost dificilă de implementat, ci pur şi simplu am schiţat un număr de linii de raţionament, cunoscând paşii de început şi de sfârşit, lăsând intenţionat un număr de reguli incomplete la aproximativ mijlocul liniilor de raţionament. În mod normal, identificare informaţiei lipsă se face prin cunoaşterea semanticii propoziţiilor care formează linia de raţionament. În cazul meu, unde propoziţiile nu au o semantică (relativ la cuvinte din limbajul natural) definită, identificarea propoziţiilor care sunt necesare ca "hint"-uri, se face studiind regulile din modulul care urmează să continue linia de raţionament. Modul acesta de lucru este poate nenatural, dar repet, cele două soluţii erau fie stabilirea intenţionată la proiectare a tuturor punctelor de întrerupere (şi am remarcat că este dificil de făcut pe măsură ce sistemul creşte în dimensiune), fie prin asignarea unei semantici propoziţiilor din sistem (dar m-aş fi îndepărtat de problema în sine).

Am dezvoltat o baza de cunoştinţe formată din reguli predicative, care putea fi utilizată prin mecanismul de forward-chaining. Am folosit un vocabular de 43 de cuvinte, împărţite pe categorii în felul următor:

- 6 clase (c1 - c6)
- 15 relaţii (p1 - p15)
- 22 individualităţi (f1 - f22)

Regulile le-am separat în două module pur simbolice, criteriul de separare fiind vocabularul, astfel încât cele două vocabulare asociate modulelor să fie cât mai echilibrate ca număr de cuvinte. Numărul de reguli rezultate a fost de 30 pentru fiecare modul. Se poate afirma creşterea sistemului "cu un ordin de mărime", deşi într-un context pur simbolic este dificil de definit "ordinul de mărime". Vocabularul a crescut de trei ori, iar numărul de reguli de aproximativ 10 ori. Numărul de module simbolice a crescut doar cu un modul, al treilea modul simbolic fiind un modul bazat pe mecanismul de backward-chaining (primele două fiind module bazate pe forward-chaining).

Ca şi la sistemul minimal, linia de raţionament era începută de modulul bazat pe backward-chaining (notat MB), care citea informaţia pusă pe BS de utilizator (şi care era sub forma unei întrebări). Modulul MB genera o propoziţie rezultat, care era scrisă pe MB. În acest moment, ar fi fost necesară existenţa unui secvenţiator care să aleagă între cele două module bazate pe forward-chaining (notate MF1 şi MF2). Pentru a simplifica gestiunea liniilor de raţionament, am ales o schemă de alegere a modulului care preia informaţia de pe BS bazată pe vocabular, într-un stil asemănător cu metoda folosită la sistemul multimodular omogen (conexionist - vezi capitolul 5). Am asociat fiecărui modul (MF1 şi MF2) câte un vocabular, iar propoziţiile care puteau fi citite de un modul de pe blackboard, erau doar acelea care aveau cuvintele componente membre ale vocabularului modulului. În cazul în care propoziţia putea fi redusă la ambele vocabulare, am acceptat ca ambele module să fie capabile să citească propoziţia şi să genereze câte un rezultat. Cazul favorabil era ca numai un modul să genereze răspuns, iar în cazul în care apăreau două propoziţii, se punea problema unei conjuncţii logice, care să modifice comportamentul modulului care trebuia să preia linia de raţionament (considerat de mine a fi modulul MB). Important este din punctul meu de vedere că o astfel de construcţie duce la un comportament nedeterminist.

Am adaptat şi modificat regulile din cele trei module astfel încât să obţin aproximativ zece linii de raţionament (de la o propoziţie start la una ţintă) de lungimi aproximativ egale (12-14 paşi). Modulul MB putea să pornească şi pe alte linii de raţionament, dar acestea nu au fost prevăzute intenţionat, ele fiind un potenţial de lucru pe care l-am folosit ulterior (au fost detectate în final încă zece linii neintenţionat gândite). Studiind cele zece linii intenţionat pregătite, am detectat 7 puncte de întrerupere, care se apăreau întotdeauna după o scriere pe BS efectuată de modulul MB. Figura 6-2 ilustrează locul unde apăreau aceste puncte. Cele 7 "starvation points" se puteau grupa în două PGB-uri, datorită atât vocabularului, cât şi lungimii diferite a liniilor de raţionament care canduceau la aceste puncte. Primul PGB consta din:

WL-PHINT1 - de lungime 5
WL-PHINT2 - de lungime 6
WL-PHINT3 - de lungime 7

iar al doilea din următoarele patru WL-PHINT-uri (4, 5, 6 şi 7) de lungimi 8, 8, 9 şi

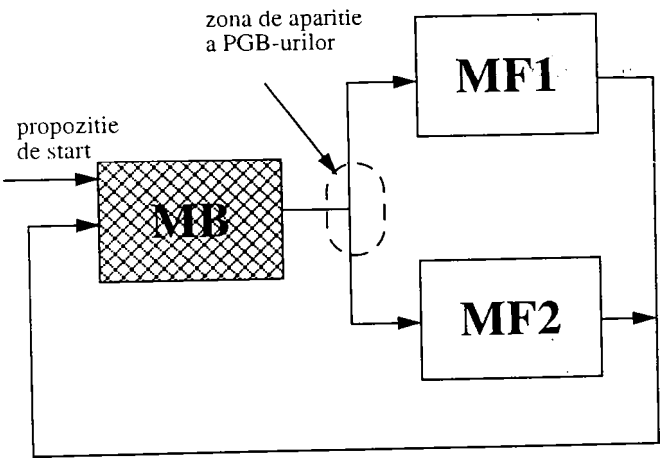


figura 6-2.

respectiv 10 paşi. Conform schemei MILD de partiţionare, am construit un modul conexionist pentru învăţarea celor trei traiectorii rezultate din WL-PHINT-urile 1, 2, şi 3. Lungimea acestor traiectorii a fost limitată la 5. Acest modul conexionist (notat MC1) a fost uşor de antrenat, dat fiind numărul mic de traiectorii. Pentru celelalte 4 traiectorii, de lungime limitată la 8, antrenarea devenea prea intensivă computaţional, astfel încât am folosit un modul de prefaţă (MCP2), cu 4 traiectorii de lungime 4 şi un modul principal (MC2), cu partea a doua a celor patru traiectorii, de lungime 4 (lungime totală - 8). Antrenarea celor trei module conexioniste a fost echilibrată, dat fiind şi echilibrul aproximativ între vocabularele folosite, 20 de cuvinte pentru MC1, 19 cuvinte pentru MC2 şi 22 de cuvinte pentru MCP2. Dimensiunile reţelelor recurente folosite au fost:

- 300 cu 140 de unităţi de intrare şi 140 de ieşire pentru MC1
- 280 cu 135 de unităţi de intrare şi 135 de ieşire pentru MC2
- 300 cu 160 de unităţi de intrare şi 160 de ieşire pentru MCP2

Am remarcat din nou că reuniunea vocabularelor modulelor conexioniste reprezintă întregul vocabular de la nivelul simbolic.

Potrivirea acestor valori s-a dovedit o muncă destul de dificilă. Regula empirică care se poate aplica este: numărul de unităţi de intrare şi de ieşire datorate vocabularului, înmulţit cu doi, dă numărul de unităţi al reţelei. În funcţie de numărul de puncte care formează traiectoriile de antrenat, numărul de unităţi suplimentare (nededicate pentru ieşire) poate oscila. Dacă avem 4-5 paşi regula aceasta duce la un număr de aproximativ 500 până la 1500 de epoci pentru fiecare reţea (de ordinul orelor pe o staţie obişnuită). Dacă traiectoriile sunt mai lungi, numărul de unităţi suplimentare trebuie să crească, altfel numărul de epoci devenind mult mai mare (zeci de mii). Mai mult, creşterea numărului de unităţi şi a lungimii traiectoriilor măresc simţitor (polinomial) timpul necesar pentru o epocă. Limita admisibilă consider că este maximum 30 de cuvinte/modul şi lungimi de traiectorii de maximum 6 puncte (400 de unităţi), antrenarea pe o staţie performantă putând să dureze zile şi chiar săptămâni dacă numărul de epoci necesare atingerii unui minim de eroare de 5% este destul de mare.

Important a fost că am reuşit să demonstrez că un astfel de sistem poate fi extins la un număr mai mare de module şi pentru un vocabular mai consistent. Ceea ce a fost interesant şi consider că este reuşita majoră a acestui experiment este următorul fenomen. Descoperind noile linii de raţionament, prin experimentarea funcţională a sistemului (ele ar fi putut fi descoperite şi automat, funcţionarea fiind nedeterministă doar din punctul de vedere al extinderii seturilor de reguli), am identificat şi situaţii care puteau fi interpretate ca puncte de întrerupere. Mai exact, furnizând un "hint" aceste linii de raţionament (relativ scurte, de 7-8 chiar 9 paşi) puteau fi continuate prin paşi posibili, până la o lungime aproximativ egală cu liniile construite intenţionat (12-14 paşi şi chiar mai mult). Repet, aceste linii de raţionament nu au o semantică precizată, relativ la limbajul natural, sau la realitate, iar sensul lor ca întreg nu poate fi precizat. Este însă plauzibil să afirm, că în cazul în care o astfel de linie de raţionament este dezvoltată de un utilizator, ea să fie folositoare şi dorită

(în cazul unui sistem cu semantică asociată simbolurilor utilizate). Evident că şi afirmaţia contrară poate fi susţinută, adică un "hint" nepotrivit să ducă la un rezultat nedorit. Este puţin credibil însă că două linii parţiale de raţionament, din care una începe cu pasul de start şi se termină la pasul *t* din motiv de starvation, putând fi continuată la pasul *t+1* prin adăugarea unei singure propoziţii lipsă, să nu fie de fapt o singură linie căreia îi lipseşte o legătură.

Am identificat două astfel de linii incomplete (pe lângă cele 10 linii complete "neintenţionate"). Nici nu m-aş fi gândit la astfel de linii incomplete dacă o comportare neaşteptată a sistemului nu mi-ar fi atras atenţia. Acest lucru s-a întâmplat în timpul în care căutam în mod empiric, "lucrând" cu sistemul, liniile de raţionament introduse în mod nedeterminist. Această căutare se putea face în două moduri. Fie lucrând doar cu nivelul simbolic al sistemului, fie lucrând şi cu nivelul conexionist conectat, pentru a obţine şi liniile care au fost completate prin intervenţia modulelor conexioniste antrenate. *Lucrând cu întreg sistemul, am observat că una din cele două linii de raţionament care nu fusese prevăzută iniţial, era rezolvată prin intervenţia unui modul conexionist* (mai exact MC2) în pasul 6 (în pasul 2 al modulului MC2). Mai exact, după pasul al şaselea al raţionamentului, din motive de starvation, raţionamentul era oprit. Pentru că am considerat că doar liniile cu minim 10 paşi sunt complete, sistemul citea automat de pe BC pe BS, ultimele propoziţii generate de modulele conexioniste. Erau două astfel de propoziţii, una generată de MC1, iar a doua de MC2. Dacă prima nu era "apropiată", nici ca vocabular, nici ca poziţie în linia de raţionament cu "hint"-ul necesar, a doua era chiar propoziţia cerută pentru a continua raţionamentul. A fost o întamplare, dar acest lucru mi-a atras atenţia asupra efectelor pe care le puteau induce modulele conexioniste. Cu toate că am considerat că efectul constatat era favorabil şi îmbucurător, am căutat să găsesc şi efectele nefavorabile ale intervenţiei prin "hint", oferită de nivelul conexionist.

Efectul nefavorabil ar fi fost crearea unor linii de raţionament "parazite" care să fie formate din mai multe linii parţiale (mai multe decât două), prin oferirea unui număr exagerat de mare de "hint"-uri care să se şi potrivească. Căutând astfel de situaţii, nu am mai găsit decât o singură linie, care era incompletă doar într-un singur pas (pasul 8), deci putea fi considerată o linie favorabilă, care are nevoie de un singur hint. Mai mult, în cazul aceste linii incomplete, **am descoperit că în pasul 6, pe BC apărea propoziţia lipsă, care trebuia oferită ca hint in pasul 8**. Din cauza protocolului de comunicare (cel mai simplu, notat cu unu roman în secţiunea 6.2.3) această propoziţie era ştearsă, nemaiputând fi utilizată în paşii ulteriori. Din acest motiv am încercat şi alte protocoale de comunicare, cu riscul de a introduce linii parazite, prin aglomerarea de propoziţii pe cele două blackboard-uri. Utilizând protocolul al doilea de comunicare, în care doar propoziţiile consumate de pe BC erau şterse, propoziţia respectivă rămânea pe BC şi era folosită de sistem în momentul starvation point-ului. Dar nici acest lucru nu este suficient.

Al treilea protocol l-am folosit pentru a evita o situaţia în care propoziţia care este utilă într-un pas ulterior, este consumată prematur de către modulele conexioniste,

pentru că vocabularul acestora permite acest lucru. Nu acesta era cazul în exemplul de mai sus, pentru că doar MC2 putea consuma această propoziţie, iar încercarea de a procesa această propoziţie particulară în pasul respectiv, ducea la stagnare, adică reţeaua rămânea în acelaşi punct de pe traiectorie. Am considerat însă acest caz ca fiind foarte particular, probabilitatea ca propoziţia utilă să fie consumată la nivel conexionist şi să dispară să fie foarte mare. Acest protocol, în ambele sale variante are însă dezavantajul că induce un potenţial pentru apariţia unor linii de raţionament parazite prin păstrarea unui număr de propoziţii mult prea mare pe cele două BB-uri. Însă la un sistem cu un număr atât de mic de module, relevanţa acestui fenomen nu a putut fi detectată, nici o linie de raţionament suplimentară nemaifiind detectată, chiar dacă se folosea cea mai complicată schemă de protocol (IIIb). Singura soluţie pentru studiul acestui fenomen parazit, era construirea unui sistem şi mai mare (mai ales ca număr de module). În plus, un astfel de sistem avea şi avantajul că putea arăta dacă fenomenul favorabil de rezolvare automată a unor "hint"-uri nedeterminate iniţial se păstrează şi la o scară mai largă şi reprezintă o posibilă caracteristică a acestui tip de model.

6.4.3. Faza de urmărire a generalizărilor

Pentru a studia efectul de generalizare constatat anterior am apreciat că cea mai bună metodă este tot una experimentală. Metoda mea a fost de a utiliza un sistem mai mare, care dacă ar fi prezentat acelaşi comportament favorabil, într-o masură egală cu creşterea lui relativă, ar fi reprezentat un argument necesar (dar nu şi suficient - o abordare teoretică, prin formalizarea sistemului de mai sus este după părerea mea necesară, chiar dacă ar fi extrem de dificilă). Este însă destul de greu de apreciat ce înseamnă un sistem "mare", în raport cu cele două sisteme experimentale dezvoltate anterior. Pur şi simplu am considerat că un sistem sufiecient de mare pentru studiul propus este sistemul cel mai voluminos care poate fi construit de către un singur experimentator, într-un timp limitat. Poate că ar fi necesare experimente cu un sistem şi mai mare, dar aceasta presupune munca unui colectiv, timp, şi mai multe resurse de calcul.

Prima dimensiune pe care am operat mărirea sistemului a fost cea a vocabularului. Numărul total de cuvinte a fost 120, împărţite pe categorii în felul următor:

9 clase,
43 de predicate,
68 de individualităţi.

Numărul de reguli care pot fi scrise cu un astfel de vocabular este foarte mare. A existat însă o limită la care să ma opresc, astfel încât baza completă de cunoştinţe să fie lizibilă şi coerentă (se pot genera reguli şi aleator, dar probabilitatea de a avea linii de raţionament este mult mai mică). Dacă aş fi păstrat aceaiaşi reguli/modul ca la sistemul extins cu trei module simbolice, aş fi putut obţine un număr mare de module, în jur de 20. Ar fi fost o creştere semnificativă a numărului de module, deci şi o

creştere aparent semnificativă a sistemului. Deşi informaţia înglobată sub formă de reguli era de aproximativ de zece ori mai mare, creşterea numărului de module, păstrând aproximativ constantă rata de reguli/modul, nu este o soluţie prea bună, pentru ca duce la fragmentarea exagerată a liniilor de raţionament posibile, adică la apariţia a prea multor puncte de întrerupere. Practic, am observat că folosind un număr mare de module, nici o linie de raţionament nu este completă. Plauzibil este să considerăm că doar o mică parte din liniile de raţionament posibile au linii de întrerupere (deşi la sistemul minimal au fost trei linii şi toate au fost întrerupte). Este greu de apreciat care este cea mai bună rată de **linii_întregi/linii_întrerupte** pentru sistemele "normale" (la sistemul extins, rata era de 11/9, dar era un sistem relativ mic).

Cu acelaşi număr de reguli, se puteau face diverse partiţionări în module. Metoda de partiţionare pe care am folosit-o a fost cea de a scrie toate regulile folosite prin forward-chaining într-un singur bloc (astfel încât să existe coerenţa necesară pentru realizarea unor linii de raţionament) şi apoi de a împărţi aceste reguli între module care astfel încât regulile care erau înlănţuite să fie în module diferite (tocmai pentru a oferi posibilitatea de a produce linii de raţionament compoziţionale). La fel am procedat şi cu regulile care erau evaluate prin backward-chaining, scriindu-le pe toate la un loc şi apoi separându-le. Cu cât împărţeam regulile în mai multe module, cu atât obţineam mai puţine linii complete de raţionament. Acestea sunt necesare, pentru că există linii asemănătoare ca şi structură a propoziţiilor, care sunt incomplete, metoda de căutare prin similaritate facilitând identificarea lor. Această metodă nu va găsi toate liniile incomplete (considerând ca incompletă, orice linie din care lipseşte doar o singură propoziţie), dar este cel mai simplu mod de lucru. Dacă pentru un număr mare de module numărul de linii de raţionament complete era zero, odata cu scăderea numărului de module, numărul de linii identificate complete era din ce în ce mai mare.

Cele mai bune rezultate le-am obţinut pentru următoarea configuraţie: două module bazate pe backward-chaining şi patru module bazate pe forward chaining; care au dus la identificarea a 55 de linii de raţionament complete. Ce înseamnă în acest context o linie de raţionament completă? Plecând de la sistemul iniţial monobloc, putem identifica numărul de paşi necesar pentru găsirea unui rezultat. Acest lucru se poate varia în funcţie de "forma" grafului de decizie care poate fi construit pornind de la reguli. Cu acelaşi număr de reguli, putem obţine linii de raţionament lungi de 30-40 de paşi (şi atunci vor fi mai puţine linii de raţionament posibile) sau linii scurte (de 12-14 paşi, ca şi în cazul anterior - dar având multe linii de raţionament). Pentru că posibilitatea de a identifica linii de raţionament foarte lungi este mică, putând să existe prea multe puncte de întrerupere, am ales o variantă cu linii relativ scurte de raţionament (14-16 paşi). Ca schemă de funcţionare, sistemul este prezentat în figura 6-3. Se poate argumenta că dimensiunea sistemului dată de lungimea liniilor de raţionament nu a fost crescută proporţional cu celelate dimensiuni (vocabular, număr de reguli, număr de module). Însă în acest caz sistemul ar fi fost foarte greu de controlat din punct de vedere al raţionamentului însuşi.

Pornind de la aceste linii de raţionament am gasit un număr de 38 de linii de

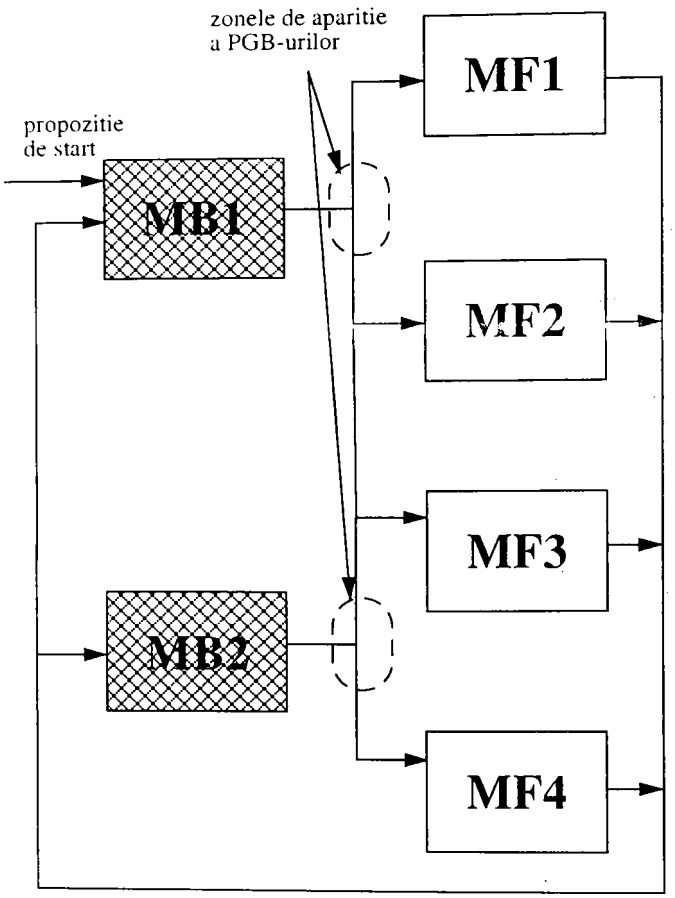


figura 6-3.

raţionament incomplete (care completate cu o propoziţie aveau lungimea de 14-16 paşi). Se poate remarca că proporţia de linii_întregi/linii_întrerupte este aproximativ aceiaşi ca şi la sistemul anterior. Momentele predilecte în care apăreau întreruperile erau paşii 5, 6, 7, 8 şi 9. Punctele în care apăreau situaţiile de "starvation" erau înainte ca raţionamentul să fie continuat de unul din modulele MF_i (vezi figura). Puteam să asociez punctele cu modulele "în faţa" cărora apăreau, dar am decis gruparea lor să se facă doar pe baza vocabularului folosit. Pe baza acestor linii de raţionament incomplete se puteau face traiectoriile de antrenare pentru nivelul conexionist. Pentru că în această fază a experimentării am căutat să determin dacă generalizarea este o caracteristică a sistemului am folosit o metodă de "cross-validation". Am împărţit liniile de raţionament întrerupte în două grupuri egale, urmărind ca şi împărţirea după vocabular şi lungime să fie proporţională. Pornind de la primul grup, am trasat 19 WL-PHINT-uri pe baza cărora am construit modulele nivelului conexionist, folosind schema MILD.

Problema alocării acestor traiectorii la module conexioniste s-a dovedit deosebit de dificilă, în acest punct fiind probabil necesară o metodă de optimizare, multicriterială, dar care ar reprezenta un subiect de cercetare separat. Totuşi, până la acest nivel de complexitate, problema a mai putut fi rezolvată manual. Criteriile de partiţionare ale celor 19 traiectorii erau vocabularele şi lungimile lor. Prima împărţire am făcut-o pe baza vocabularelor, obţinând patru grupe distincte, cu vocabulare destul de puţin apropiate, şi care aveau un grad mic de acoperire.

i. primul grup era format din 8 WL-PHINT-uri, cu un vocabular total de 35 de cuvinte (foarte mare). Lungimea traiectoriilor era de 10 paşi pentru 5 dintre traiectorii, una era de 8 paşi, una de 7 şi una de 4 paşi. Aplicând schema MILD, am obţinut pentru trei PGB-uri, centrate astfel:
- primul în pasul 10 (cu 5 traiectorii),
- al doilea în pasul 7 (cu 2 traiectorii),
- al treilea în pasul 4 (cu o traiectorie).

Am construit trei module conexioniste MC1, MC2 şi MCP1. MC1 şi MCP1 (prefaţa lui MC1) erau antrenate cu cele 5 traiectorii de lungime 10, împărţite în semitraiectorii de lungime 4 (pentru MCP1) şi lungime 6 (pentru MC1). Pentru a nu face un modul separat pentru traiectoria cu lungimea 4 (oarecum izolată de restul), am antrenat această traiectorie în modulul MCP1. Cele două traiectorii de lungime 7, au fost baza de antrenare pentru MC2. Caracteristicile pentru aceste module sunt date în tabelul următor:

nume	număr traiectorii	lungime traiectorii	cardinalitate vocabular	număr de unităţi (total + I/O)	necesar de memorie (MBytes)
MCP1	6	4	28	340 + 220	1.9
MC1	5	6	25	300 + 190	1.4
MC2	2	7	18	250 + 135	0.9

Efortul de antrenare a fost relativ dezechilibrat, pentru MCP1 fiind necesare câteva zile (aproximativ 34 de ore pentru 1200 de epoci) şi trecerea prin câteva încercări nereuşite din cauza unor bifurcaţii. Celelalte două module au putut fi antrenate cu preţul unei zile de rulare pentru fiecare. Se poate observa că apropierea de limita de 30 de cuvinte, duce la obţinerea unor timpi de antrenare prohibitivi.

ii. al doilea grup era format din 7 WL-PHINT-uri, cu un vocabular foarte diferit de primul grup, format tot din 35 de cuvinte. Am împărţit aceste traiectorii în două PBG-uri. Primul conţinea 4 traiectorii (de lungime 9, 9, 9 şi 8) şi am limitat lungimea T2P-urilor la 8 paşi. Al doilea conţinea 3 traiectorii (de lungime 5, 6 şi 7) şi am limitat lungimea T2P-urilor la 5 paşi. Configuraţia modulelor construite pe baza acestui grup de WL-PHINT-uri este dată în tabelul următor:

nume	număr traiectorii	lungime traiectorii	cardinalitate vocabular	număr de unităţi (total + I/O)	necesar de memorie (MBytes)
MCP3	4	4	25	300 + 200	1.5
MC3	4	4	22	300 + 180	1.4
MC4	3	5	20	250 + 150	1.1

Efortul de antrenare a fost destul de mic, singurul modul care a avut nevoie de peste 24 de ore de rulare fiind MCP3. Am observat că păstrând un număr de unităţi suplimentare (diferenţa dintre numărul total şi numărul de unităţi de I/O) mai mare decât în experimentele anterioare, numărul de epoci de antrenare necesar pentru scăderea erorii la 5% era destul de mic (340 pentru MC3 şi 387 pentru MC4 faţă de valorile apropiate de 10^3 constatate anterior).

iii. mai rămâneau doua grupe, una formată din două WL-PHINT-uri de lungime 4 cu un vocabular foarte diferit de restul (22 de cuvinte) şi cealaltă formată dintr-un WL-PHINT izolat de lungime 10 şi deasemenea cu un vocabular foarte diferit de restul (15 cuvinte). Modulele rezultate au avut următoarele caracteristici:

nume	număr traiectorii	lungime traiectorii	cardinalitate vocabular	număr de unităţi (total + I/O)	necesar de memorie (MBytes)
MC5	2	4	22	300 + 180	1.4
MC6	1	10	15	300 + 125	1.2

Antrenarea a fost foarte simplă pentru aceste ultime module. Se poate observa că efortul este proporţional cu vocabularul, şi mai puţin cu numărul de traiectorii. Lungimea conteaza destul de mult, dar doar în cazul în care numărul de traiectorii este

mai mare de 2.

Am obţinut în final 8 module, a căror antrenare a fost totuşi relativ echilibrată. Faptul că numărul de module conexioniste este aproape egal cu numărul de module simbolice este pură întâmplare şi cred că nu există nici o formulă empirică care să definească vreo relaţie între aceste două numere. Doar vocabularul şi factorul de acoperire, precum şi structura traiectoriilor dă în final o schemă echilibrată, optimă pentru modulele conexioniste. Ceea ce trebuie remarcat însă este faptul că din vocabularul total de 120 de cuvinte, aceste module folosesc doar 88. Numărul însumat de cuvinte pe grupuri (35+35+22+15) este 107, dar există şi un uşor factor de acoperire între vocabularele acestor grupuri. Factorul de acoperire este mult mai mare pentru vocabularele modulelor care aparţin aceluiaşi grup (MC5 şi MC6 nu fac parte din acelaşi grup - au vocabulare diferite). Am afirmat anterior că este indicat ca nivelul conexionist să cunoască toate cuvintele din sistem. Şi în acest caz, cunoaşterea tuturor cuvintelor ar fi necesară, în cazul în care am fi antrenat mai multe module, pentru toate cele 38 de WL-PHINT-uri descoperite. Selectând numai jumătate din ele, am redus din păcate şi vocabularul necesar (nu cu jumătate, dar semnificativ).

Căutând care din cele 19 traiectorii neantrenate (lăsate pentru "cross-validation") sunt completate cu "hint"-ul corespunzător la momentul potrivit, am aplicat toate cele patru tipuri de protocoale de comunicare între cele două nivele. Rezultatele prezentate au două aspecte: unul pozitiv, dat de numărul total de WL-PHINT-uri "rezolvate" (prezentat procentual, raportat la totalul de 19) şi unul negativ, dat de numărul de linii de raţionament care nu mai sunt terminate din cauza intervenţiei nedorite a modulelor conexioniste (un fenomen apropiat ca efect cu cel de "over-learning" de la reţelele feed-forward). Acest număr, este format atât din numărul de linii iniţiale complete, şi din numărul de WL-PHINT-uri "rezolvate" explicit prin antrenare. Am numit numărul de "rezolvări" favorabile, **"factor de generalizare"**. Numărul de linii de raţionament pur simbolic (în cazul curent, un total de 55 de linii descoperite) care sunt alterate de funcţionarea nivelului conexionist este **"factorul de deteriorare la nivel simbolic"**. Numărul de linii de raţionament corectate prin nivelul conexionist (prin construirea explicită a modulelor - mă refer la cele 19 WL-PHINT-uri care au dus la antrenarea traiectoriilor), dar care nu sunt terminate din motivul funcţionării defectuoase al acestuia l-am numit **"factor de deteriorare la nivel conexionist"**. Există şi un **"factor de deteriorare globală"** care le îmbină pe cele două anterioare, relativ la toate liniile de raţionament care pot fi executate (mai puţin cele posibile prin WL-PHINT-urile neantrenate). Primele rezultate sunt ilustrate în tabelul următor, reflectând o trecere prin toate cele 55+19+19 traiectorii identificate:

protocol	generalizare (nr.traiectorii, procent)	deteriorare la nivel simbolic	deteriorare la nivel conexionist	deteriorare globală
I	2 (10.5%)	0	0	0
II	3 (15.7%)	0	4 (21.05%)	4 (5.4%)
IIIa	8 (42.1%)	12 (23.6%)	4 (21.05%)	16 (22.6%)
IIIb	6 (31.5%)	6 (10.9%)	3 (15.7%)	9 (12.1%)

Se poate observa că în cifre absolute cel mai bun protocol este primul (I). Aparent, cel mai bun este IIIa, care produce un factor de generalizare foarte mare, aproape jumătate din WL-PHINT-urile neantrenate ducând la traiectorii complete. Dar, o mare parte din traiectoriile inițiale (aproape un sfert) sunt pierdute. Câştigând 8 noi traiectorii, pierdem 16 (un minus de 8 traiectorii, față de cazul protocolul I, unde câştigam două traiectorii şi nu pierdem nici una). Pentru protocolul II, pierdem în final o traiectorie, iar în cazul IIIb pierdem trei traiectorii.

Am căutat să găsesc metode care să îmbunătățească performanțele protocolului I şi metode care să ducă la factori de deteriorare cât mai mici pentru protocoalele care dau generalizări bune. Au fost mai multe idei aplicate, dar *singura care a avut succes* a fost **mărirea vocabularului nivelului conexionist**, până la numărul total de cuvinte din sistem. După mai multe încercări, am observat că aşa se obținea cea mai bună generalizare. O altă metodă, care a generat însă o creştere mică a fost cea a reducerii numărului de module şi creştere a gradului de acoperire. Aceasta a fost aplicată mai întâi şi a condus apoi la idea folosirii tuturor cuvintelor. Prima dată, am încercat să fac modificări relative numai la cele 19 WL-PHINT-uri pe baza cărora am construit primul set de module. Considerând că modulele MC5 şi MC6 sunt prea dezechilibrate ca dimensiune şi foarte depărtate ca vocabular de restul modulelor, cu prețul unor eforturi de antrenare mult mai mari (de ordinul sutelor de ore de rulare - adică săptămâni) am modificat structura modulelor în felul următor:

nume	număr traiectorii	lungime traiectorii	cardinalitate vocabular	număr de unități (total + I/O)	necesar de memorie (MBytes)
MCP1'	7	4	38	380 + 350	2.2
MC1'	6	6	37	400 + 345	2.5
MC2	2	7	18	250 + 135	0.9
MCP3	4	4	25	300 + 200	1.5
MC3	4	4	22	300 + 180	1.4
MC4'	5	4	32	350 + 275	1.75

Cel mai greu de antrenat a fost modulul MCP1 (care a primit prima jumătate din traiectoria dedicată în experimentele anterioare modulului MC6 şi aproape întregul vocabular asociat), necesitând 225 de ore de rulare şi mai multe şedinţe din motive de bifurcaţii. Pot să spun că a fost cea mai frustrantă parte din toată munca desfăşurată pe parcursul celor 4 ani de experimente. Totuşi, în final, am obţinut o eroare de sub 5%. Deşi de o dimensiune mai mare şi un timp_de_antrenare/epocă mai mare, MC1 (care avea acum în plus a doua jumătate a respectivei traiectorii) a putut fi antrenată mult mai uşor (109 ore şi fără bifurcaţii). Cele două traiectorii de lungime 4 al modulului MC5 au fost alocate acum modulului MC4, prin trunchierea traiectoriilor acestuia de la 5 la 4 paşi. Este important de observat aici că cele trei traiectorii iniţiale au fost de lungimile 5, 6 şi 7 paşi. Voi nota cele trei traiectorii cu TFIVE, TSIX şi TSEVEN, pentru că trunchierea lor este relevantă la nivelul experimentelor.

Am reluat experimentele cu noul nivel conexionist format de data aceasta din cele 6 module, din care trei erau de dimensiuni mărite, ca vocabular şi ca număr de traiectorii învăţate (şi ca unităţi şî necesar de memorie, dar acest lucru nu este semnificativ la nivel funcţional). Am măsurat aceiaşi factori de performanţă şi deteriorare şi am obţinut următoarele rezultate:

protocol	generalizare (nr.traiectorii, procent)	deteriorare la nivel simbolic	deteriorare la nivel conexionist	deteriorare globală
I	5 (26.3%)	0	0	0
II	3 (15.7%)	0	3 (15.7%)	3 (4.05%)
IIIa	10 (52.6%)	12 (23.6%)	3 (15.7%)	14 (20.2%)
IIIb	7 (36.8%)	7 (12.7%)	3 (15.7%)	10 (13.5%)

Rezultatele sunt mult mai bune pentru protocolul I (un plus de 5 linii de raţionament - mai mult de un sfert din cele 19 lăsate pentru cross-validation), pentru protocolul II nu avem nici un raţionament câştigat, iar pentru ultimele două avem o pierdere de 5, respectiv 3 linii. Este poate forţat să afirm că micşorarea numărului de module duce la o îmbunătăţire pentru toate protocoalele. Totuşi, pentru protocolul IIIb, se observă o mică creştere a generalizării (repet, la nivelul acesta de mărime al sistemului şi la numărul de experimente diferite ca şi configuraţie de module conexioniste, este poate o întâmplare).

Semnificativ este însă că cele trei traiectorii care sunt deteriorate (sunt din cele 19 antrenate explicit) sunt TFIVE, TSIX şi TSEVEN. TSIX şi TSEVEN erau şi în cazul anterior deteriorate, împreună cu două alte traiectorii de lungime iniţială 9 (integrate în MCP3 şi MC3), care au fost trunchiate la 8 paşi. Se poate trage concluzia că trunchierea traiectoriilor, pentru a le putea antrena în module care învaţă şi alte traiectorii mai scurte, este dăunătoare, ducând la deteriorarea aproape sigură a liniilor

de raţionament asociate atunci când nu folosim protocolul I de comunicare. Acelaşi lucru l-am observat şi făcând un alt experiment, integrând traiectoria modulului iniţial MC6 nu în MCP1' şi MC1' ci în MC2 (care avea două traiectorii de lungime 7). Am trunchiat traiectoria de lungime 10 la 7 paşi, pentru a putea fi antrenată cu celelalte două şi în plus a mai trebuit să măresc vocabularul MC3 la 32 de cuvinte. Antrenarea a fost mai dificilă (dar mai uşoară decât la MCP1' şi MC1'), iar sistemul părea mult mai echilibrat. Rezultatul a fost că am obţinut rezultate mai slabe decât în cazul prezentat (o generalizare de 15.7% pentru protocolul I), iar traiectoria trunchiată de la 10 la 7 apărea tot timpul la capitolul traiectorii deteriorate (pentru celelalte protocoale). Configuraţia şi rezultatele acestui experiment sunt date în tabelele următoare:

nume	număr traiectorii	lungime traiectorii	cardinalitate vocabular	număr de unităţi (total + I/O)	necesar de memorie (MBytes)
MCP1	6	4	28	340 + 220	1.9
MC1	5	6	25	300 + 190	1.4
MC2'	3	7	32	350 + 240	2.0
MCP3	4	4	25	300 + 200	1.5
MC3	4	4	22	300 + 180	1.4
MC4'	5	4	32	350 + 275	1.75

(rezultate)

protocol	generalizare (nr.traiectorii, procent)	deteriorare la nivel simbolic	deteriorare la nivel conexionist	deteriorare globală
I	3 (15.7%)	0	0	0
II	3 (15.7%)	0	4 (21.05%)	4 (5.4%)
IIIa	9 (47.3%)	12 (23.6%)	4 (21.05%)	16 (22.2%)
IIIb	6 (31.5%)	7 (12.7%)	4 (21.05%)	11 (14.8%)

Aceste rezultate indică totuşi o uşoară creştere a factorului de generalizare pentru două din cele patru protocoale. Spre deosebire de setul de experimente anterior prezentat, în acest caz factorul de acoperire al vocabularelor este mai mic, pentru că în loc să distribui vocabularul modulului MC6 în două module (MCP1' şi MC1'), l-am reunit doar cu MC2, construind noul modul MC2'. Făcând aceiaşi operaţie de distribuire a vocabularului peste două module şi nu unul singur, rezultatele sunt sensibil mai bune. **Observaţia mea a fost ca e mai bine să am module care să aibă vocabulare cât mai mari, iar intersecţia lor (factorul de acoperire) să fie deasemenea cât mai mare.**

Altă observaţie a fost relativă la numărul de module folosite. Un experiment care nu a căutat să obţină performanţe, a fost antrenarea fiecărei traiectorii cu un modul separat (au rezultat 19 module foarte simple, cu vocabulare izolate). Era normal ca generalizarea să fie zero, dar am constatat o creştere semnificativă a factorului de deteriorare global. Am grupat apoi traiectoriile două câte două (obţinând 10 module), iar factorul de deteriorare a fost mai mic. Pentru experimentele cu 8 şi apoi cu 6 module s-a observat o scădere a aproape liniară a factorului de deteriorare, după cum se observă şi în tabelul următor, care ilustrează evoluţia factorului de deteriorare globală odată cu scăderea numărului de module:

protocol	19 module	10 module	8 module	6 module
II	17.5%	8.1%	5.4%	4.05%
IIIa	44.5%	25.6%	22.6%	20.2%
IIIb	29.1%	20.2%	12.1%	13.5%

Am încercat să găsesc o explicaţie a acestei tendinţe a factorului de deteriorare prin urmărirea liniilor de raţionament în cele 4 cazuri de mai sus. Ca şi la sistemul multimodular omogen, partea conexionistă a acestui sistem are nevoie de "sincronicitate", modulele conexioniste trebuind să urmărească nişte linii de raţionament pe care dacă le "pierd" şi pornesc pe alte linii, conduc la o funcţionare nedorită. Cu cât sistemul are mai multe module conexioniste, cu atât este mai greu de asigurat sincronicitatea, posibilitatea ca unele module să accepte propoziţii la momente nedorite şi să genereze propoziţii nedorite fiind mai mare. Cu cât numărul de module este mai mic, cu atât e mai puţin posibil ca acestea să fie "deviate" de la traiectoriile pe care le-au început. Este însă adevărat că o dată cu creşterea gradului de acoperire al vocabularelor, perturbarea care poate să fie făcută de un modul către altul (prin transmiterea unei propoziţii nedorite în pasul respectiv) este mai uşor să apară.

6.4.4. O partiţionare bazată pe întregul vocabular al sistemului

Urmărind manual liniile de raţionament, am observat ca mai mult de jumătate din cele 19 WL-PHINT-uri neantrenate ca traiectorii la nivel conexionist (T2P-uri), nu puteau să constituie corecţii ale liniilor de raţionament incomplete (la nivel simbolic) din cauza faptului că vocabularul lor era diferit de cel al WL-PHINT-urilor învăţate. Am arătat în subcapitolul anterior care au fost criteriile de alegere şi partiţionare a WL-PHINT-urilor propuse pentru antrenare. Experimentele au arătat că este bine ca numărul de module să fie mic, vocabularele să fie cât mai apropiate (cu un grad mare de acoperire), iar traiectoriile să fie cât mai puţin trunchiate (trunchierea poate duce în plus şi la reducerea vocabularului). Dar cele 19 traiectorii au fost mereu aceleaşi. Am mai făcut însă un experiment, cu alt set de WL-PHINT-uri, alese din setul iniţial de 38, astfel încât vocabularul obţinut prin reuniunea acestor WL-PHINT-uri, să fie exact vocabularul complet al sistemului (120 de cuvinte). Acest lucru s-a putut obţine, fără

a include în setul de antrenare toate cele 38 de WL-PHINT-uri; au fost necesare doar 25. Puteau fi şi mai puţine pentru a obţine vocabularul, dar am ales traiectoriile în aşa fel încât lungimea maximă a T2P-urilor obţinute să fie de 6 (cu două excepţii, de lungime 8), iar trunchierile să fie pe cât posibil evitate.

De această dată, am încercat să grupez WL-PHINT-urile doar în funcţie de lungime, nu în funcţie de vocabular (avantajul experimentului anterior era că în cazul acesta vocabularul era minimal). Chiar dimpotrivă, am ales pentru acelaşi modul WL-PHINT-uri cu vocabulare cât mai diferite, pentru a avea prin reuniune, un vocabular cât mai mare. Folosind această metodă, obţineam cel mai mare grad de acoperire între vocabulare. Ţinând cont şi de efortul computaţional necesar pentru antrenare (nu era recomandat ca modulele să aibă un vocabular mai mare de 30 de cuvinte) şi de echilibrarea acestui efort între module, configuraţia modulelor a fost următoarea:

nume	număr traiectorii	lungime traiectorii	cardinalitate vocabular	număr de unităţi (total + I/O)	necesar de memorie (MBytes)
MC1	2	8	24	340 + 180	1.7
MC2	3	6	27	300 + 190	1.3
MC3	3	5	27	270 + 190	1.2
MC4	3	5	22	250 + 150	0.9
MC5	3	4	25	230 + 190	0.8
MC6	3	6	29	330 + 220	1.8
MC7	3	4	24	270 + 180	1.1
MC8	3	4	27	280 + 200	1.3
MC9	3	4	28	250 + 200	1.1

Se poate observa un număr mic de traiectorii/modul, relativ la un vocabular/modul destul de bogat. Însumând numărul de cuvinte din vocabularele tuturor modulelor, obţinem un total de 223 de cuvinte, care raportat la numărul total de 120, dă un factor de acoperire global de 1.86, mai mare decât la încercările anterioare (unde a fost de aproximativ 1.4). Raportarea se face de această dată faţă de numărul total al cuvintelor, faţă de cazul anterior, unde se făcea faţă de cele 88 de cuvinte care compuneau vocabularul reunit al celor 19 WL-PHINT-uri folosite ca bază de învăţare. Pentru a vedea dacă acoperirea este şi locală trebuie facute şi calculele pentru cele C_9^2 perechi de module. Factorul de acoperire între două module este egal cu raportul dintre numărul total de cuvinte din cele două module şi numărul de cuvinte din mulţimea reunită a vocabularelor. Dacă acoperirea este nulă, acest factor este egal cu 1, iar în caz contrar creşte până la maximum 2 (când acoperirea este totală, modulele având vocabulare identice). Acoperirea ideală am considerat-o ca fiind egală cu 1.5 (media

celor două extreme). Făcând calculele pentru toate perechile de module (28 de perechi posibile), am obţinut valori apropiate de 1.5, cu câteva excepţii (1.31 pentru MC4 cu MC5 - cel mai mic grad de acoperire; 1.77 pentru MC3 şi MC8 - cel mai mare grad de acoperire). Trebuie observat că factorul de acoperire global şi cel local sunt diferite ca şi exprimare matematică, cel global variind între 1.0 şi un număr real egal cu numărul de module (atunci când toate modulele cunosc toate cuvintele), iar factorul local între 1.0 şi 2.0.

Din cauza faptului că numărul de traiectorii/modul este mic, antrenarea reţelelor recurente a fost relativ uşoară, chiar dacă dimensiunea reţelelor a fost mare, datorată vocabularelor consistente (mult peste limita propusă iniţial de 22 cuvinte). Din totalul de linii de raţionament incomplete (38), 25 au fost antrenate, iar pe restul de 13 urma să se facă testul legat de factorul de generalizare. Rezultatele, folosind toate cele patru protocoale de comunicare, sunt date în tabelul următor:

protocol	generalizare (nr.traiectorii, procent - din total 13)	deteriorare la nivel simbolic (din total 55)	deteriorare la nivel conexionist (din total 25 antrenate)	deteriorare globală (din total 80 de linii)
I	7 (54.8%)	0	0	0
II	8 (61.5%)	2 (3.6%)	7 (28%)	9 (11.2%)
IIIa	12 (92.3%)	15 (27.3%)	11 (44%)	26 (32.5%)
IIIb	9 (69.2%)	10 (10.5%)	9 (36%)	19 (23.7%)

Cu toate că factorul de deteriorare creşte foarte mult pentru protocoalele II, IIIa şi IIIb, rezultatele sunt foarte bune relativ la factorul de generalizare (pentru toate protocoalele, mai ales la IIIb). **Factorul de deteriorare este mare pentru că numărul de module este prea mare.** Câştigând multe linii de raţionament, pierdem însă un număr şi mai mare, singura excepţie fiind protocolul cel mai simplu (I). Remarcabil este rezultatul obţinut pentru protocolul IIIa, unde aproape toate (12 din 13) liniile de raţionament incomplete sunt rezolvate de nivelul conexionist. Dar în acelaşi timp, deteriorarea globală este cea mai mare (aproximativ o treime din liniile complete iniţiale sunt pierdute).

6.5. Relevanţa experiementelor

După efectuarea acestor experimente, am tras o serie de concluzii, care pot fi utile pentru cel care dezvoltă un astfel de sistem:

• **soluţia ideală** este ca nivelul conexionist să fie compus dintr-un singur modul, care să cunoască toate cuvintele din vocabularul nivelului simbolic şi să fie antrenat cu cât

mai multe traiectorii identificate ca şi WL-PHINT-uri. În acest mod se poate obţine o perfectă sincronicitate cu nivelul simbolic iar factorul de deteriorare la nivel conexionist poate fi redus la 0. Am presupus că în cazul în care la un pas de raţionament avem o singură propoziţie rezultată la nivel conexionst şi folosim unul din protocoalele de tip III, posibilitatea ca aceasta să perturbe procesul de raţionament este foarte mică.

• pentru ca **nu există niciodata soluţii ideale**, proiectantul este constrâns să dezvolte mai multe module conexioniste. Relaţia combinatorială între vocabular şi dimensiunea reţelelor, complexitatea polinomială de grad mare a algoritmilor de antrenare impun acest lucru. Numărul de module trebuie să fie însă cât mai mic, dus până la limita dimensională a modulelor şi a timpilor lor de învăţare.

• **vocabularele** trebuie să aibă un grad de acoperire locală (două câte două) cat mai aproape de 1.5. Acest lucru va facilita creşterea factorului de generalizare. Am observat că sistemul funcţionează cel mai bine când vocabularul de la nivelul conexionist este identic cu vocabularul de la nivel simbolic.

• **protocolul de comunicare** între cele două nivele trebuie să fie cât mai simplu (ca şi protocolul I). Protocoalele care implică un grad mai mare de transmitere a informaţiei şi păstrarea propoziţiilor pe blackboard-uri timp de mai mulţi paşi, conduc de regulă la degradarea unor linii de raţionament care în cazul protocolelor simple nu erau afectate. Chiar dacă factorul de generalizare creşte spectaculos, trebuie întotdeauna verificat dacă numărul de linii de raţionament "câştigate" nu este mai mic decât numărul de linii de raţionament "pierdute".

Experimentele prezentate mai sus sunt pe departe exhaustive. Din cauza naturii excesive a antrenării reţelelor recurente este foarte dificil a se face în timp rezonabil, de către un singur experimentator, fără resurse deosebite (un accelerator matematic, ar fi indicat) un număr suficient de mare de experimente. Se poate observa că sunt mai multe dimensiuni ale problemei:

- număr de module simbolice
- număr de module conexioniste
- vocabular la nivel simbolic
- vocabular la nivel conexionist
- număr de linii de raţionament
- număr de WL-PHINT-uri identificate
- număr de WL-PHINT-uri antrenate
- vocabularele asociate grupelor de WL-PHINT-uri
- gradul de acoperire global şi cel local al vocabularelor modulelor conexioniste.

Toate acestea ar trebui investigate sistematic, prin variaţia uneia şi fixarea celorlalte pe diverse valori. Se poate observa însă că volumul de experimente este foarte mare. Am încercat să variez acele dimensiuni care mi s-au părut esenţiale (vocabular, număr de

module conexioniste, grad de acoperire) făcând totuşi un număr cât mai mare posibil de experimente.

Un aspect important care ar urma să fie investigat este metoda de a comunica între modulele conexioniste prin vectori de activare şi nu prin structuri simbolice. Şi din punct de vedere al plauzibilităţii cognitive şi al analogiei cu creierul uman, acest mod de implementare ar fi fost mult mai potrivit. Metoda de a translata în fiecare pas de raţionament, în timpul transmiterii informaţiei de la modul la modul, informaţia din starea ei neuronală, distribuită, în structuri simbolice, este o metode foarte rigidă. Cele mai moderne teorii (vezi [McLelland&O'Reilly94], [Fahlman97]) susţin că cea mai bună modelare a creierului uman este prin utilizarea unor reţele recurente conectate prin trunchiuri de transmitere a informaţiei cu dimensiune mult mai mică decât a reţelelor (bottleneck) unde informaţia poate fi la nevoie restabilită în formă simbolică. Realizarea acestor bottleneck prin şiruri de aşteptare care conţin grafuri conceptuale, este cel puţin artificială. Ar fi necesar aplicarea conceptului de "comunicare laterală" introdus în schemele ierarhice multi-reţele de [Miikulainen96]. Comunicarea între modulele conexionste s-ar realiza prin legături tipic conexioniste, existând şi posibilitatea introducerii unor matrici de ponderi, cu o dimensiune mică, care să indice gradul de legătură şi influenţă laterală. Acesta ar fi evident în raport direct cu gradul de similaritate între vocabulare. Problema cea mai spinoasă este fireşte modul în care se vor calcula aceste ponderi. Blackboardul BS trebuie să rămână, ca un segment al punţii de legătură între cele două nivele, dar nu ar mai avea un rol în comunicarea între modulele conexioniste.

Fireşte, asocierea unor cuvinte din limbajul natural ar fi cel mai mare câştig. Dar soluţia pentru introducerea unei semantici "naturale" este lărgirea vocabularului. Dar am văzut că aceasta duce la o modularizare excesivă, ideal fiind de fapt utilizarea unei singure reţele, sau mai multe, dar legate prin comunicarea laterală (formând de fapt una singură, cu o topologie mai deosebită). Ca de obicei, rezolvarea oricărei probleme tehnice presupune un "trade-off", câştigul într-o parte însemnând pierdere în alta.

Concluzii

C1. Un sumar al rezultatelor

Teza de doctorat prezintă o metodologie de îmbunătăţire a performanţelor calitative în sistemele de raţionament compoziţionale eterogene. Motivele pentru care am propus această arhitectură specifică sunt prezentate dealungul celor 6 capitole. Un scop mai general, aflat deasupra scopului general pragmatic, cu un caracter aplicativ de dezvoltare strictă a metodologiei, este investigarea unor modele de integrare simbolic-conexionistă (neurosymbolic integration).

Metodologia se poate aplica la orice sistem compoziţional eterogen fără supervizare, la care problema întreruperii liniilor de raţionament datorată "starvation"-ului se rezolvă prin intervenţia manuală a proiectantului. Soluţia oferită de mine este de a adăuga un subsistem, numit nivel conexionist, compus din unul sau mai multe module bazate pe reţele neurodinamice discrete. Aceste reţele pot fi antrenate cu traiectorii formate din puncte în spaţiul euclidian, care sunt analoage printr-o transformare bijectivă, cu o linie de raţionament a sistemului compoziţional, formată dintr-o secvenţă ordonată de structuri simbolice.

Paşii pentru utilizarea metodologiei pot fi rezumaţi succint în felul următor:

• Sistemul compoziţional pur simbolic este testat de către utilizator şi proiectant, care furnizează "hint"-urile necesare deblocării raţionamentului pentru anumite linii specifice unde apar întreruperi datorate fenomenului de "starvation". Cu cât sistemul este mai exhaustiv testat, cu atât posibilitatea de a lăsa puncte de blocare nerezolvate este mai mică, dar este imposibil de a le rezolva pe toate.

• Se grupează toate liniile de raţionament identificate în primul pas, în funcţie de lungimea lor şi în funcţie de gradul de acoperire al vocabularelor. Se caută ca acesta din urmă să fie cât mai mare.

• Se împart liniile astfel încât să rezulte un număr cât mai mic de grupe, dar din punct de vedere al antrenării, niciuna din grupe să nu ducă prin lungime sau vocabular la module conexioniste care să conţină reţele cu un necesar computaţional exagerat. Dacă este necesar, traiectoriile prea lungi pot fi alocate pe două module consecutive (modul de prefaţă + modul principal).

• Se trasează pentru fiecare linie de raţionament, o traiectorie discretă în spaţiul euclidian, astfel încât acesta să fie netedă şi centrată, iar punctele sale să fie în relaţie corectă cu punctele altor traiectorii similare.

• Se antrenează reţelele modulelor conexioniste, în următoarea ordine: prima dată FFO cu algoritmul Cascor, se stabileşte apoi mărimea stratului ascuns pentru FFI (aceiaşi ca la FFO) şi se antrenează reţeua neurodinamică.

• Se ataşează nivelul conexionist sistemului compoziţional simbolic şi se testează în ordine, cele patru protocoale de comunicare (în ordinea dată în teză: I, II, IIIa, IIIb) şi se evaluează experimental factorul de generalizare şi cel de deteriorare. Se alege în funcţie de acestea, cel mai favorabil protocol.

O cale necesară de extindere a metodologiei este automatizarea procesului de identificare a traiectoriilor cu puncte de blocare şi o partajare asistată a acestora pe modulele, astfel încât să existe o optimizare a procesului de antrenare. Acest aspect devine şi mai important dacă se pune problema ca nivelele să poată fi extinse prin adăugarea unor noi module. Adăugarea de noi cunoştinţe în sistem prin ataşarea unor noi module la nivel simbolic, va duce foarte probabil la apariţia unor noi linii de raţionament cu puncte de blocare. Ar fi de dorit, ca ataşarea unor noi module la nivel conexionist care să rezolve aceste puncte de blocare să fie cât mai automatizată, existând şi posibilitatea ca unele linii noi care trebuiesc antrenate să fie ataşate unor module existente, care vor fi reantrenate. Această posibilitate apare fireşte doar în cazul în care modulul respectiv mai are capacităţi de antrenare nefolosite, iar vocabularul ataşat este suficient pentru linia respectivă.

Ca şi o concluzie mai generală pot spune că cercetarea pentru stabilirea metodologiilor potrivite pentru implementarea sistemelor cognitive este extrem de laborioasă, fapt datorat în primul rând interacţiunii software-mediu de lucru şi de interdependenţa unor aspecte legate de metodologia în sine. Aici nu s-a dovedit deloc fructuoasă izolarea componentelor problemei şi cercetarea concentrată pe subprobleme. Deşi am folosit rezultate notabile ale cercetării din domeniul inteligenţei artificiale clasice şi din domeniul conexionist, cred că nici unul din cele două domenii, în mod separat, nu

vor putea realiza "Modelul", implementabil în software, pentru un sistem cognitiv.

Metodologia propusă de mine pentru implementarea sistemelor cognitive este un exemplu de abordare inginerească, constructivistă şi empirică (la modul non-trivial). Dintr-un anumit punct de vedere, metodologia propune o *ipoteză ştiinţifică*. Mai exact: Se pot simula procese cognitive robuste prin cooperarea a două nivele de raţionament complet diferite ca natură? (simbolic şi conexionist). Dată fiind complexitatea covârşitoare a subiectului, este extrem de dificil, atât experimental cât şi teoretic, de a demonstra dacă ipoteza este falsă sau nu.

Rezultatul cel mai important obţinut este descoperirea că linii de raţionament cu puncte de blocare neantrenate, dar potenţial posibile, apar la funcţionare ca şi confirmare a puterii de generalizare a grupului de module conexioniste utilizate. Interpretarea acestui rezultat este dificilă şi chiar riscantă, dimensiunea sistemului folosit la experimente nefiind suficient de mare pentru a se face supoziţii legate de eventualele performanţe ale unui "real-life system", sau de plauzabilitatea psihologică, atât de căutată în cercetarea din domeniul integrării neuro-simbolice. Utilitatea acestui rezultat va fi găsită mai mult ca sigur în continuarea dezvoltării metodologiilor de acest tip, ca şi fenomen relevat şi analizat pe larg.

Consider că alte două rezultate sunt importante:

• demonstrarea experimentală a generalizării traiectoriilor simbolice (subcapitolul 4.4.3.), împreună cu schema de evitare a bifurcaţiilor (subcapitolul 4.5.3.)

• detectarea artificialităţii distribuirii nivelului conexionist în module separate.

Deasemenea, este important de ştiut că se pot executa raţionamente robuste cu o reţea neurodinamică. Întrebarea este: de ce nu s-a încercat până acum şi de ce nu se aplică metoda pe larg ? (deşi o demonstraţie teoretică a acestei posibilităţi - pentru reţele liniare - există din 1986, vezi [Smolensky86]). Răspunsul: complexitatea computaţională a antrenării unei reţele recurente (sistem dinamic neliniar - nu există algoritmi de antrenare pentru reţele recurente linare) este foarte mare, în special pentru un sistem viabil pentru piaţă. Dar la fel de important este de remarcat că algoritmii folosiţi nu sunt intractabili, o estimare personală, pe baza experienţei dobândite, fiind de 10^{20}, iteraţii necesare pentru construirea unor sisteme comerciale bazate pe această metodologie. În acest caz, întrebarea devine: când vom avea la dispoziţie, cu un preţ acceptabil, o putere de calcul suficientă (eventual dedicată pentru taskul de antrenare). Probabil că într-un viitor destul de apropiat. Tot ca o estimare rezultată în urma antrenării reţelelor, ordinul de mărime actualmente fezabil pentru aplicarea metodologiei este de 10^{16}, în condiţiile unei dotări modeste (staţii de lucru la nivelul anului 1995). Tot din considerente computaţionale, metoda distribuirii nivelului conexionist este ad-hoc şi a fost necesară pentru a se realiza experimentele

practice, imposibile în cazul unei reţele de mari dimensiuni.

C2. Contribuţii originale

Cea mai elocventă trecere în revistă a contribuţiilor originale este o lectură a celor 9 articole publicate în perioada 1995-1997, care sunt trecute în bibliografia tezei. Alte 5 lucrări prezente în bibliografie, din perioada imediat premergătoare (1993-1994) prezintă deasemenea idei originale, dar nu sunt legate de conţinutul tezei decât prin unele metodologii folosite, având un aspect mult mai aplicativ. Trei din lucrări [Szirbik&al95], [Szirbik95b] [Szirbik97] sunt publicate în străinătate şi au fost recenzate conform unei proceduri stricte, impuse de editurile respective. Recenziile indică o notă bună pentru factorul de originalitate, fapt semnalat de toţi referenţii acestor lucrări.

Ideea în sine, de a folosi reţele neurodinamice cu antrenare supervizată pentru raţionamente simbolice robuste, este o idee totalmente originală şi nu am găsit-o în literatura de specialitate până la nivelul anului 1997. Din discuţile personale avute cu Prof. Scott Fahlman, de la Carnegie-Mellon University (Pittsburgh - Pennsylvania), unul din iniţiatorii domeniului integrării neuro-simbolice în anii 80, am realizat că drumul pe care am mers a fost unic, fără precedent în istoria (foarte scurtă dealtfel) a domeniului. Din alte discuţii, cu Prof. Lee Giles de la Princeton University şi Assoc. Prof. Riisto Miikulainen de la University of Texas (Austin), am aflat că au fost o serie de tentative nepublicate de a folosi reţele recurente cu reprezentarea locală a simbolurilor, dar s-a reuşit doar simularea unor automate finite (cu un cost prohibitiv însă).

Pe capitole, mai pot fi evidenţiate unele aspecte de originalitate:

• în capitolul de introducere, se face o taxonomizare a sistemelor si modelelor folosite în ultimii ani în domeniul integrării simbolic-conexioniste. Deşi informaţia primară este culeasă din volumele unor conferinţe indicate în bibliografie, modul în care am realizat clasificarea sistemelor hibride, este personal şi diferit de cel propus de Melanie Hilario in 1996 [Hilario96].

• în capitolul 1, prezentarea unei reţele Towell ca o reţea neurodinamică fără recurenţă, cu o posibilă traiectorie trasabilă în spaţiul stărilor dată de parcurgerea paşilor de raţionament succesiv de straturile reţelei.

• în capitolul 2: demonstrarea formulei de antrenare pentru reţele neurodinamice continue, este făcută prin calcul matriceal şi nu integral (deşi în acest mod demonstraţia este mai directă). Avantajul este apariţia în clar a calculului matricei inverse, care ilustrează complexitatea task-ului de antrenare. La reţeaua discretă, o contribuţie personală este

adăugarea pe stratul de intrare a unor unităţi speciale de "fan-out", doar pentru intrare, aceasta de obicei fiind furnizată direct unor unităţi din stratul de calcul (fără a mai fi ponderată). Avantajul a fost o antrenare mai rapidă în prima fază (de la o eroare de 25% la 10%), iar explicaţia este dată de reprezentarea mai facilă a unor dependenţe liniare între intrare şi ieşire prin aceste ponderi suplimentare.

• în capitolul 3: asocierea grafurilor conceptuale cu reprezentări conexioniste distribuite; metoda de translatare tensorială.

• începând cu capitolul 4, abordarea mea este diferită de toate abordările anterioare sau curente. Practic, toate ideile ilustrate sunt originale. În plus, metoda de paralelizare a algoritmului de antrenare, înregistrată ca şi raport de cercetare la Vrije Universiteit din Amsterdam [Szirbik95a], este originală, contrazicând unele afirmaţii că accelerarea antrenării neuronale pe sisteme distribuite cu cuplare redusă (gen ZOO) este neperformantă.

C3. Posibile direcţii de continuare a cercetării

Există două stadii pentru continuarea aceste cercetări. Primul este imediat şi se referă pe de o parte la identificarea unor metode de automatizare a procesului de identificare a traiectoriilor de antrenare necesare pentru nivelul conexionist al unui sistem compoziţional hibrid şi pe de altă parte, la automatizarea procesului de creare şi antrenare a unor module în funcţie natura traiectoriilor detectate.

Dacă primul stadiu se referă mai mult la un proces de implementare al unor activităţi relativ bine determinate şi are un caracter mai redus de cercetare, al doilea se referă la găsirea unor modele îmbunătăţite pentru nivelul conexionst. Am afirmat în capitolul 6, că este ideal să folosim o singură reţea neurodinamică şi că împărţirea pe module este artificială şi dictată de considerente computaţionale. O idee posibilă este transformarea setului de module bazate pe reţele recurente într-un singur tot, prin unirea reţelelor prin mănunchiuri de comunicare conexioniste (legături ponderate), reduse ca dimensiune, dar fără o semnificaţie simbolică. Ideea unei astfel de reţele (fără ca aceasta să aibă o aplicaţie imediată în procesarea simbolică) este întâlnită deja în literatură [Fahlman97], dar încă nu există algoritmi de antrenare propuşi care să aibă performanţele de timp dorite. Deasemenea, pentru o astfel de topologie se poate încerca şi utilizarea unor algoritmi de antrenare nesupervizaţi şi aceasta este momentan calea pe care îmi voi continua cercetarea.

Bibliografie

Ajjanagadde V., 1994, "Unclear Distinctions Lead to Unnecessary Shortcommings: Examining the role vs filler, rule vs fact and type vs predicate distinctions from a connectionist reprezentation and reasoning perspective", Proc. of the Twelfth National Conference on Artificial Intelligence (AAAI-94).

Allen J.F., 1987, Natural Language Understanding, capitol citat: p324-333, Benjamin-Cummings, Menlo Park, CA.

Alexandre F., 1996, "Neural Computing and Artificial Intelligence" în Connectionist-Symbolic Integration: From Unified to Hybrid Approaches (editori R. Sun şi F. Alexandre), p1-26, CSCSI Print, Montreal.

Alonso-Betanzos A., Guijaro-Berdiñas B., Moret-Bonillo V., Lopez-Gonzales S., 1995, "The NST-EXPERT project: The Need to Evolve", Artificial Intelligence in Medicine, vol.7, no.4, p297-314, Elsevier New-York.

Bal H.E., 1989, "The Shared Data Model as a Programming Language Paradigm for Parallel and Distributed Applications", PhD Thesis, Faculteit der Viskunde en Informatika, Vrije Universiteit, Amsterdam, Olanda.

Becraft W.R., Lee P.L., Newell R.B., 1991, "Integration of Neural Networks and Expert Systems" în Proc. of IJCAI'91, p832-837, Morgan-Kaufmann, San Mateo, CA.

Beiu V., 1996, "Digital integrated circuit implementations - Chapter E1.4" în Handbook of Neural Computation (editori, E.Fiesler şi R.Beale), Oxford University Press, NY.

Bookman L.A., 1987, "A Microfeature Based Scheme for Modelling Semantics" în Proc. of IJCAI'87, Morgan Kaufmann, San Mateo, CA.

Chang C.C., Kreisler H.J., 1973, Model theory, North Holland.

Clark A., 1989, "Connectionism and the multiplicity of mind", Artificial Intelligence Review 3/89, p49-65.

Cohen P.R., 1983, Techniques for reasoning using context, self-reflection and partitioning, teză de doctorat, Stanford University, CA, re-editată de Pitman Publishers, Londra, în 1985.

Craven M. W., Shavlik J. W., 1994, "Using Sampling and Querries to Extract Rules from Trained Neural Networks", în Machine Learning: Proceedings of the Eleventh International Conference (editori W. W. Cohen şi H. Hirsch), Morgan Kaufman, San Mateo, CA.

Crevier D., 1993, "The Tumultuous History of the Search for Artificial Intelligence, Basic Books.

Cristea L.L., **Szirbik N.**, Holban S., 1997, "Some Complexity Issues Concerning a Nonlinear Dynamical System", în Proc. of Intl. Conference on Analysis and Numerical Computation of Solutions of Nonlinear Systems (Timişoara), p343-354, Editura Universităţii de Vest.

Crucianu M., Memmi D., 1992, "Implicit Structure Extraction with Connectionist Networks", Proc. of NEuro'92 (Nanterre, Franţa), p491-502.

DeFelipe J., 1997, "Microcircuits in the Brain" în Biological and Artificial Computation: From Neuroscience to Technology (editori, J.Mira, R.Moreno-Diaz, J.Cabestany), p1-14, Springer, Berlin.

Dietterch T.G., Flann N.S., 1995, "Explanation-based Learning and Reinforcement Learning: An Unified View" în Proc. of 12th International Conference on Machine Learning", p176-184, Tahoe City, CA, Morgan Kaufmann.

Dragomirescu M., **Szirbik N.**, Dragomirescu C., Babii S., 1993a, "An Expert System for Infectious Diseases Research", Proc. of CSCS-93, vol.2, p321-325.

Dragomirescu M., **Szirbik N.**, Dragomirescu C., Kerekes K., Andriuţă A., Sălăgean L., Băbălău A., 1993b, "Sistem expert pentru modelarea procesului infecţios", Conferinţa Naţională de Informatică medicală (Timişoara, nov.93), p10-18.

Dragomirescu M., **Szirbik N.**, Dragomirescu C., Sălăgean L., 1994, "Modelarea procesului infecţios cu ajutorul reţelelor neuronale" în Interdisciplinaritatea medicinii interne (editor I.Romoşan), p188-198, Editura Helicon-Banat, Timişaoara.

Doya K., 1993, "Bifurcation of Recurrent Neural Networks in Gradient Descent Learning, TR-10-93, Department of Biology, University of California at San Diego, La Jolla, CA.

Dyer M.G., 1991, "Symbolic Neuro-Engineering for Natural Language Processing" în Advances in Connectionist and Neural Computation Theory (editori, J. Barnden şi J. Pollack), Ablex Publishing, New-York, NY.

Edelman G., 1992, On the Matter of the Mind, Basic Books.

Elman J.L., 1990, "Finding Structure in Time", Cognitive Science 14 (2/1990).

Fahlman S.E., 1988, "An Empirical Study of Learning Speed in Backpropagation Networks", TR-CMU-CS-88-162, Carnegie-Mellon University, Pittsburgh, PA.

Fahlman S.E., 1989, "Faster-Learning Variations on Back-Propagation: QuickPropagation", TR-CMU-CS-89-45, Carnegie-Mellon University, Pittsburgh, PA.

Fahlman S.E., Lebiere C., 1991, "The Cascade-Correlation Architecture", TR-CMU-CS-91-100, Carnegie-Mellon University, Pittsburgh, PA.

Fahlman S.E., 1997, "Training Boltzmann Neural Networks Using Symbolic Features", acceptată la NIPS'97, vezi la http://www.cs.cmu.edu/~sef.

Farhat N.H., 1997, "Dynamic Neural Networks for Cognition and Control", Proc. of NEURAP'97 (Marsilia, Franţa), p1-22.

Feldman J., Ballard D., 1982, "Connectionist Models and Their Properties", Cognitive Science, 6/82, p205-254.

Fodor J.A., Pylyshyn Z.W., 1988, "Connectionism and Cognitive Architectures: A critical Analysis", Cognition 28, p3-72.

Fu L.M., Fu L.C., 1990, "Mapping Rule-Based Systems into Neural Architectures", Knowledge-Based Systems, 3/90, p48-56.

Gallant S.I., 1988, "Connectionst Expert Systems", Communications of the ACM, 2/88 (no.31).

Giacometti A., 1992, Hybrid Expert Systems, PhD Thesis, Ecole Nationale Supérioure des Télécommunications, Paris.

Giles C.L., Miller C.B., Chen D., Sun G.Z., Chen H.H., Lee Y.C., 1992, "Extracting and Learning an Unknown Grammar with Recurrent Neural Networks", în Advances in Neural Information Processing Systems 4, (editori J. E. Moody, S. J. Hanson şi R. P. Lippmann), Morgan Kaufman, San Mateo, CA.

Giles C.L., Chen D., Sun G.Z., Chen H.H., Lee Y.C., Goudreau M.W., 1995, "Constructive Learning of Neural Networks", IEEE Trans. on Neural Networks, vol.6, no.4, p829-836.

Ginsberg A., 1990, "Theory Reduction, Theory Revision, and Retranslation" în Proc. of National Conference on Artificial Intelligence 1990, p777-782, U.S.A.

Giunchiglia F., Weyhrauch R.W., 1988, "Meta-level Architectures and Multi-context Axiomatization" în Reasoning, Reflection and Non-monotonicity (editori, P.Maes şi D.Nardi), p271-295, Elsevier, New-York.

Goodman R.K., Miller J.W., Smith P., 1989, "An Information Theoretic Approach to Rule-Based Connectionist Expert Systems" în Advances in Neural Information Processing 1 (editor D.Touretzky), p256-263, Morgan Kaufmann, San Mateo, CA.

Gutknecht M., Pfeifer R., Stolze M., 1991, "Cooperative Hybrid Systems" în Proc. of IJCAI'91, p824-829.

Handelman D.A., Lane S.H., Gelfand J.J., 1989, "Integrating Knowledge-based System and Neural Network Techniques for Robotic Skill Aquisition" în Proj. of IJCAI'89, p193-198, Morgan Kaufmann, San Mateo, CA.

Haykin S., 1994, Neural Networks: A Comprehensive Foundation, Macmillan College Publishing Comp, New-York, NY.

Hayes S., Ciesielski V.B., Kelly W., 1992, "A Comparision of an Expert System with a NN for Respiratory System Monitory", TR92/1, Royal Melbourne Institute of Technology.

Hecht-Nielsen R., 1990, "Neurocomputing", Addison-Wesley, New-York.

Hendler J.A., 1989, "Problem Solving and Reasoning: A Connectionist Perspective" în Connectionism in Perspective (editori R.Pfeifer, Z.Schreter şi F.Fogelman-Soulié), p229-243.

Hilario M., 1996, "An Overview of Neurosymbolic Integration Process" în Connectionist-Symbolic Integration: From Unified to Hybrid Approaches (editori R. Sun şi F. Alexandre), CSCSI, Montreal.

Honavar V., 1994, "Symbolic AI and Numeric ANNs: Toward a Resolution of Dichotomy", TR94-14(CS), Iowa State University.

Hölldobler S., Kurfess F., 1991, "CHCL - a Connectionist inference System" în Parallelization in Inference Systems (editori, B. Fronhofer şi G. Wrightson), Springer-Verlag, Berlin.

Ienne, P., 1997, "Digital Connectionist Hardware: Current Problems and Future Challenges", în Proc. of IWANN'97 (Lanzarote, Spania), p688-713, Springer, Berlin.

Jabri M., Picard S., Leong P., Chi Z., Flower B., Xie Y., 1992, "ANN Based Classification for Heart Defibrilators" în Advances in Neural Information Processing 4, p637-644, Morgan-Kaufmann, San Mateo, CA.

Kay J., Phillips W. A., 1994, "Activation Functions, Computational Goals and Learning Rules for Local Processors with Contextual Guidance", Centre for Cognitive an Computational Neuroscience, University of Stirling, Technical Report CCCN-15.

Khosla R., Dillon T., 1996, "Task Structure and Computational Level: Architectural Issues in Symbolic-Connectionist Integration" în Connectionist-Symbolic Integration: From Unified to Hybrid Approaches (editori R. Sun şi F. Alexandre), CSCSI, Montreal.

Kibler D., Langley P., 1988, "Machine Learning as an Experimental Science" în Proc. of Third European Working Session on Learning, p1-12, Edinburgh, UK.

Kokinov B., 1996, "Micro-level Hybridization in DUAL" în Connectionist-Symbolic Integration: From Unified to Hybrid Approaches (editori R. Sun şi F. Alexandre), CSCSI, Montreal.

Koppel M., Feldman R., Segre A. H., 1994, "Bias-Driven Revision of Logical Domain Theories", Journal of Artificial Intelligence Research 1/94, p159-208.

Hertz J., Krogh A., Palmer R.G., 1991, "Introduction to the Theory of Neural Computation", Addison-Wesley, Reading, MA.

Krovvidy S., Wee W.G., "An Intelligent Hybrid System for Wastewater Treatment" în Hybrdid Architectures of Intelligent Systems, cap.17, p358-377, CRC Press, Boca Raton, CA.

Kunicky D.C., Hruska S.I., Lacher R.C., 1992, "Hybrid Systems: The Equivalence of Rule-Based Expert Systems and Artificial Neural Networks Inference", International Journal of Expert Systems, 4/92, p281-297.

Labbi A., 1993 "Neural Networks for Decision Making in Dynamic Environments" în Proc. of ICANN'93, Amsterdam, Springer-Verlag, Berlin.

Lendaris G.G., 1992, "A Neural Network Approach for Implementing Conceptual Graphs" în Conceptual Structures: Current Research and Practice, (editor T.E.Nagle) p165-177, Ellis Horwood Inc., Chichester, UK.

Leonard J.J., Durrant-Whyte H.F., 1992, Mobile Robot Navigation Using Neural Nets, Kluwer Academic Publishers, Boston, MA.

Leung K.S., Lam W., 1988, "Fuzzy Concepts in Expert Systems", IEEE Expert, 3/88, p43-56.

Luger S., Stubblefield J., 1989, "Artificial Intelligence and the Design of Expert Systems", Benjamin/Cummings, New-York.

MacKay D., 1992, "A Practical Bayesian Framework for Backpropagation Networks", Neural Computation 4/92, p448-472.

Maclin R., Shavlik J. W., 1994, "Incorporating Advice into Agents that Learn from Reinforcements", Proceedings of the Twelfth International Conference on Artificial Intelligence (AAAI-94).

McClelland J.L., O'Reilly R.C., 1994, "Brain Conjunctive Encoding, Storage and Recall: Avoiding a Tradeoff", TR-PDP-CNS-94-4, Carnegie-Mellon University, Pittsburgh, PA.

McCulloch W.S., 1988, Embodiments of Mind, MIT Press, Cambridge, MA.

Memmi D., 1990, "Connectionism and Artificial Intelligence as Cognitive Models", AI&Society 4, p115-136.

Miikulainen R., 1991, "Script Recognition with Hierarchical Feature Maps", TR-UCLA-AI-91-10, University of California at Los Angeles, CA.

Miikulainen R., 1994, "Integrated Connectionist Models: Building AI Systems on Subsymbolic Foundations" în AI and Neural Networks (editori V.Honavar şi L.Uhr), p483-508, Academic Press, San Diego, CA.

Miikulainen R., 1996, "Subsymbolic Case-Role Analysis of Sentences with Embbeded Clauses, TR-CS-45-96, University of Texas, Austin.

Minsky M.L., Papert S.A., 1969, Perceptrons, MIT Press, Cambridge, MA.

Minsky M.L., 1986, The Society of Mind, MIT Press, Cambridge, MA.

Minsky M., 1991, "Local versus Analogical or Symbolic versus Connectionist or Neat versus Scruffy", AI Magazine, 3/91, p7-33.

Nagayama I., Akamatsu N., 1994, "Learning of Chaos by Neural Networks and Bifurcation Phenomena", Transactions of the Insitute of Electronics, Information and Communication Engineers, vol J77A, no.11, p1593-6.

Newell A., Simon H.A., 1972, Human Problem Solving, Prentice-Hall, Engelwood Cliffs, NJ.

Newell A., 1990, Unified Theories of Cognition, Harvard University Press, Cambridge, MA.

Omlin C. W., Giles C. L., 1992, "Training Second-Order Recurrent Networks Using Hints", Machine Learning: Proceedings of the Ninth International Conference, edited by D. Sleeman and P. Edwards, Morgan Kaufman, San Mateo, CA.

Opitz D.W., Shavlik J.W., 1997, "Connectionist Theory Refinement: Genetically Searching the Space of Network Topologies", Journal of Artificial Intelligence Research 6/97, p177-209.

Orsier B., Labbi A., 1996, "NESSY3L: A NeuroSymbolic System with 3 Levels" în Connectionist-Symbolic Integration: From Unified to Hybrid Approaches (editori R. Sun şi F. Alexandre), CSCSI, Montreal.

Pazzani M., Brunk C., 1991, "Detecting and Correcting Errors in Rule-Based Expert Systems: An Integration of Empirical and Explanation-Based Learning", Knowledge Acquisition 2/91 (3), p157-173.

Pearl J., 1988, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufmann, San Mateo, CA.

Pearlmutter B.A., 1990, "Dynamic Recurrent Neural Networks", TR-CMU-CS-90-191, Carnegie-Mellon University, Pittsburgh, PA.

Pfeifer, W.T., 1996, "Symbols and Patterns: Beyond the Classical Information Processing", în Enciclopedia of Computers (editor, Marcel Dekker), vol.17, p253-275, Elsevier, New-York.

Phan K.M., 1995, "A Neural Multi-Agent Approach for Modelling, Distributed Processing and Learning" în Intelligent Hybrid Systems (editori, S.Goonatilak şi S.Kebal), John Wiley Comp., Chichester, UK.

Pineda F.J., 1987, "Generalization of Back-Propagation to Recurrent Neural Networks", Physical Review Letters 19/87, p2229-32.

Pineda F.J., 1988, "Dynamics and Architecture for Neural Computation", Journal of Complexity 4.

Pinker S., Prince A., 1988, "On Language and Connectionism: Analysis of Parallel Distributed Processing Model of Language Aquisition", Cognition 28, p73-193.

Piramuthu S., Shaw J.M., 1994, "On Using a Decision Tree as Feature Selector for Feedforward Neural Networks" în Proc. of International Symposium on Integrating Knowledge and Neural Heuristics" (Penslacola, FL), p67-74.

Quillian R., Collins J., 1969, "A Graf Based Defined Lexicon as a Model for Plausibile Representation in Human Mind", Psychological Review 80, p413-51.

Rao S.S., Ramamurti V., 1993, "A Hybrid Technique to Enhance the Performance in Recurrent neural Networks", Proc. of ICNN'93, San Francisco, cat.no.93CH3274-8.

Reeke G.N., Edelman G.M., 1988, "Real Brains and Artificial Intelligence" în The Artificial Intelligence Debate: False Starts, Real Foundations (editor S. Graubard), p144-173, MIT Press, Cambridge, MA.

Riesbeck C.K., 1986, "Realistic Natural Language Processing" în Issues in Cognitive Modelling (editori, A.Aitkenhead şi J.Slack), p193-206, Lawrence Erlbaum Assoc., London, UK.

Roberts D.D., 1992, "The Existential Graphs" în Semantic Networks in Artificial Intelligence, p639-663, (editor F.Lehman), Pergamon Press, Oxford, UK.

Robinson G., 1965, "The Resolution Principle", AI Review 4/65.

Röbel A., 1994, "The Dynamic Pattern Selection Algorithm: Effective Training and Controlled Generalization of Recurrent Backpropagation Neural Networks", Technical Report, Institut für Augewandte Informatik, Technische Universität Berlin.

Rumelhart D.E., Smolensky P., McClelland J.L., Hinton G.E., 1986, "Schemata and Sequential Thought Process in PDP Models" în Parallel Distributed Processing (vol.2): Explorations in the Microstructure of Cognition (editori, D.Rumelhart şi J.McClelland), p7-57, Bradford Books, Cambridge, MA.

Schank R.C., 1981, Inside Computer Understanding, Lawrence Erlbaum Assoc., Hillsdale, NJ.

Scherer A., Schlegeter G., 1995, "A Multi-Agent Approach for the Integration of Neural Networks and Expert Systems" în Intelligent Hybrid Systems (editori, S.Goonatilak şi S.Kebal), John Wiley Comp., Chichester, UK.

Scott G., Shavlik J., Ray W., 1992, "Refining PID Controllers Using Neural Networks", Neural Computation 5/92, p746-757.

Sharkey N. E., Reilly R., 1990, "Connectionist Natural Language Processing", Parallel Processing Specialist Group Newsletter, No. 13.

Sharkey N., 1992, "The Gost in the Hybrid: a Study of Uniquely Connectionist Representations", AISB Quarterly 79, p10-16.

Sharkey N., 1995, "Searching the Weight Space for Solutions" în Current Trends in Connectionism (editori, L.Niklasson şi M.Boden), p103-120, Lawrence Erlbaum Publishers, Hove, UK.

Shastri L., 1988, "A Connectionist Approach to Knowledge Representation and Limited Inference", Cognitive Science, 12/88, p331-392.

Shavlik J.W., 1991, "Hybrid symbolic+connectionist reasoning automata", TR-118-91, University of Wisconsin, Madison.

Shavlik J.W., 1997, "How to Close the Gap Between Neural and Symbolic Representations", Machine Learning, vol. 25 (Jan/Feb/Mar), p295-301.

Shortliffe E.H., 1976, Computer-Based Medical Consultations: MYCIN, American Elsevier, New-York, NY.

Sowa J.F., 1984, Conceptual Structures: Information Processing in Mind and Machine, Addison-Wesley, Reading, MA.

Sowa, J.F., 1995, Knowledge Representation: Logical, Philosophical and Computational Foundations, PWS Publishing Comp. Boston, MA.

Smolensky P., 1986, "Neural and Conceptual Interpretation of PDP Models" în Parallel Distributed Processing (vol.2): Explorations in the Microstructure of Cognition (editori, D.Rumelhart şi J.McClelland), p390-431, Bradford Books, Cambridge, MA.

Smolesky P., 1988, "On the Proper Treatment of Connectionism", Behavioral and Brain Sciences, 11/88, p1-74.

Smolensky P., 1990, "Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems", Artificial Intelligence 46/1990, p159-216.

Smolensky P., Legendre G., Miyata Y., 1992, "Principles for an Integrated Connectionist/Symbolic Theory of Higher Cognition", Computer Science Department, University of Colorado-Boulder, Technical Report CU-CS-600-92.

Somlo G., 1996, "Improving Learning Speed in Recurrent Neural Networks", Research Progress Report-520 (Computer Science), Colorado State University, Fort Collins, CO.

Sun R., 1991, "Integrating Rule and Connectionism for Robust Reasoning. A Connectionist Architecture with Dual Representation, PhD Thesis (TR-CS-91-160), Brandeis University, Waltham, MA.

Sun R., 1992, "A Connectionist Model for Commonsense Reasoning Incorporating Rules and Similarities", Knowledge Acquisition 4, p293-321, Academic Press, Cambridge, UK.

Sun G. Z., Giles C. L., Chen H. H., Lee Y. C., 1993, "The Neural Network Pushdown Automaton: Model, Stack and Learning Simulations", University of Maryland, Technical Report UMIACS-TR-93-77.

Szirbik N., Babii S., Dragomirescu C., 1993, "A Method for Inserting PROLOG type Rules in a Neural Network", Proc. of CSCS-93 (Bucureşti), vol.2, p202-206.

Szirbik N., 1993, "Modus Ponens and Neural Networks" în Buletinul Ştiinţific şi Tehnic al UTT, Tom 38(52), p145-149.

Szirbik N., Somlo G., Buliga D., 1995, "Using the Conceptual Graph Model as Intermediate Representation for Knowledge Translation in Hybrid Systems" în Current Trends in Connectionism (editori, L.Niklasson şi M.Boden), p141-152, Lawrence Erlbaum Publishers, Hove, UK.

Szirbik N., Babii., 1995, "Intermediate Representation in Parallel-Distributed Processing", Proc. of CSCS-95 (Bucureşti), vol.3, p243-252.

Szirbik N., 1995a, "Recurrent Neural Networks Learning on the ZOO Multiprocessor: Speed-ups and Other Results", Research Progress Report, Faculteit der Viskunde en Informatika, Vrije Universiteit, Amsterdam, Olanda.

Szirbik N., 1995b, "Towards a Model of Multi-Agent Connectionist Hybrid System" în Proc. of ANNES'95 (Dunedin, NZ), p273-6, IEEE Computer Soc.Press, Los Alamitos, CA.

Szirbik N., Somlo G., 1995, "Parallelizing the Williams&Zipser Algorithm using PVM", Lucrare de diplomă (cu rezumat extins în engleză), Catedra de calculatoare, Universitatea "Politehnica" din Timişoara.

Szirbik N., Babii S., 1996, "Modelling Connectionist Reasoning Patterns by Interacting Hybrid Reasoning Agents" în Buletinul Ştiinţific al UPT, Tom 41(55), p92-100.

Szirbik N., 1996, "NeuroSymbolic Integration: A Way Towards Building Intelligent Systems" în Proc. of CONTI'96 (Timişoara), vol.1, p135-44.

Szirbik N., 1997, "A Two-Level Heterogeneous Hybrid Model" în Proc. of IWANN'97 (Lanzarote, Spania), p644-650, Springer, Berlin.

Tannenbaum A.S., Rennese R., 1989, "Experiences with the Amoeba Operating System", Report IR-194, Faculteit der Viskunde en Informatika, Vrije Universiteit, Amsterdam, Olanda.

Thomason R.H., Touretzky D.S., 1990, "Inheritance Theory and Networks with Roles", TR-CMU-CS-90-139, Carnegie-Mellon University, Pittsburgh, PA.

Todd B.S., 1991, "Bayesian Inference Theory", TR-PRG-95, Oxford University Computing Laboratory, Oxford, UK.

Touretzky D.S., Geva S., 1987, "A Distributed Connectionist Representation for Concept Structures" în Proc. of Ninth Annual Conference of the Cognitive Science Society (Seattle,

WA).

Touretzky D.S., Hinton G.E., 1988, "A Distributed Connectionist Production System", Cognitive Science, 12/88, p423-466.

Touretzky D.S., 1989, "BoltzCONS: a Connectionist Symbolic Machine", TR-CMU-CS-89-182, Carnegie-Mellon University, Pittsburgh, PA.

Touretzky D.S., 1990, "Dynamic Symbol Structures in a Connectionist Network", Artificial Inteiligence, 46 (1-2).

Towell G. G., Shavlik J. W., 1991, "The Extraction of Refined Rules from Knowledge-Based Neural Networks", Machine Learning 8/91.

Towell G. G., 1991, "Symbolic Knowledge and Neural Networks: Insertion, Refinement and Extraction", Ph.D. Thesis at the Computer Science Department of the University of Wisconsin, Madison.

Treur J., 1988, "Design of Modular and Interactive Knowledge-based Systems", Report P8826, Faculteit der Viskunde en Informatika, Vrije Universiteit, Amsterdam, Olanda.

Treur J., 1990, "Modelling Non-classical Reasoning Patterns by Interacting Reasoning Modules", Report IR-236, Faculteit der Viskunde en Informatika, Vrije Universiteit, Amsterdam, Olanda.

Treur J., 1992, "Towards Dynamic and Temporal Semantics of Meta-Level Architectures in DESIRE", Report IR-321, Faculteit der Viskunde en Informatika, Vrije Universiteit, Amsterdam, Olanda.

Tsung F.S., Cottrell G.W., 1993, "Hopf Bifurcations and Hopf Hopping in Recurrent Neural Networks" în Proc. of The IEEE Conference on Neural Networks, vol.1, p39-45, IEEE Computer Soc.Press, New-York, NY.

Van Gelder T., 1995, "Modelling Connectionist and Otherwise" în Current Trends in Connectionism (editori, L.Niklasson şi M.Boden), p217-36, Lawrence Erlbaum Publishers, Hove, UK.

Vilain M., McAllester D., 1989, "An Overview of NIKL: Tackling the Huge Number of Assertions" în Annual Report of Research in Knowledge Representation, (editori, B.Beranek şi L.Newman), p45-79, BBN Press, Cambridge, MA.

Wang Y. Y., Waibel A., 1992, "Dialogue Processing with Neural Networks", Carnegie-Mellon University, Pittsburgh, Technical Report CMU-CS-92-165.

Watrous R., Towell G., Glassman M., 1993, "Synthesize, optimize, analyze, repeat (SOAR): Application of Neural Network Tools to ECG patient monitoring" în Proc. of Symposium on Nonlinear Theory and its Applications, p565-570, Honolulu, Hawaii.

Wermter S., Weber W., 1997, "SCREEN: Learning a Flat Syntactic and Semantic Spoken

Language Analysis Using Artificial Neural Networks", Journal of AI Research 6, 1/97, p35-85.

Wiggins S., 1990, Introduction to Applied Nonlinear Dynamical Systems, Bifurcations and Chaos, Springer-Verlag, New-York, NY.

Wilkins D.C., 1988, "Knowledge Base Refinement Using Apprenticenship Learning Techniques" în Proc. of National Conference on Artificial Intelligence 1988, p646-653.

Williams R. J., Zipser D., 1989, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks", Neural Computation, vol 1/89, p270-280.

Williams R.J., Zipser D., 1990, "Gradient Based Learning Algorithms for Recurrent Connectionist Networks", TR-NU-CCS-90-9, Northeastern University, Boston, MA.

Williams R.J., Peng J., 1990, "An Efficient Gradient Based Algorithm for On-line Training of Recurrent Network Trajectories", Neural Computation 2/90, p490-501.

Winograd T., 1980, "What Does It Mean to Understand Language ?", Artificial Intelligence 6/80, p231-263.

Notă autobiografică a autorului

Data naşterii: 31 mai, 1963
Locul naşterii: Timişoara

Clasele I-VIII: Colegiul "C.D. Loga", Timişoara, 1969-77
Liceul: Liceul de informatică "Grigore Moisil", Timişoara, 1977-1981
Stagiul militar: Rgt.56 Paraşutişti, Caracal, 1981-1982
Facultatea: Electrotehnică, Specialitatea Calculatoare, Institutul Politehnic Traian Vuia Timişoara, 1982-1987.

Inginer stagiar la: I.A.M.S.A.T. Bucureşti şi Centrul de calcul al Universităţii Tehnice din Timişoara, 1987-1990

asistent universitar la: Catedra de calculatoare a Universităţii Tehnice din Timişoara, 1991-1994

înscris la doctorat din: 1992

şef de lucrări la: Departamentul de Calculatoare al Universităţii "Politehnica" Timişoara, 1994-prezent

titular al cursurilor: Sisteme Expert (anul V), Modelul Computaţional Conexionist (Reţele Neuronale - curs pentru studenţii de la Studii aprofundate)