Een Semantisch Connectionisch Redeneersysteem (SCORE)

A.P.J. Sadon, 1990

TU Delft
Technische Universiteit Delft
Delft University of Technology

Faculteit der Technische Wiskunde en Informatica
Faculty of Technical Mathematics and Informatics

Voorwoord

Dit rapport is het resultaat van mijn afstuderen bij de deelgroep Kennisgestuurde Systemen, onder leiding van Prof. H. Koppelaar, binnen de studie Informatica aan de Technische Universiteit te Delft.

Ik heb gepoogd een wezenlijk inzicht in Connectionist Systems, ambitieus meestal 'Neurale Netwerken' genoemd, te verbinden met traditionele AI benaderingen. Ik ben daarom de uitdaging aangegaan meer inzicht te krijgen in min of meer filosofische achtergronden van deze nieuwe AI ontwikkeling, om van daaruit met de techniek bezig te zijn. Binnen het afstuderen heb ik gemerkt dat deze manier van techniek bedrijven mij veruit de meeste voldoening geeft.

Dit rapport wordt daarom begonnen met een zoveel mogelijk van de techniek geabstraheerde beschouwing van redeneersystemen in het algemeen en Connectionist Systems in het bijzonder, waarna langzaam maar zeker wordt toegewerkt naar een zuiver technisch interpretatie ervan. Ik ben tevreden over het resultaat, temeer omdat het bij mij een proces in werking heeft gezet waar ik denk nog lange tijd plezier van te kunnen hebben. Ik hoop dat die vonk, middels het onderhavige rapport, op de lezer zal overslaan.

Ik wil mijn afstudeercommissie, bestaande uit de heren E. Frietman, E. Kerckhoffs, A. Klaassen en H. Koppelaar, hartelijk bedanken voor het feit dat deze opdracht binnen de groep mogelijk was. In het bijzonder dank ik dhr. Kerckhoffs voor zijn enthousiasme en directe begeleiding en dhr. Klaassen voor zijn spontane medewerking en inspirerende adviezen.

Delft, juli 1990 A.P.J. Sadon

Tussenwoord

Onderhavig document is een bewerkt document van de oorspronkelijke scriptie. Dit behoeft enige toelichting.

Nadat de in het voorwoord genoemde begeleiders mij in 1990 een prachtige 9 hadden gegeven voor het afstudeerproject, ben ik van Delft verhuisd naar Amsterdam. Daar heb ik een aantal jaren gewerkt voor het bedrijf AI-Engineering, waar ik mijn inzichten in het fenomeen van de Neurale Netwerken op verschillende gebieden heb toegepast. Het betrof onder andere de toepassing van Neurale Netwerken binnen de optimalisatie van TV-commercials op basis van kijkersgegevens en de optimalisatie van de uitwisseling van data binnen computernetwerken. Daarna is het gebied van de Neurale Netwerken langzaam maar zeker uit mijn blikveld verdwenen geraakt, en ben ik voor verschillende bedrijven als softwareontwikkelaar actief geweest. Vanaf 2004 raakte ik gefascineerd door de zoekmachine van Google en begreep ik dat het voor bedrijven zeer interessant kon zijn om kennis te krijgen van het algoritme dat de zoekresultaten ordent. Sindsdien heb ik vanuit mijn eenmanszaak onder de naam SEOguru boeken geschreven, bedrijven geadviseerd en vele trainingen gegeven op het gebied van de zoekmachine optimalisatie (Search Engine Optimization – SEO).

Ik leek dus behoorlijk ver afgedreven van mijn oorspronkelijke afstudeergebied. Maar wat blijkt de laatste jaren: het Google-algoritme maakt in toenemende mate gebruik van Neurale Netwerk technologieën. Nadat het onderzoek op het gebied van Artificiële Intelligentie in het algemeen en Neurale Netwerken in het bijzonder niet alleen bij mij maar ook wereldwijd nauwelijks nog vorderingen maakte, is het gebied de laatste vijf jaar opeens weer springlevend geworden. Dit komt met name door het feit dat computers zoveel sneller zijn geworden, en ook de opslagmogelijkheden zo enorm zijn gegroeid. Google is inmiddels één van de koplopers op het gebied van onderzoek en toepassing van –wat ze noemen- 'Machine Learning', of concreter 'Deep Learning', dat een uitwerking betreft van de Neurale Netwerk theorieën. In relatie tot hun rankingalgoritme hanteert Google de term: 'RankBrain'. Ook partijen als IBM, Microsoft en Apple zijn volop met dit vakgebied bezig, voornamelijk met het oog op beeld-, spraak- en patroonherkenning.

Door deze ontwikkeling werd ik als het ware 'gedwongen' mij opnieuw te verdiepen in de huidige technologische stand van zaken op dat gebied. En natuurlijk moet je dan beginnen waar je gebleven was, o.a. bij mijn afstudeerscriptie, waar ik nog precies één geprint exemplaar van bleek te hebben. Dat er geen elektronische versie van de scriptie bestond (ook op de TU Delft bleek geen exemplaar meer aanwezig) doet een SEO-deskundige pijn, want Google kan fysieke boekenkasten (vooralsnog) niet lezen. Ik besloot daarom alle pagina's van mijn enige exemplaar te scannen en door Googledocs te laten vertalen naar concrete karakters. Dat blijkt al heel goed mogelijk. Het uiteindelijke resultaat is echter een rommeltje

en vraagt om nauwkeurige inspectie en aanpassing. Ook afbeeldingen verdwijnen bij zo'n scan: die moeten afzonderlijk worden toegevoegd. En dan waren er ook nog de vele wiskundige vergelijkingen en formules, die helemaal opnieuw moesten worden ingevoerd. Ik werd gegrepen door het compleet elektroniseren van de scriptie, waarbij ik mij continu afvroeg waarom ik dat in godsnaam aan het doen was. Want wie, behalve de zoekrobot van Google, is geïnteresseerd in een onderzoek uit 1990? Ik kon het voor mijzelf enigszins legitimeren door het feit dat het een goede manier was om mijn kennis over Neurale Netwerken te revitaliseren. Ik zag ook dat ik er destijds met veel enthousiasme en creativiteit mee bezig ben geweest. Dat sloeg over op mijn huidige zelf. Inmiddels ben ik gedreven aan het studeren op de modernste ontwikkelingen en inzichten in het vakgebied, via artikelen van Google e.a., via YouTube-colleges afkomstig van universiteiten wereldwijd, etc. Ongelofelijk eigenlijk, hoe schitterend je tegenwoordig kennis kan vergaren. Deze studie zal leiden tot een nieuw boekje van mijn hand, met een moderne introductie op het gebied van de Deep Learning. Veel van wat in deze scriptie is beschreven blijkt daarbij eigenlijk nog verrassend bruikbaar.

Amsterdam, 2017 Ir. A.P.J. Sadon

Samenvatting

Eén van de nieuwste ontwikkelingen binnen het Artificiële Intelligentie-onderzoek (AI-onderzoek) is die van de Neurale Netwerken. Vanwege onvoldoende kennis over de werkelijke neurale principes worden ze, vanwege hun structuur, ook wel Connectionist Systems genoemd. Deze AI-benadering wijkt radicaal af van de traditionele AI-aanpak. Modellen die aan deze systemen ten grondslag liggen vallen onder het 'Connectionist AI-Paradigma'. Daar tegenover staat het 'Traditionele AI-Paradigma', welke alle AI-benaderingen bevat die niet onder het Connectionist AI-Paradigma vallen.

Een ander en ouder deelgebied van het AI-onderzoek betreft het expertsysteem-onderzoek, alwaar men de kennis van een expert tracht te isoleren, vervolgens te formaliseren om het uiteindelijk te kunnen reproduceren.

Binnen de traditionele expertsystemen wordt kennis gedefinieerd in termen van regels (of frames o.i.d., maar dat zijn in feite ook regels, echter meer gestructureerd). Een regel is een uitspraak in termen van symbolen. Een symbool is bijvoorbeeld de windrichting voor een meteorologisch expertsysteem of een pixel voor een patroonherkenningssysteem. Door deze verschillende 'expertsymbolen' d.m.v. regels aan elkaar te relateren ontstaat kennis. Het idee 'hoe meer regels, hoe meer en genuanceerder de kennis', is de grondgedachte achter het traditionele AI-paradigma.

Bij Connectionist Systems vindt de probleembeschrijving op een lager niveau plaats. Dit wordt het subsymbolisch niveau genoemd. Aldaar wordt in termen van een groot aantal, 'zwakke' regels het probleem gedefinieerd. Deze regels zijn, in tegenstelling tot de regels binnen het traditioneel AI-Paradigma, *overrulebaar*: de regels worden geïnterpreteerd als adviezen en niet als autoritaire uitspraken. Het doel van Connectionistisch inferentiemechanisme is, na een waarneming, dié oplossing te vinden waarbij zoveel mogelijk adviezen worden gehonoreerd. In het geval van inconsistenties, onvolledigheden en onzekerheden wordt dus, zonder extra aanpassingen, óok automatisch gezocht naar een zo goed mogelijke oplossing.

Binnen het Connectionist AI-Paradigma is inferentie een iteratief proces waarbij de subsymbolen voortdurend van waarde veranderen, totdat er een evenwichtssituatie is bereikt die 'zo goed mogelijk' bij de waarneming aansluit. De instellingen van de subsymbolen representeren dan de oplossing van het probleem. Vele subsymboolinstellingen worden hiermee impliciet uitgesloten omdat die geen evenwichtssituatie opleveren. Groepen van subsymbolen worden vervolgens geïnterpreteerd als de symbolen van het systeem, waardoor ook vele symboolcombinaties worden uitgesloten. Op het subsymbolisch niveau vindt dus een zeer genuanceerd, flexibel proces plaats wat resulteert in even genuanceerde en flexibele conclusies op symboolniveau.

Connectionist Systems zijn intrinsiek parallel doordat ieder subsymbool slechts informatie nodig heeft van dié subsymbolen waar het van afhankelijk is en het vervolgens zélf zijn conclusie kan trekken. Zonder dat er een alles overkoepelende besturing boven het systeem 'hangt' streeft ieder subsymbool op eigen wijze naar zijn aandeel in de oplossing: de subsymbolen zijn, in tegenstelling tot de symbolen binnen het traditionele AI-Paradigma, actieve objecten.

Deze 'holistische' benadering heeft de basis gevormd voor het in dit rapport besproken model SCORE, dat staat voor “Semantisch COnnectionistisch REdeneersysteem". Het resultaat is met opzet geen expertsysteem maar een redeneersysteem genoemd, om niet de indruk te wekken dat het in deze vorm meteen al commercieel toepasbaar is. SCORE kan de basis vormen voor een volwaardig expertsysteem. De, voor een expertsysteem van elementair belang zijnde, fundamentele inferentie principes, kennisacquisitie principes en kennisrepresentatievormen hebben in SCORE de meeste aandacht gehad.

In SCORE is het Subsymbolisch- en het Symbolisch Paradigma met elkaar vermengd. De bovenbeschreven connectionistische methode van inferentie is er in verwerkt, waarbij de expertsymbolen als de subsymbolen van het Connectionist System worden beschouwd. Na een waarneming wordt het inferentieproces automatisch aangevangen doordat de subsymbolen hierdoor uit evenwicht worden gebracht. Alle subsymbolen gaan nu actief op zoek naar hun aandeel in de oplossing. Een stabiele situatie representeert voor ieder subsymbool (dus het expertsymbool), een waarde die zo goed mogelijk bij de waarneming aansluit. Deze set van waarden wordt beschouwd als de (of een) oplossing van het probleem. SCORE is zó opgezet dat stabiele situaties zo goed mogelijk overeenkomen met legale patronen (in dit rapport legale afleidingen genoemd), d.w.z. symboolcombinaties zoals de expert die aan het netwerk heeft geleerd.

Logisch, door de expert gedefinieerde, bij elkaar behorende expertsymbolen (bijvoorbeeld de symbolen kleur[groen]; kleur[rood]; kleur[geel]) worden units genoemd en staan onder de hoede van sequentiële unitbesturingen. De unit is in SCORE het kleinst ondeelbare element en kan in het geval van een parallelle implementatie nog worden toegekend aan een processor. De unit is ingewikkelder dan het subsymbool uit het Connectionist AI-Paradigma.

Voordat inferentie kan plaatsvinden moet de expert, zoveel mogelijk rekening houdend met duidelijk gedefinieerde semantische eisen, een kennisstructuur, d.w.z. een structuur tussen de verschillende units, definiëren. De structuur heeft invloed op de performance van het systeem, maar is niet van cruciaal belang. Het is bedoeld om het aantal relaties tussen de subsymbolen in te perken. Vervolgens moet het netwerk worden getraind met voorbeelden. Deze voorbeelden worden de legale afleidingen. Aan de hand van deze voorbeelden bepaalt SCORE of er semantische eisen worden geschonden. Indien dat het geval is, worden er automatisch (voor de gebruiker onzichtbaar) dan wel extra units, dan wel extra subsymbolen

geïntroduceerd. SCORE zorgt er automatisch voor dat de kennis voortdurend consistent blijft. Van een inconsistente kennisstructuur mag men immers niet verwachten dat het consistent kan redeneren. Om de betekenis van de relaties tussen de subsymbolen niet te verliezen is de kanstheorie zover als mogelijk in SCORE verwerkt.

Het inferentiemechanisme is gericht op het, uitgaande van de waarneming, vinden van een bijbehorend trainingsvoorbeeld. Als er meerdere voldoen, of als er geen enkele voldoet, dan moet een zo goed mogelijke worden gevonden. Om met dat zoekproces niet in lokale optima terecht te komen, is 'Simulated Annealing', een afgeleide van de tweede hoofdwet van de thermodynamica, toegepast.

Aan het geheel is een kennisacquisitiemodule gekoppeld waarmee inferentieresultaten in de verbindingen van het netwerk kunnen worden verwerkt. Hiermee kan het op de praktijk worden afgestemd.

Inhoud

1 Inleiding

Met het oog op het feit dat we pretenderen met AI bezig te zijn, is het om te beginnen aardig een aantal AI-definities te poneren. Er zijn er meer dan één opgenomen vanwege de verwarring in de literatuur omtrent dat begrip. Deze verwarring zegt iets over de onduidelijkheid van het onderzoeksgebied: er is weinig consensus over de benadering ervan.

AI wordt door Lucas en van der Gaag [Lucas, 1988], onderzoekers binnen het Centrum voor Wiskunde en Informatica in Amsterdam, omschreven als:

> *"het deelgebied van de informatica dat zich bezighoudt met de ontwikkeling van computerprogramma's die resultaten produceren waarvoor men menselijk gedrag nodig zou achten"*

De AI-definitie van E. Rich [Rich, 1983], luidt als volgt:

> *"AI is de studie over hoe computers te ontwerpen die dingen kunnen die, op het moment, mensen beter doen"*

Een derde AI-definitie die het meest interessant is (om redenen die later in dit rapport duidelijk zullen worden) , komt van D. Hofstadter [Hofstadter, 1988], schrijver van het alom geroemde boek 'Gödel, Escher, Bach: Een eeuwige gouden band'.

> *"AI is het zoeken naar manieren om computers zo te programmeren dat hun gedrag eens flexibiliteit, gezond verstand, inzicht, creativiteit, zelfbewustzijn, humor enzovoort zal vertonen"*

De drang naar het verkrijgen van meer kennis over de mens, in het bijzonder kennis over het denken van de mens, vindt al plaats vanaf de oudheid. Vele filosofen hebben reeds hun licht over dit probleem laten schijnen. De eeuw waarin we nu leven biedt ons opeens een hulpmiddel tot reflectie van ideeën die tot voor kort slechts op het speculatieve vlak konden plaats vinden: de computer. Bovenstaande AI-definities zijn definities van onze nieuwe mogelijkheden. De AI-definities nader bekeken blijken even vaag te zijn, misschien juist daarom, als ons huidig begrip van intelligentie. Hoewel men Artificiële Intelligentie relateert aan menselijke intelligentie is men er nog lang niet uit wat menselijke intelligentie precies betekent.

Wetenschappers zijn veelal te snel geneigd 'intelligentie' toe te schrijven aan iets wat binnen een klein probleemdomein handelingen verricht die een mens in principe ook zou hebben kunnen verrichten. In dit rapport zal worden beargumenteerd waarom we binnen het traditionele AI-onderzoek duidelijk nog niet met intelligentie bezig zijn. Een alternatief, waar dit afstudeeronderzoek ook op gebaseerd is, zal er uitgebreid tegenover worden gezet. Ook hierbij mag men zeker nog niet spreken van intelligentie, maar door de gebruikte principes staat het waarschijnlijk dichter bij menselijke intelligentie dan traditionele AI-systemen dat staan.

Maar meer nog dan alleen het feit dat aan de huidige generatie AI-machines ten onrechte 'intelligentie' wordt toegeschreven, ligt er omgekeerd het probleem dat juist vanuit dat begrip 'intelligentie' zo'n enorme aantrekkingskracht uitgaat. 'Hoe intelligentie zich aan ons aandient', heeft lange tijd het karakter van het AI-onderzoek bepaald. En omdat die buitenkant van onze intelligentie zo grijpbaar lijkt, dacht men snel resultaat te kunnen boeken: in de jaren '70 ontstond binnen het AI-onderzoek een deelgebied, Knowledge Based Systems genoemd, waarvan verwacht werd dat het commercieel interessant zou kunnen gaan worden. Knowledge Based Systems zijn "informatiesystemen waarin het gebruik van menselijke kennis, veelal op een wijze die doet denken aan het menselijk redeneren, een belangrijke rol speelt" [Lucas, 1988].

In de eerste benadering van Knowledge Based Systems werd voor ieder probleem een nieuw systeem ontwikkeld. Later werd duidelijk dat het ontwikkelen van op kennis gebaseerde systemen anders aangepakt kon worden. Deze tweede benadering van Knowledge Based Systems resulteerde in flexibelere kennissystemen: de 'expertsystemen'. Men begon namelijk in te zien dat er onderscheid gemaakt kon worden tussen kennis en inferentie. Expertsystemen zijn dan ook zo opgezet dat een soort 'general purpose' inferentiemechanisme (of redeneermechanisme) losgelaten kan worden op een variabele kennisbank, de rulebase genoemd. Van deze doorbraak op het gebied van kennissystemen werd veel verwacht: grote groepen onderzoekers konden zich vanaf dat moment heel gericht bezig gaan houden met enerzijds onderzoek naar de verschillende mogelijkheden van kennisrepresentatie en anderzijds met het bedenken van verschillende typen inferentiemechanismen. Alle modellen die ten grondslag liggen aan traditionele AI-benaderingen vallen onder het 'Traditionele AI-Paradigma'.

Door de verrassende resultaten van parallelle systemen maar ook door wat genoemd wordt de 'AI-winter' (waarin de beperkingen van de traditionele AI duidelijk werd), en vooral door de toenemende kennis omtrent de werking van de menselijke hersenen werden ideeën ontwikkeld voor een fundamenteel nieuwe kijk op inferentie en kennisrepresentatie gevat in, wat in de volksmond genoemd wordt, 'Neurale Netwerken'. Neurale Netwerken zijn implementaties van modellen van ons brein, op het niveau van de neuronen. Aangezien de huidige modellen nog steeds benaderingen zijn van wat er zich werkelijk in ons brein afspeelt,

wordt de benaming 'Neuraal Netwerk' in dit rapport niet meer gebruikt. Hiervoor in de plaats wordt, doelend op de structuur van het netwerk, gesproken van 'Connectionist Systems'. (Hoewel deze benaming sinds kort binnen de Theoretische Psychologie ook al weer voor iets heel specifieks schijnt te zijn gereserveerd). Modellen die ten grondslag liggen aan Connectionist Systems worden in het vervolg aangeduid door het begrip 'Connectionist AI Paradigma'.

In dit rapport wordt een onderzoek beschreven naar een koppeling van het Connectionist- en het Traditionele AI-Paradigma met als uiteindelijk doel het ontwikkelen van een Connectionistisch Expertsysteem. In het kader van dit rapport wordt alleen het skelet van inferentie, kennisacquisitie en kennisrepresentatie onderzocht. Dit systeem wordt in het vervolg geen expertsysteem, maar redeneersysteem genoemd omdat niet alle facetten van een expertsysteem erin aanwezig zijn. Met dit Connectionistische Redeneersysteem moet geredeneerd kunnen worden met onvolledige, onzekere en inconsistente informatie. Om dit te kunnen doen moet er voor een duidelijk kennisrepresentatie formalisme worden gekozen, dat past bij de connectionistische manier van redeneren. Verder moet het systeem zijn kennis eenvoudig kunnen uit breiden. Tot zover deze vage probleemstelling. In hoofdstuk drie wordt de probleemstelling uitgebreider gedefinieerd omdat in hoofdstuk twee meer achtergrond informatie, nodig voor wezenlijk inzicht in de probleemstelling, wordt gegeven. De lezer met enige mate van Connectionist System kennis zou in principe meteen kunnen doorstappen naar hoofdstuk drie.

In hoofdstuk twee wordt aan beide paradigma's aandacht besteed. In datzelfde hoofdstuk worden ook de belangrijkste Connectionist Systems vanuit historisch perspectief behandeld en wordt er een concreet connectionistisch expertsysteem behandeld.

Om de ambitieuze probleemstelling uit hoofdstuk drie in te dammen wordt in hoofdstuk vier een 'speelgoeddomein' gedefinieerd. Een speelgoeddomein bevat de belangrijkste, elementaire principes van een expertsysteem en mist een aantal zaken die voor een echt expertsysteem wél van belang zijn, maar in het kader van dit onderzoek alleen maar afleiden.

Hoofdstuk vijf bevat het technische deel van het onderzoek. Daarin wordt het redeneersysteem in al zijn facetten uit de doeken gedaan. Dit hoofdstuk wordt gevolgd door hoofdstuk zes, alwaar aan de hand van een simulatie van het model wordt bekeken in hoeverre aan de probleemstelling van hoofdstuk drie is voldaan. Het laatste hoofdstuk, hoofdstuk zeven, bevat een bediscussiëring van de resultaten. Er zal worden aangegeven wat de sterke en zwakke punten van het model zijn, van waaruit aanbeveling voor verder onderzoek worden gedaan.

2 Redeneersystemen

2.1 Inleiding

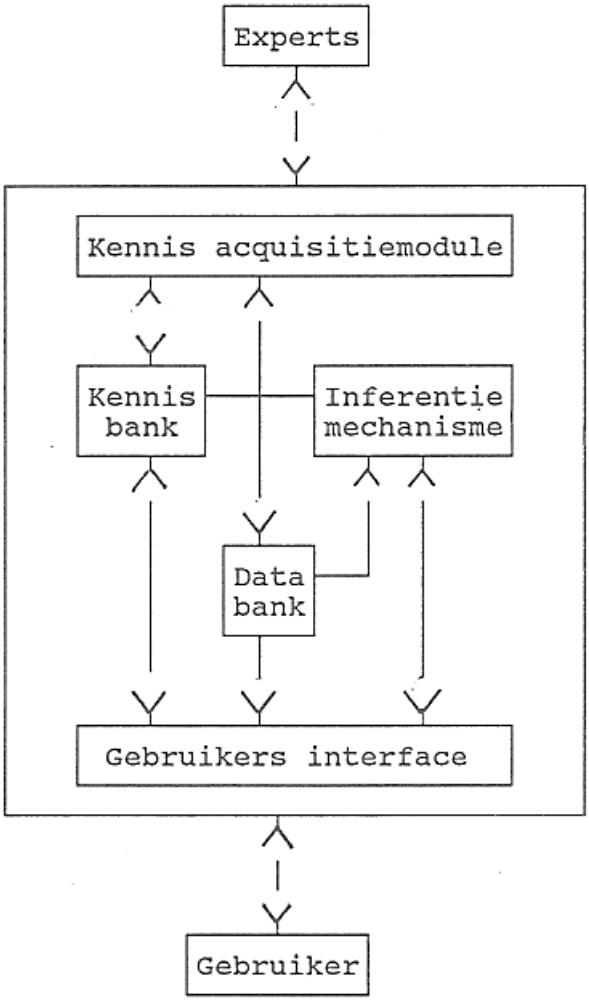
In dit hoofdstuk zullen het traditionele AI-Paradigma en het Connectionist AI-Paradigma tegenover elkaar worden gezet: in respectievelijk paragraaf 2.2 en 2.3. Paragraaf 2.4 bevat een historische behandeling van op het Connectionist AI-Paradigma gebaseerde systemen of modellen. Om te beginnen volgt hier eerst een overzicht van de synoniemen van deze twee paradigma's. In de literatuur kom je namelijk verschillende benamingen tegen:

Traditioneel AI-Paradigma	Connectionistisch AI-Paradigma
Symbolisch Paradigma	Subsymbolisch Paradigma
Paradigma van de Cognitie	Paradigma van de subcognitie

De eerste wordt bijvoorbeeld gebruikt door [Thornton, 1989l , de tweede door [Smolensky, 1987] en de derde door [Hofstadter, 1988]. In het algemeen kom je ze door elkaar tegen.

2.2 Het traditionele AI-Paradigma

Sinds het begin van de jaren '80, toen verschillende expertsystemen zoals DENDRAL, PROSPECTOR en CADUCEUS succesvol dreigden te gaan worden is het onderzoeksgebied van de expertsystemen het meest belangrijke deelgebied van de AI geworden. Een schematische weergave van een expertsysteem kunt U vinden in figuur 2.2.a.



Figuur 2.2.a De anatomie van een Knowledge Based System. Overgenomen uit [Li-Min, 1989]

De drie belangrijkste componenten hierin zijn:

- Kennisbank. Deze bevat in één of andere representatievorm de kennis over een bepaald probleemdomein.
- Inferentiemechanisme. Dit mechanisme, ook wel redeneermechanisme genoemd, tracht aan de hand van de kennis uit de kennisbank de oplossing te bepalen van de door de gebruiker gestelde vraag. De belangrijkste kennisrepresentatievormen met bijbehorende inferentiemechanismen zijn in het volgende schema weergegeven.

Kennisrepresentatie	Inferentiemechanisme
Produktieregels	Top-Down Bottom-Up Waarschijnlijkheidsrekening
Semantische netten of Frames	Overerving
Logica	Resolutieprincipe

In het kader van dit rapport worden alleen de eerste twee kennisrepresentatievormen met bijbehorende inferentiemechanismen besproken, en wel in paragraaf 2.2.

- Kennisacquisitiemodule. Ook wel leermodule genoemd. Deze kan worden gezien als interface tussen expert en expertsysteem. Indien een expertsysteem (nog) niet naar behoren functioneert, d.w.z. met onjuiste antwoorden terugkomt, kan de expert de antwoorden corrigeren opdat de kennisacquisitiemodule de kennisbank zal aanpassen. Deze module is bij de meeste expertsystemen niet aanwezig. Meestal manipuleert de expert rechtstreeks de kennisbank.

Verder ziet U in de figuur een 'Databank', welke gebruikt wordt voor het opslaan van allerlei tussenresultaten, en een 'Userinterface' welke het contact met de gebruiker wat vriendelijker maakt.

Expertsystemen vanuit deze opzet zijn ontworpen vanuit het idee van een sequentiële computerarchitectuur. In principe hoeft er niets gelijktijdig plaats te vinden: de rules zijn sequentieel opgeslagen en moeten sequentieel worden doorlopen en ook het inferentiemechanisme benadert het probleem sequentieel. Er zijn echter wel mogelijkheden tot het parallelliseren van kennisbank en inferentiemechanisme, ter verbetering van de performance. Binnen de deelgroep Kennisgestuurde Systemen, waarbinnen ook dit afstudeeronderzoek plaats vond, wordt onderzoek gedaan naar parallellisatie van inferentie en kennisopslag op een parallelle computer met de naam NCube 4+. Het parallelliseren van expertsystemen gebaseerd op het traditionele AI-paradigma vereist een technisch nogal ingewikkelde aanpak, doordat dergelijke typen expertsystemen zich hier niet 'natuurlijk' voor lenen. Belangrijk is dat het parallelliseren van sequentiële expertsystemen het type paradigma niet verandert: dergelijke expertsystemen gaan niet uit van een wezenlijk andere benadering van cognitie.

In de volgende twee subparagrafen wordt een overzicht gegeven van een aantal belangrijke klassen van expertsystemen. Twee typen traditionele AI-benaderingen, namelijk die van produktieregel systemen en semantische netten, komen aan de orde.

2.2.1 Produktieregel Systemen

Produktieregelsystemen zijn de oudste manier van kennisrepresentatie. Er is inmiddels ontzettend veel onderzoek gedaan naar dit type systemen. Door het vele materiaal hierover is het niet zinvol om allerlei technische problemen en oplossingen van produktieregelsystemen te bespreken. In deze subparagaraaf worden daarom alleen de basisprincipes vermeld met het oog op dié zaken die voor dit rapport van belang zijn. De bron voor deze subparagraaf is [Lucas, 1988] geweest.

A. Newell heeft in het begin van de jaren '70 het begrip 'produktieregelsysteem' geïntroduceerd als model voor het menselijk redeneren. Verondersteld wordt dat kennis te beschouwen is als een verzameling als-dan regels: de produktieregels. Redenatie of inferentie is dan een proces waarbij wordt geprobeerd, uitgaande van een begintoestand (bijvoorbeeld een waarneming) d.m.v. deze produktieregels, in één of meer stappen een gewenste eindtoestand te bereiken. Een produktieregel ziet er, informeel, als volgt uit.

als
    aan bepaalde condities is voldaan
dan
    trek bepaalde conclusies

Een voorbeeld van een produktieregel is :

if
    same (klacht, buikpijn) **and**
    same (ausculatie, ruis) **and**
    same (palpatie, kloppende zwelling)
then
    assign (aandoening, kloppende-zwelling)
fi

Een verzameling produktieregels wordt een 'rulebase' genoemd. Aan de hand van deze rulebase kunnen twee soorten redenaties plaatsvinden:

1. Top down redenatie
Van een 'doelvariabele' moet worden bepaald wat zijn waarde is. Er wordt als het ware achterwaarts geredeneerd en wordt daarom ook wel 'backward reasoning' genoemd.

2. Bottom up redenatie

Gegeven een waarneming moet worden bepaald welke gevolgen dit heeft voor een verzameling doelvariabelen. De redenatie vindt voorwaarts plaats en wordt daarom ook wel 'forward reasoning' genoemd.

Beide methoden werken met een 'feitenverzameling', waar tijdens het redeneren alle informatie, tot dan toe bekend, in wordt opgeslagen. Het beste kan dit worden toegelicht aan de hand van figuur 2.2.b.

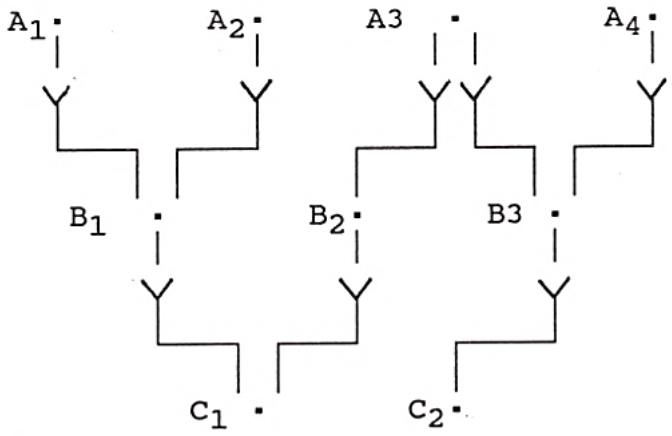


Fig. 2.2.b een schematische weergave van een rulebase

De pijlen stellen if-then constructies voor. Bijvoorbeeld de pijl van A1 naar B1 stelt de regel: if f(A1) then g(B1) voor. f en g geven de functies aan die op de variabele wordt uitgevoerd. In een top-down redenatieproces kan bijvoorbeeld worden onderzocht wat de waarde van C1 is. Uit de verzameling rules wordt dan geconstateerd dat hiervoor de waarde van B1 nodig is. Vervolgens wordt geconstateerd dat ook de waarde van A1 nodig is. Als A1 een dusdanige waarde heeft dat A1 niets over B1 kan zeggen (d.w.z. de rule 'faalt') dan wordt gekeken naar de waarde van A2. Als ook deze rule faalt wordt geprobeerd C1 te bepalen d.m.v. de A3-B2 constructie. De bottom-up methode wordt gebruikt als het bijvoorbeeld niet duidelijk is welke conclusies eventueel kunnen worden getrokken, d.w.z. welke variabelen een waarde zouden moeten krijgen, bijvoorbeeld in het geval van zeer grote knowledge bases. Er zijn dan een aantal waarnemingen bekend, bijvoorbeeld A3, en er wordt geprobeerd zoveel mogelijk resultaat te verkrijgen. In dit voorbeeld probeert het redenatie mechanisme waarden voor B2 en B3 af te leiden om vervolgens iets te kunnen zeggen over C1 en C2.

Vaak echter, gelden de produktieregels met een bepaalde mate van onzekerheid. Binnen het vakgebied van de stochastiek bestaat een formule voor het doorrekenen van probabilistische uitspraken:

Stelling (Regel van Bayes):

> Zij Ω de uitkomstruimte van een experiment en P een kansverdeling op Ω. Zij H_i, i=1..n, disjuncte hypothesen zodanig dat de vereniging van alle H_i gelijk is aan Ω. Als E_k aanwijzingen zijn, die conditioneel onafhankelijk zijn onder elk van de hypothesen H_i; i=1..n; k=1..m; m,n>=1; dan geldt:

$$P(H_i \mid \bigcap_{k=1}^{m} E_k) = \left(\pi_{k=1}^{m} P(E_k \mid H_i) * P(H_i)\right) \Big/ \left(\sum_{j=1}^{n} \pi_{k=1}^{m} P(E_k \mid H_i) * P(H_j)\right)$$

Einde stelling

Het bewijs van deze stelling kunt U naslaan in [Lucas, 1988].

Het grootste nadeel van de toepassing van Bayes in expertsystemen is de grote hoeveelheid kansen die expliciet bekend moeten zijn voor het berekenen van een 'eenvoudige' a posteriori kans. Daardoor zijn er varianten ontwikkeld als benadering voor Bayes. In het kader van dit rapport worden een drietal slechts genoemd.

- De subjectieve Bayesiaanse methode
- Het certainty factor model
- De theorie van Dempster en Shafer

Tot slot van deze subparagraaf volgt een opsomming van de voor- en nadelen van expertsystemen met produktieregels. De meeste punten zijn overgenomen uit [Ringland, 1988].

Voordelen:

1. Kennis is expliciet aanwezig. Waarom bepaalde conclusies kunnen worden getrokken, kan eenvoudig en begrijpelijk worden getraceerd.
2. De if-then vorm sluit goed aan bij de menselijke manier van reproductie van kennis.
3. Ook grote rulebases kunnen worden afgehandeld. Doordat de rulebase een vergaarbak is van regels, zonder structuur, levert de opslag van grote hoeveelheden kennis geen extra technische problemen voor de expert op.

Nadelen:

1. Niet alle problemen kunnen worden gedefinieerd in termen van if-then regels.
2. In een verzameling if-then regels verdwijnt al snel het overzicht over de structuur, waardoor de knowledgebase slecht onderhoudbaar is.
3. Sequentiële aard. Dit heeft twee gevolgen:

   a. Traag, waardoor het slecht of niet toepasbaar is voor real-time toepassingen.
   b. Vaak onduidelijk in welke volgorde de rules afgehandeld dienen te worden.
4. Wat te doen met een onvolledige of inconsistente rulebase?
5. Moeilijk testbaar. Dit is een nadeel voor veiligheidskritische applicaties.
6. Storingsgevoelig. In zowel hardware als software.

In dit rapport wordt fundamenteler terug gekomen op de redenen van bovenstaande zwakke punten van produktieregelsystemen.

2.2.2 Semantische Netten en Frames

Een tweede vorm van kennisrepresentatie is het semantische net en de hiervan afgeleide frames. De meeste informatie die U in deze subparagraaf kunt vinden is ook afkomstig uit [Lucas, 1988]. Het semantische net en frames hebben veel gemeen, maar zijn niet hetzelfde. Het semantische net werd ontwikkeld als psychologisch model voor het menselijk geheugen. Frames zijn hier halverwege de jaren '70 van afgeleid met het oog op praktische toepasbaarheid. Het semantisch net wordt ook wel “associatief net” genoemd; frames vind je in de literatuur ook wel onder de termen “unit” en “concept”.

In tegenstelling tot kennis gerepresenteerd in produktieregels is de structuur van de kennis gerepresenteerd d.m.v. semantische netten en frames zeer belangrijk. Zoals we in de vorige paragraaf hebben gezien is er geen eis aan de kennisstructuur binnen een produktieregelsysteem. Aldaar kan kennis worden gedefinieerd als één grote verzameling als-dan regels. Binnen semantische netten en frames draait het juist om die structuur en de implicaties van een structuur. De plaats van de kenniselementen in die structuur wordt bepaald door de betekenis van die elementen. Bij produktieregelsystemen is die betekenis niet van belang.

Een semantisch net is een gerichte, geëtiketteerde graaf. De graaf bestaat uit knopen en verbindingen tussen de knopen. Een knoop staat voor een concept. Een concept kan een specifiek object of een klasse van objecten aanduiden. Een specifiek object wordt een instantie of instantiatie genoemd. De verbindingen tussen de concepten geven relaties weer tussen de objecten. Er bestaan twee soorten relaties, te weten 'is-onderdeel-van' en 'is-een' relaties. Deze relaties kunnen dus liggen tussen twee klassen van objecten, maar ook tussen een klasse van objecten en een instantie. Voorbeelden hiervan zijn (en deze zijn letterlijk overgenomen uit [Lucas, 1988]):

'Het hart is een onderdeel van het hart-vaat stelsel' en
'De grote arterie is een arterie '

Via de grafenrepresentatie ziet dat er uit als in figuur 2.2.c.

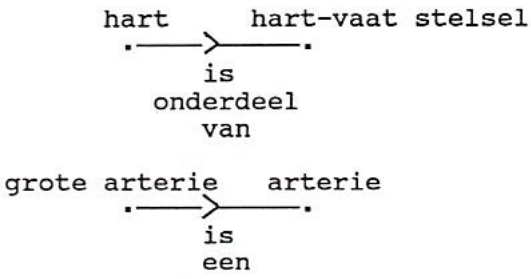


Fig. 2.2.c Grafenrepresentatie van twee eenvoudige uitspraken.

Een wat groter semantisch net is weergegeven in figuur 2.2.d.

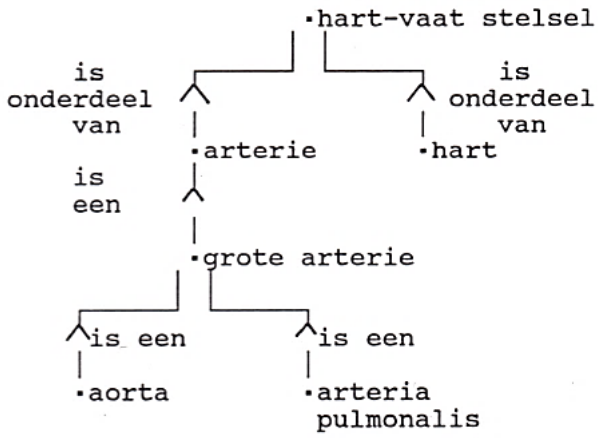
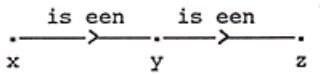


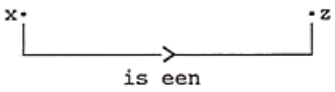
Fig. 2.2.d. Een semantisch net van het hart-vaat stelsel

Redeneren in een semantisch net berust op overerving (inheritance). Dit is een vorm van redeneren waarbij nieuwe uitspraken kunnen worden afgeleid uit de structuur. Overerving is voornamelijk van toepassing op is-een relaties, maar in speciale gevallen vallen ook is-onderdeel-van relaties daaronder. overerving berust op de transitiviteit van een is-een verbinding. Transitiviteit is binnen een semantisch net als volgt aanwezig:

Als x, y en z concepten zijn zodanig dat



dan geldt:



Aan concepten kunnen eigenschappen worden gekoppeld. Van het concept 'arterie' kan bijvoorbeeld worden vastgelegd dat de wand gespierd is en dat het bloed zuurstofrijk is. De grote arterie, die tenslotte een arterie is, kan deze eigenschappen overerven. Nu blijkt dat het hebben van een zuurstofrijk bloed voor een arterie wel heel erg vaak opgaat, maar niet altijd. De arteria pulmonalis, ook een arterie, heeft bijvoorbeeld zuurstofarm bloed. Als we overerving toepassen komen we in een contradictie terecht. Op grond van de is-een relaties zou de arteria pulmonalis, omdat het een arterie is, zuurstofrijk bloed moeten hebben en aan de andere kant héeft de arteria pulmonalis zuurstofarm bloed. Dit kan op twee manieren worden opgelost. De eerste is de eigenschap zuurstofrijk bloed wegnemen bij het concept arterie, om die eigenschap bij alle concepten specifiek op te nemen. Het nadeel hiervan is dat die algemene eigenschap, namelijk het zuurstofrijk zijn van arteriën, verloren is gegaan en we een heleboel redundantie moeten introduceren (want de grote arterie heeft natuurlijk nog veel meer instanties dan alleen de aorta en de arteria pulmonalis, maar die zijn in figuur 2.5.d niet getekend) om er voor te zorgen dat ook de arteria pulmonalis consistent in het geheel past.

Een ander nadeel van het semantisch net is het feit dat het van de concepten niet duidelijk is of het een klasse van objecten dan wel een specifiek object of instantie voorstelt. Dit kan problemen geven in het geval van interactie met de gebruiker. Verder dient opgemerkt te worden dat niet domweg alle eigenschappen aan een concept gekoppeld kunnen worden, hoe wáár ze ook zijn. Als we bijvoorbeeld bij het concept arterie opnemen dat het percentage-bloedvolume 20% is, geldt dat niet meer voor de aorta en de arteria pulmonalis. Dit percentage is natuurlijk kleiner.

Tot zover de semantische netten. In een framerepresentatie kent men vooral is-een verbindingen. De structuur kan op dezelfde manier worden weergegeven als in een semantisch net. De graaf die dan ontstaat wordt wel een frame-taxonomie, een taxonomische hiërarchie of kortweg taxonomie genoemd. De knopen, dus de concepten, worden nu de frames genoemd. Er bestaan twee soorten frames: instanties en klasseframes. Een instantie

geeft de beschrijving van een individu, terwijl een klasseframe (ook wel generieke frame genoemd), een klasse van individuen aangeeft.

Het frame model gaat verder dan een semantisch net en wel in het volgende. Een frame bevat meer dan alleen maar de naam van het concept. Van een frame wordt natuurlijk nog wel, net als binnen het semantisch net, opgegeven wat de hoger in de hiërarchie gelegen frame is. De verbinding van een instantie naar een klasseframe wordt een instantie-van verbinding genoemd en van een klasseframe naar een ander klasseframe een superklasse verbinding. Specifieke kennis over het concept kan binnen een frame worden vastgelegd en wel in attributen of slots. Er is binnen een framerepresentatie voor gezorgd dat we met de attributen iets méér aankunnen dan alleen maar het toekennen van een vaste waarde.

Datgene wat we met een attribuut aan willen wordt gespecificeerd in 'facetten'. De waarde van het attribuut wordt vastgelegd in het 'waarde facet'. Een facet kan bestaan uit een eenvoudige toekenning van een constante aan het attribuut (het 'default facet'). Een facet kan ook, bestaan uit complexere operaties. Voor dat soort operaties hebben we twee soorten facetten, namelijk 'demon facetten' en 'domain facetten'. Een demon facet is een procedure die op een bepaald moment tijdens de manipulatie van een frame wordt uitgevoerd. Via een aparte procedure kan er dan bijvoorbeeld een berekening worden uitgevoerd, een meter afgelezen worden of een vraag aan de gebruiker worden gesteld. Er zijn verschillende demonfacetten die verschillen in het moment waarop de procedure wordt uitgevoerd:

> if-needed demons: De procedure wordt uitgevoerd als binnen het frame een waarde nodig is, maar nog niet bekend.
>
> if-added demons: De procedure wordt uitgevoerd op het moment dat het attribuut een waarde heeft gekregen.
>
> if-removed demons: De procedure wordt nu uitgevoerd als de waarde van het attribuut wordt verwijderd.

Het domain facet wordt gebruikt om het domein van de attribuutwaarden te specificeren. Dit kan zijn in de vorm van een verzameling constanten, maar ook een continu bereik kan worden gespecificeerd.

Er bestaan twee soorten van overerving voor attributen met facetten, namelijk N-overerving en Z-overerving. In het geval van N-overerving wordt eerst geprobeerd het waardefacet over te erven van hoger in de taxonomie gelegen frames. Als dat niet lukt wordt er gekeken of er een defaultfacet kan worden overgeërfd. Als dát op zijn beurt ook niet lukt kijken we naar het if-needed demon etc. De Z-overerving pakt het anders aan. Betreffend frame gaat in dat geval pas kijken naar hoger in de taxonomie gelegen frames als waarde facet, default facet, if-

needed demon, etc. binnen het eigen frame is bekeken, en niets is gevonden om het attribuut een waarde te geven.

Voorgaande sloeg louter op boomvormige frame taxonomieën. Dat houdt in dat ieder frame slechts één, wat wel genoemd wordt, ouder heeft. De bijbehorende manier van overerving wordt 'enkelvoudige overerving' of 'single inheritance' genoemd. Indien we meerdere ouders per frame hebben, dan krijgen we te maken met 'meervoudige overerving' of 'multiple inheritance'. Ieder frame kan dan van meerdere kanten gegevens overerven. Wat nu wel en bij enkelvoudige overerving niet kan ontstaan is het feit dat een frame inconsistente gegevens kan overerven. Het leidt te ver om in dit verband precies uit te leggen wat daar voor een constructies voor bedacht zijn, maar kort gezegd komt het hier op neer. Om te beginnen wordt er een verzameling 'overervingsketens' opgesteld. Deze verzameling bevat alle overervingen die in principe mogelijk zijn binnen de taxonomie. Vervolgens wordt er een 'conclusieverzameling' bepaald welke alle attribuutovererving voor alle concepten bevat. Als hier inconsistenties in aanwezig zijn, wordt gezocht naar welke verbindingen geschrapt zouden moeten worden om het geheel consistent te krijgen.

2.3 Het Connectionist AI-Paradigma

2.3.1 Inleiding

In paragraaf 2.2 zijn een aantal op het traditionele AI-paradigma gebaseerde redeneersystemen besproken. In deze paragraaf wordt ingegaan op de consequentie van dat paradigma. Want stel dat we de kennis van een expert willen vastleggen. Als we dat gaan doen binnen het traditionele AI-paradigma, houdt dat in dat we luisteren naar de expert, verbanden proberen te leggen tussen zijn kenniseenheden (symbolen genoemd) om uiteindelijk zijn/haar kennis op de één of andere manier te formaliseren. De expert produceert op grond van zijn/haar jarenlange ervaring een waterval van symbolen en geeft de relaties aan tussen deze symbolen. De 'knowledge engineer' legt deze kennis vast in een formalisme: een voor het expertsysteem begrijpbare code. Op dat moment is de kennis afgebakend en gedefinieerd. De knowledge engineer is er verantwoordelijk voor dat de kennisregels afkomstig van de expert bruikbaar zijn. Dit houdt in dat de kennis consistent en volledig moet worden gerepresenteerd en dat er een bijpassend redeneermechanisme wordt gekozen waarmee de kennis kan worden gemanipuleerd. Op een klein probleemdomein zal dat in bepaalde gevallen wel lukken, maar de ervaring heeft geleerd dat dergelijke eisen voor een wat groter probleem al snel te zwaar worden. Wat dan ontstaat is een beperkt, inflexibel en vooral moeilijk te onderhouden systeem.

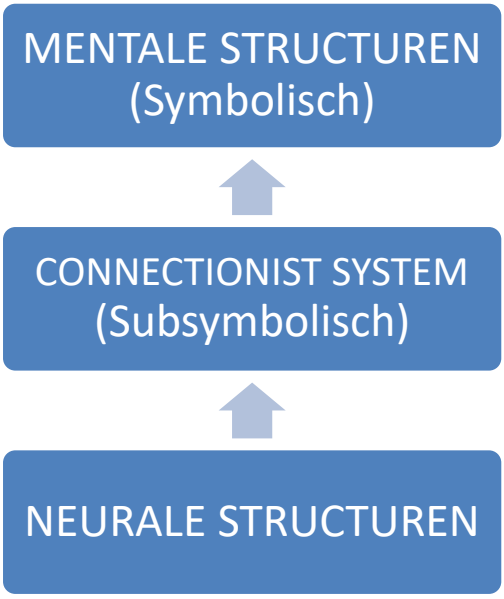
Binnen het Connectionist AI-Paradigma wordt dit probleem onderkend en geeft ons een alternatieve benadering van cognitie. Het introduceert namelijk een lager niveau van probleembeschrijving alwaar veel flexibeler met kennis kan worden omgesprongen. Dit

niveau heet het 'subsymbolisch niveau'. De resultaten verschijnen op het hoge niveau wat meestal aangeduid wordt door de naam 'symbolisch niveau'.

In deze paragraaf zal worden geprobeerd te beargumenteren waarom een benadering gebaseerd op het Connectionist AI-Paradigma een logisch alternatief voor het traditionele AI-Paradigma is. Om dit te kunnen bewerkstelligen wordt er in paragraaf 2.3.2 begonnen met een beschrijving van de hardware van een Connectionist System. In de paragrafen 2.3.3 en 2.3.4 kan dan worden ingegaan op de belangrijkste principes ervan.

2.3.2 De Hardware van Neurale Structuren

Een probleembeschrijving op het subsymbolisch niveau is een poging op een bepaald abstractieniveau de werking van onze hersenen te formaliseren. Er wordt dus niet gepretendeerd dat het subsymbolisch niveau de werking van de hersenen tot op het laagste niveau vangt. Het subsymbolisch niveau bevindt zich dus hoger dan het neurale niveau, maar ook lager dan een symbolisch niveau welke, in het geval van de mens, opgevat kan worden als een mentaal niveau.



Figuur 2.3.a De plaats van het Connectionist System t.a.v. neurale- en mentale structuren.

De stap van neurale structuren (de wérkelijke werking van onze hersenen) naar een Connectionist System is heden ten dage nog erg onduidelijk, maar het is in ieder geval wél duidelijk dat een Connectionist System dichter bij neurale structuren staan dan symbolische systemen staan.

Hoe zo'n Connectionist System er precies uit ziet wordt in het hiernavolgende uitgelegd. De belangrijkste facetten ervan worden weergegeven. Het is beschreven aan de hand van de werking van ons brein en de voor het Connectionist System van belang zijnde kennis daaromtrent. De meeste feiten zijn overgenomen uit [Tank, 1987], [Hofstadter, 1979] en [Feldman, 1988].

De belangrijkste cellen van de menselijke hersenen zijn de 'zenuwcellen' of 'neuronen'. Ieder neuron bezit een aantal toegangspoorten: de 'synapsen', en één outputkanaal: het 'axon'. Ieder neuron heeft gemiddeld ca. 2500 en kan maximaal zelfs 200.000 afzonderlijke toegangspoorten hebben. Verder kan een neuron duizenden andere neuronen sturen. Dit afgezet tegen het feit dat een volwassen mens zo'n tien miljard van die neuronen heeft, geeft een indruk omtrent het ongelofelijk technische resultaat van onze evolutie. Een neuron kent twee toestanden: vuren of niet-vuren. Beslissingen omtrent het al dan niet vuren worden genomen in het 'cellichaam', gelegen in de kern van het neuron: indien de som van alle input aan de synapsen van een neuron boven een bepaalde drempelwaarde uitstijgt vuur het, anders niet. De input en output zijn elektrochemische stroompjes, d.w.z. bewegende ionen. Als een neuron zijn beslissing heeft genomen stuurt het zo'n stroompje, een puls genoemd, langs het axon naar zijn bestemming. Doordat het axon zich kan (en ook meestal zál) vertakken kunnen meerdere neuronen worden bereikt. Ze kunnen hun bestemmingen op verschillende tijdstippen bereiken vanwege het feit dat de axonen onderling een verschillende lengte kunnen hebben. Verder hebben axonen, en dat is ook essentieel voor een Connectionistische Structuur, een verschillende weerstand of 'sterkte'. Een neuron kan dan een gedoseerde invloed uitoefenen op andere neuronen. Op grond van zijn input berekent ieder neuron via een eenvoudige, weliswaar niet-lineaire, functie of het al dan niet actief gaat worden. Een externe activering van één of meer neuronen (een waarneming bijvoorbeeld) resulteert in een informatiestroom tussen de neuronen onderling. Tijdens dit proces van informatie-uitwisseling kan een neuron het ene moment aangespoord worden tot vuren terwijl het andere moment een beslissing tot niet-vuren kan worden gemaakt. Dit iteratieve proces gaat net zo lang door tot er een stabiele situatie ontstaat: alle neuronen zijn het met elkaar eens. Het patroon van welke neuronen wél en welke neuronen niet vuren, representeert het antwoord van het probleem.

2.3.3 Een Filosofisch/Theoretische Legitimatie

Alle traditionele AI-benaderingen, waarvan een aantal besproken in paragraaf 2.2, hebben iets gemeenschappelijks: men tracht cognitie te vangen door slechts naar het niveau van de semantiek te kijken. Als we naar een eenvoudige produktieregel kijken:

ALS Symptoom=Koorts **DAN** Hypothese=Infectie

zien we twee symbolen, namelijk een symbool Koorts en een symbool Infectie. In een traditioneel expertsysteem worden deze symbolen rechtstreeks gemanipuleerd. Het

traditionele AI-Paradigma is ontstaan vanuit de gedachte dat een probleembeschrijving op het symbolische niveau exact en compleet kan zijn. Dit betekent dat lagere niveaus onnodig zijn voor de probleembeschrijving op een hoog niveau. Bovenstaande produktieregel wordt binnen deze benadering 'een beetje intelligent' genoemd en een heleboel produktieregels tezamen kan je dan 'intelligent' noemen. Als het zó wordt gesteld klinkt het al bijna te flauw om uit te leggen dat dit idee rammelt, maar toch is het best verleidelijk om dit paradigma (en nu begrijpt U waarom Smolensky dit het Symbolisch Paradigma heeft genoemd) te aanvaarden. Aan de buitenkant ziet het er namelijk best slim uit wanneer een computer opmerkt dat een patiënt een infectie heeft als deze hem/haar vertelt dat hij/zij koorts heeft. En dat terwijl daar maar één produktieregel aan ten grondslag ligt! Puur en alleen lettende op de buitenkant, op het symbolisch niveau, op het niveau van de semantiek of, zoals Hofstadter het uitdrukt, op het niveau van de cognitie *verschijnt* inderdaad de intelligentie. Aldaar *verschijnen* de symbolen. Als we deze verschijningen echter verabsoluteren door daar een concrete naam, de symboolnaam, aan te geven en aan elkaar relateren door primitieve operatoren zijn we natuurlijk wel het 'idee' achter het symbool kwijtgeraakt. Door dat 'idee' aan het symbool te ontnemen verliest het symbool zijn semantiek. En door het systeem zijn begrip van de semantiek te ontnemen, ontneem je hem zijn intelligentie. Het resulteert in inflexibele en ongenuanceerde systemen.

Een alternatief, het Connectionist System, is gebaseerd op het Subsymbolisch Paradigma. Een kernpunt van dat paradigma is dat het subsymbolisch niveau meer is dan 'slechts implementatie' van een hoger niveau formalisme. Beide niveaus zijn essentieel! Op het lage niveau wordt gedefinieerd wat het systeem *is*, op het hoge niveau wat het *betekent*. Het lage niveau is het niveau van de neuronen, zoals beschreven in de vorige subparagraaf, en de verbindingen ertussen: het niveau van de hardware. Het hoge niveau is een semantische interpretatie van de resultaten verkregen uit het lage niveau. Op het hoge niveau spelen verbindingen geen rol, maar is alleen het patroon van al dan niet vurende neuronen van belang. Op het hoge niveau worden uitspraken gedaan over welke patronen wél en welke patronen níet mogen voor komen (de constraints), resulterend in stabiele respectievelijk instabiele patronen van vurende neuronen. Op het subsymbolische niveau spelen zich continue processen af. Dit in tegenstelling tot de traditionele AI aanpak waar in termen van discrete processen over cognitie wordt gedacht. Binnen Connectionist Systems wordt dan ook niet met logica gewerkt: logica heeft namelijk een discreet karakter. Een wiskunde met een continu karakter en bruikbaar voor Connectionist Systems is de statistiek. Smolensky heeft dit idee gevangen in wat hij noemt de Statistical Connection: "de sterkte van een verbinding tussen twee units is een maat voor de statistische relatie tussen hun activiteiten".

Vanwege het grote aantal toegangspoorten en het hieruit voortvloeiende feit dat een neuron met een ander neuron kan 'praten' maar hem nooit in zijn eentje kan 'overhalen' tot een bepaalde beslissing, worden de verbindingen in een Connectionist System 'Soft Constraints'

genoemd. Het ene neuron kan het andere neuron slechts adviseren. Beslissingen worden genomen op grond van consensus. Uit [Hofstadter, 1988] uit volgende citaat:

> *"Ik geloof dat het van wezenlijk belang is ons te richten op collectieve verschijnselen, met name op het idee dat er informatie of kennis of ideeën op het niveau van collectieve activiteiten (het symbolisch niveau) kan bestaan, terwijl ze op het laagste niveau (het subsymbolisch niveau) volledig ontbreken. In feite zou je zelfs kunnen stellen dat er op het laagste niveau geen informatie bestaat. Dat is niet zo'n verrassende onthulling als je dit vertaalt naar de hersenen: in die neurotransmitters die heen en weer vonken tussen neuronen, bewegen geen ideeën. Toch ondergraaft deze simpele notie het idee dat denken en 'symboolmanipulatie' hetzelfde zijn, als je met 'symbool' een formeel teken, als een bit of een letter of een Lisp-Pnaam bedoelt."*

Even verder geeft Hofstadter een definitie over massacommunicatie die aardig bij dit verhaal aansluit. Deze definitie is afkomstig uit 'The Insect Societies' van E.O. Wilson.

> *"Massacommunicatie wordt gedefinieerd als overdracht van informatie tussen groepen die niet door één afzonderlijk individu aan een ander kan worden overgebracht."*

Er wordt veel geëxperimenteerd met de toepassing van het Subsymbolisch Paradigma in Connectionist Systems. Het werd daardoor duidelijk dat er tijdens het inferentieproces op het symbolisch niveau beslissingen te onderkennen zijn. Op dit niveau, door Smolensky het macro-niveau genoemd, zijn in principe 'hard constraints' terug te vinden. Toch springt het systeem makkelijker, flexibeler om met deze constraints: als het echt niet anders kan zal het systeem geen moeite hebben om een aantal constraints naast zich neer te leggen om hierdoor een zo acceptabel mogelijke oplossing te vinden. De afhandeling van onvolledige en inconsistente informatie geschiedt op een natuurlijke manier. Er zit dus wel structuur in die macrobeslissingen, maar ze kunnen slechts worden benaderd door produktieregels. Dat doet vermoeden dat we met Connectionist Systems de goede kant op gaan, want de reden dat we van de traditionele AI aanpak af wilden was juist dat ook de menselijke beslissingen, in het beste geval, slechts kunnen worden benaderd door produktieregels.

Wat is nu de structuur in die beslissingen? Wat vindt er plaats op het semantische niveau? Daarover handelt de volgende subparagraaf.

2.3.4 De Semantiek

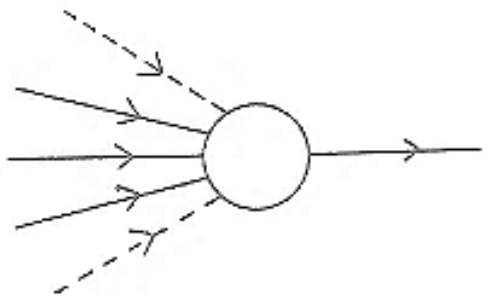
Smolensky tracht het begrip semantiek te concretiseren d.m.v. het begrip 'schemata' [Smolensky, 1986], [Rumelhart & Smolensky, 1986]. Schemata zijn complexe, samengestelde symbolen die op het macro niveau verschijnen, maar nergens expliciet zo gedefinieerd zijn. Tijdens het inferentieproces convergeren de neuronen naar 'context-gevoelige' schemata. Schemata zijn een natuurlijke samenstelling van op zich bekende symbolen aangepast aan de

context. Schemata presenteren zich aan de buitenwereld als een soort frame, besproken in subparagraaf 2.2.2, maar het verschil is dat een frame statisch is en schemata niet.

Smolensky zegt het volgende over de beslissingsstructuur van Connectionist Models:

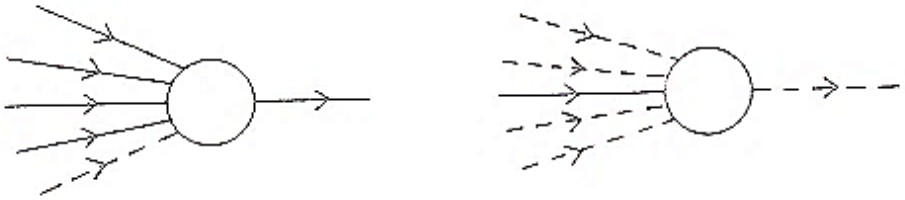
> *"De macro beslissingen (op het niveau van de schemata; het symbolisch niveau) én de microbeslissingen (op het niveau van de neuronen; het subsymbolisch niveau) vinden beiden plaats d.m.v. het zoeken naar maximale 'zelf-consistente' toestanden die consistent met de input zijn."*

Op het subsymbolisch niveau hebben we dat principe inderdaad gezien want neuronen blijven net zo lang van waarde veranderen totdat ze het met elkaar eens zijn: er wordt gestreefd naar een consistente toestand. Volgens Smolensky vindt dit dus ook plaats op het symbolisch niveau. Dat is mooi, want daar gaat het uiteindelijk om. Doordat ieder neuron, tijdens het inferentieproces, precies doet wat hem wordt opgedragen mag je verwachten dat er een proces van 'steeds meer eens worden' plaatsvindt. Beschouw figuur 2. 3. b.



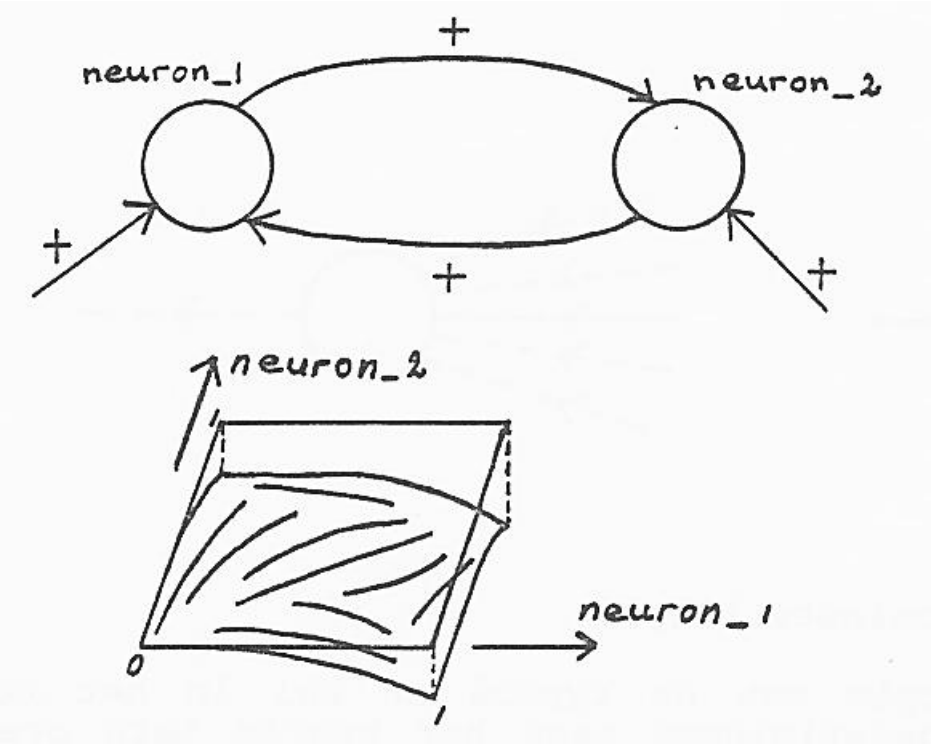
Figuur 2.3.b Een neuroninstelling

In deze figuur zijn de getrokken lijnen signalen die het neuron, zeg in gelijke mate, willen stimuleren. De gestippelde willen het neuron juist afremmen. Stel dat het neuron vuurt als z'n input groter is dan nul. In bovenstaande figuur is dat het geval (want het aantal stimulerende minus het aantal afremmende signalen is groter dan nul) en het neuron zal dus vuren. U kunt zich voorstellen dat bovenstaand neuron, met deze instelling, onderdeel kan zijn van een stabiel netwerk van neuronen: er zijn neuronen die het neuron uit figuur 2.3.b stimuleren en afremmen en het neuron zelf stimuleert of remt andere neuronen op zijn beurt ook weer af. Het geheel is in evenwicht, dus de neuronen zijn het eens geworden. Toch is het duidelijk dat bovenstaand neuron het méér eens kan zijn met zijn omgeving. In figuur 2.3.c zijn hier twee voorbeelden van gegeven.



Figuur 2.3.c Twee neuroninstellingen

Hierin ligt een analogie met de fysica en wel in het begrip 'energie'. Intuïtief gedefinieerd zegt het begrip iets over de mate waarin de neuronen het met elkaar eens zijn: hoe lager de energie, hoe meer consensus. De energie van het neuron in figuur 2.3.b is dan hoger dan die van de neuronen in figuur 2.3.c. Dit zit hem in het feit dat de neuronen uit figuur 2.3.c steviger in hun zadel zitten dan het neuron van figuur 2.3.b: als er slechts één activerend signaal in figuur 2.3.b omklapt verandert de instelling van het neuron al, terwijl dat in figuur 2.3.c niet zo is. De conclusie is daar dus met meer zekerheid getrokken. Nu willen we dat zoveel mogelijk neuronen zo'n lage energietoestand bereiken. Hiervoor moet er een totale netwerk-energie worden gedefinieerd die een soort minimaliseringscriterium weergeeft. De totale netwerk-energie kunnen we bijvoorbeeld definiëren als de som van de energie van alle neuronen afzonderlijk. In dat geval zijn alle neuronen dus even belangrijk en geeft winst in de energie van een neuron onmiddellijk dezelfde winst in de totale energie. Hoe we die energie ook definiëren, in ieder geval ontstaat van een netwerk met n neuronen zo een (n+1)-dimensionaal energielandschap: n dimensies voor de mogelijke waarden (meestal binair) van de neuronen en één dimensie voor de bijbehorende energie. Als we twee neuronen hebben kan dat energieplaatje er bijvoorbeeld uitzien als in figuur 2.3.d. Door het binaire karakter van neuronen zijn alleen de hoekpunten van het energie-landschap significant. De geometrische voorstelling, die een landschap tussen die punten creëert, is een weergave van in hoeverre het punt stabiel is. U moet zich een balletje op het twee dimensionale landschap voorstellen met in de derde dimensie de zwaartekracht naar beneden gericht. Als het balletje stil blijft liggen is het stabiel anders niet.



Figuur 2.3.d Een neuronnetwerk met bijbehorend energielandschap. De neuronen kunnen binaire waarden aannemen.

Indien in bovenstaande figuur ofwel neuron 1, ofwel neuron 2 van buitenaf tot vuren aangezet wordt, dan wordt het tweede neuron ook geactiveerd. Alle verbindingen zijn versterkend, aangegeven door een '+' . Het bijbehorende energieplaatje ziet er uit als aangegeven, waarbij 0=niet-vuren en 1=vuren. In de figuur is te zien dat indien de neuronen zich eerst in toestand (0,0) bevinden ze dan, eventueel via (0,1) of (1,0), de toestand (1,1) zullen bereiken: twee vurende neuronen is het antwoord dat het beste correspondeert met de waarneming.

Als we een complex netwerk van neuronen hebben ontstaat er een bijbehorend complex energielandschap. In figuur 2.3.e is één dimensie met de mogelijke neuronwaarden weergegeven tegen de netwerk-energie. Zo'n 2-dimensionaal plaatje is een situatie waarbij alle andere aanwezige neuronen op een bepaalde vaste waarde getriggerd zijn. Wat we zien is dus niet meer dan een plakje energielandschap. Om het plaatje wat interessanter te maken kunnen de neuronen continue waarden aannemen. Later zullen we zien dat dit in principe mogelijk is door meerdere neuronen in één te beschouwen. Zo'n logische eenheid noemen we een 'unit'.

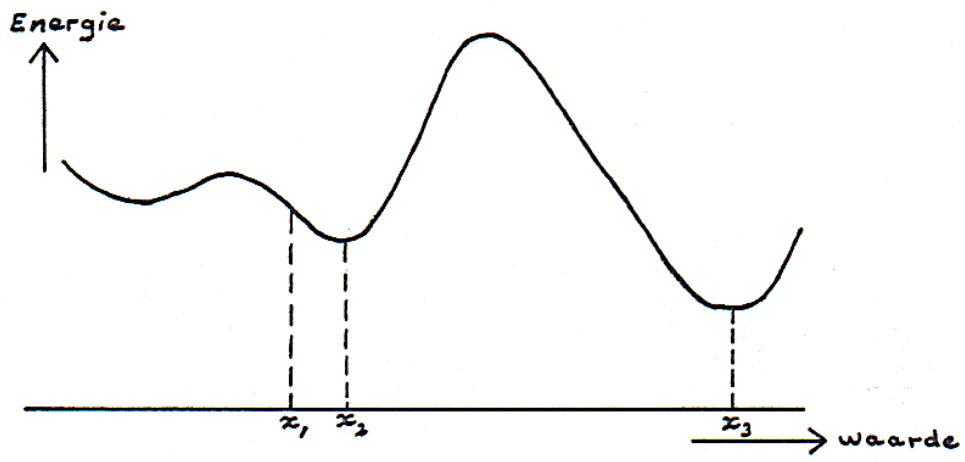


Fig. 2.3.e Een energielandschap van een neuron, met alle andere neuronen op een vaste waarde getriggerd. Het neuron kan continue waarden aannemen.

Op dit niveau van energie kunnen we aan de vorm van het energielandschap de semantiek ontdekken: als alle neuronen de waarde hebben die ze nu hebben dan moet het bovenstaand neuron daar de waarde x_3 bij aannemen. x_2 zou ook wel redelijk geweest zijn, maar x_3 is beter. Het probleem is alleen wel dat een neuron niet een blauwdruk heeft van zijn energieplaatje. Het ziet slechts de input afkomstig van de andere neuronen. Lokaal, op neuronniveau, moet er dus één of andere beslissingsstrategie worden gedefinieerd die gerelateerd is aan het globale energieniveau. Het opstellen van zo'n beslissingsstrategie is het kernprobleem van vele onderzoeken naar Connectionist Systems. Het punt is namelijk dat een neuron niet weet hoe hij zich moet gedragen om een lager (het laagste) netwerk-energie niveau te bereiken. Energie moet dan ook worden gezien als een metafoor voor de semantiek.

In dit verband kan de lezer worden verwezen naar de toepassing van energielandschappen voor het definiëren van logische functies als 'AND', 'OR', etc. in het artikel [Williams, 1986]. Hij gebruikt daarin het begrip 'Degree of Confidence' voor de grootheid Energie. In zijn artikel schetst hij een aantal logische energielandschappen waarin te zien is hoe de semantiek van een logische functie verschijnt zonder dat het expliciet is geprogrammeerd.

Een leeralgoritme binnen het symbolisch paradigma, dus de traditionele benadering van AI, is gebaseerd op het aanpassen, toevoegen of verwijderen van regels, de hard-constraints. Daar tegenover heeft binnen een Connectionist System een leeralgoritme tot doel het energielandschap te vormen of aan te passen door het aanpassen, toevoegen of verwijderen van verbindingen: de soft-constraints, of door het toevoegen of verwijderen van neuronen. Door de grote hoeveelheid verbindingen en neuronen kan dit plaats vinden met een grote mate van subtiliteit.

2.4 Connectionist Systems vanuit Historisch Perspectief

2.4.1 Inleiding en Achtergronden

In de voorgaande paragrafen zijn twee typen zeer van elkaar afwijkende redeneerparadigma's besproken. Inmiddels zijn er een groot aantal op het traditionele AI-Paradigma gebaseerde redeneersystemen ontwikkeld. Heel langzaam begon het belang van de Connectionistische principes duidelijk te worden. Er wordt nu druk gepoogd deze principes te gebruiken voor redeneersystemen in het algemeen, en expertsystemen in het bijzonder. Connectionist Systems wijken, zoals we gezien hebben, op een aantal punten nogal af van de traditionele redeneersystemen. Voor de volledigheid volgen ze hier nogmaals op rij.

Een Connectionist System is:

- intrinsiek parallel, d.w.z. er worden geen trucs uitgehaald om het systeem in een parallel jasje te stoppen, met de mogelijkheid van
- cyclisch gerichte verbindingen, d.w.z. dat door het netwerk niet alle en een feed forward maar ook een feedback informatiestroom mogelijk is. Hierdoor ontstaat na een eventueel
- onvolledige of inconsistente waarneming een
- iteratief proces dat uiteindelijk resulteert in een stabiele situatie: de oplossing van de waarneming. In dat proces zijn twee niveaus te onderscheiden: op een
- subsymbolisch niveau vindt men de systeembeschrijving, terwijl op een hoger niveau; het
- symbolisch niveau, de semantische interpretatie ervan terug te vinden is. Op het subsymbolisch niveau is de beschrijving in termen van
- soft-constraints, waardoor een flexibel netwerk ontstaat wat, zo goed als mogelijk, gehoorzaamt aan de
- hard-constraints van het symbolische niveau. Het netwerk bestaat uit
- vele relatief eenvoudige processoren zonder symbolische betekenis. Tussen de processoren ligt een
- complex netwerk van verbindingen welke dienen tot het kunnen maken van associaties tussen de processoren (subsymbolen) . Op dit gegeven berust ook het
- leren: leren d.m.v. associatie.

In de volgende subparagrafen wordt een overzicht van de ontwikkeling van Connectionist Systems vanuit historisch perspectief gegeven. Verschillende andere volgorden van behandeling van de verschillende typen Connectionist Systems zijn ook mogelijk. Voor het verkrijgen van begrip van het waaróm achter die ontwikkeling, is een historische behandeling ervan de meest voor de hand liggende aanpak. Voor tot die behandeling wordt overgegaan worden eerst een aantal andere indelingscriteria geïntroduceerd. In deze paragraaf worden niet alle termen uit het Engels voortdurend vertaald naar Nederlandse varianten. Een

vertaling van bijvoorbeeld een methode als 'regularity discovery' in 'regelmatigheids ontdekking' werkt alleen maar op de lachspieren. De Nederlandse en Engelse begrippen worden door elkaar heen gebruikt. Verder worden ook de termen unit en node veelal door elkaar heen gebruikt. Beiden staan voor hetzelfde, namelijk voor het meest elementaire rekenelement waaruit het Connectionist System is opgebouwd, ofwel de afgeleide van het neuron.

Een Connectionist System kan worden ingedeeld op redeneren op leereigenschappen. Meestal zie je dat ze op de laatste worden ingedeeld. Het redeneren gaat namelijk bij alle typen Connectionist Systems ongeveer hetzelfde: In subparagraaf 2. 3. 2 is al uitgelegd hoe het redeneren geschiedt in termen van neuronen zoals die in onze hersenen aanwezig zijn. Binnen alle typen Connectionist Systems wordt gewerkt met een vereenvoudiging van dit principe (Daarom worden 'neuronen' in het vervolg 'nodes' genoemd) - Schematisch kan dit worden weergegeven als in figuur 2.4.a.

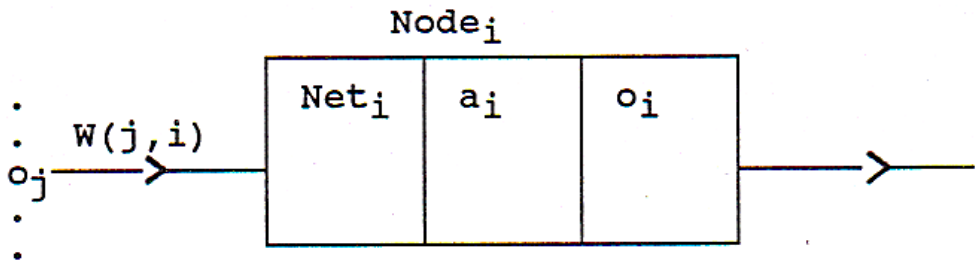


Fig. 2.4.a Schematische weergave van een node

De output van node$_j$, aangegeven door o$_j$, komt verzwakt d.m.v. het gewicht W(j,i) aan bij de ingang van node$_i$. Net$_i$ wordt dan als volgt berekend:

$$Net_i = \sum_j o_j * W(j,i)$$

Dit is bij alle Connectionist Systems, die straks zullen worden besproken, hetzelfde. Vervolgens moet de activationvalue, aangegeven door a$_i$, worden berekend. Meestal is dit een niet-lineaire functie, bijvoorbeeld een sigmoïde functie. De activationvalue geeft de 'toestand' van de node aan. Activation values kunnen binair of discreet zijn. Vervolgens wordt er een afbeelding gemaakt van de activatie op de output. De outputvalue is de waarde die weer wordt doorgestuurd naar de andere nodes. Deze afbeelding is meestal een eenvoudige: namelijk $o_i=a_i$. Soms wordt hier een threshold functie voor gebruikt: als de activation value een bepaalde waarde e overschrijdt vuurt het neuron ($o_j=1$) en anders niet ($o_j=0$).

Op deze manier vindt in alle typen Connectionist Systems redenatie plaats. Wat nog moet worden ingevuld zijn de afbeeldingen van netvalue op activationvalue en van activationvalue

op outputvalue. Nog een andere vrijheidsgraad in het opzetten van een redeneersysteem is de samenhang van de nodes onderling. [Hudson, 1990] onderkent vier soorten van kennisstructurering die hij omschrijft als vier benaderingen (approximations) van onze feitelijke hersenstructuur:

1. Zero-order approximation.

Met deze naam wordt aangegeven dat de kennis, dus het verbindingspatroon tussen de kenniseenheden (de nodes) , ongestructureerd is. Alles kan in principe met alles worden verbonden. Het limietgeval van deze benadering is dan ook een ' fully interconnected' netwerk.

Het Hopfield net, dat later nog uitgebreider aan de orde komt, is hier een voorbeeld van. Het nadeel van deze manier van kennisstructurering is de zware eis van het grote aantal verbindingen.

2. First-order approximation.

Deze slaat op gelaagde netwerken van twee of meer niveaus. Binnen elk niveau is tussen de nodes onderling volledige interconnectie mogelijk, terwijl aan verbindingen tussen de verschillende niveaus restricties zijn opgelegd. Als we drie niveaus A, B en C hebben kan de restrictie bijvoorbeeld zijn dat A en B onderling kunnen communiceren, B en C onderling kunnen communiceren maar A en C dat niet direct mogen. Zie hiervoor figuur 2.4.b.

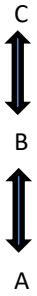


Fig. 2.4.b Voorbeeld van een First-order netwerkstructuur

Via deze benadering wordt het aantal verbindingen sterk gereduceerd. Als de verbindingen tussen twee lagen in beide richtingen liggen (zoals in figuur 2.4.b) spreken we van een 'recurrent' netwerk. Als er alleen verbindingen van input richting output mogelijk zijn spreken we van een 'feedforward' netwerk. Vaak is het zo dat bij een gelaagde netwerkstructuur drie typen nodes, ook wel: units, zijn te onderscheiden en wel de input-, output- en ‘hidden’ units. De inputunits ontvangen signalen vanaf de buitenwereld zoals bijvoorbeeld signalen afkomstig van sensoren. De outputunits geven de resultaten van de berekeningen van het

systeem weer terug aan de buitenwereld. Deze twee niveaus zijn dus zichtbaar, in tegenstelling tot de hiddenunits die nodig zijn voor het mogelijk maken van complexe input-output transformaties: die zijn voor de buitenwereld onzichtbaar.

Een interessante ontdekking is in dit verband het vermelden waard. Deze kwam van Minsky & Papert en heeft betrekking op het aantal lagen in een Connectionist System volgens een first-order approximation. Beschouw nogmaals figuur 2. 4.b. We hadden daar drie lagen van nodes. Minsky & Papert bewezen dat netwerken met drie of meer lagen gereduceerd kunnen worden tot netwerken van twee lagen als de activatie- en outputfuncties van alle nodes lineair zijn. Het bewijs hiervan is wel aardig:

> Stel we hebben drie lagen, zoals in figuur 2.4.b De verbindingen tussen de drie lagen kunnen worden aangegeven door twee verbindingsmatrices met constante coëfficiënten: W_1 en W_2. Element wij van deze matrices bevat de waarde van de verbinding van node i van de sturende laag, naar node j van de ontvangende laag. De activationvalues van de drie niveaus kunnen worden weergegeven door de vectoren a_A, a_B en a_C. a_A is dus ook een lineaire functie van de input i: $a_A = \beta_1 * i_A$. a_B is op zijn beurt weer een lineaire functie van a_A: $a_B = \beta_2 * W_1 * a_A$ en dan geldt op dezelfde manier: $a_C = \beta_3 * W_2 * a_B$. In het algemeen geldt dus voor de activationvalues a van de outputlaag als functie van de input i aan de inputlaag, van een netwerk met N lagen:
>
> $$a = \prod_{l=1}^{n} (\beta_l * W_l) * i$$
>
> Nu is het zo dat het product (van l=1 tot n) in de vorige formule te vervangen is door een nieuwe matrix W met constante coëfficiënten. We houden dan over: a = W * i. Kortom het netwerk met N lagen is te vervangen door twee lagen met de verbindingsmatrix W.

Voorbeelden van First-order netwerken zijn Multi-layer Perceptrons (subparagraaf 2.4.3), Competitive Networks ([Rumelhart & Zipser, 1986]) en ook Boltzmann machines (subparagraaf 2.4.4) en implementaties van de Harmony Theorie (subparagraaf 2.4.4) vallen hieronder.

3. Second-order approximation.

Als de kennisstructuur een meer modulair karakter heeft wordt deze term gebruikt. Modules bepalen onafhankelijk van elkaar oplossingen voor bepaalde subproblemen. Op een hoger niveau worden deze oplossingen pas met elkaar in verband gebracht. Via deze methode verkrijgen we een grotere flexibiliteit in kennisverwerking dan de voorgaande twee.

Zoals we later zullen zien is SCORE, het afstudeeronderzoek dat later in dit rapport wordt besproken, ook gebaseerd op een second-order approximation.

4. Third-order approximation.

Hierin gaan we nog verder dan de voorgaande drie. Daar werd namelijk nog verondersteld dat alle nodes van hetzelfde type waren: activation- en outputvalues waren voor alle nodes hetzelfde. Binnen een netwerk volgens een third-order approximation hoeft dit niet meer zo te zijn. Binnen het menselijk brein zijn ook verschillende typen neuronen aanwezig (bijvoorbeeld Pyramidal cells, Stellate cells, motoneuron cells, glial cells) die allen hun specifieke eigenschappen bezitten. Door de bundeling van deze eigenschappen in één netwerk kan een nog krachtiger geheel ontstaan.

Implementaties van deze benadering zijn de schrijver van dit rapport onbekend.

Bovenstaande categorisering is opgesteld in het licht van de architectuur. Als de architectuur is gedefinieerd en alle nodefuncties bekend, dan kan het redeneren in principe gaan aanvangen. Omdat de verbindingen tussen de nodes nog niet hun waarde hebben zal er ook moeten worden geleerd om tot een correct redenatieresultaat te kunnen komen. Leren is dus gericht op het dusdanig waarde geven aan de verbindingen zodat op verschillende vragen het juiste antwoord kan worden gegeven.

Hiermee komen we tot een volgende indelingscriterium, verkregen door combinering van criteria van [Rumelhart&Hinton, 1986] en [Murre, 1990], gebaseerd op verschillen in benadering van wat 'leren' nu eigenlijk precies kan betekenen.

I. Associative learning.

Dergelijk leertype staat voor het kunnen leggen van verbanden tussen het ene willekeurige patroon en een ander willekeurig patroon. Als aan de input van een netwerk een patroon wordt aangeboden dan dient er aan de output een specifiek patroon te verschijnen. Omdat er geen logisch verband tussen beide patronen hoeft te bestaan hebben we een externe leraar (teacher) nodig die het netwerk de juiste antwoorden voorhoudt. Tijdens een trainingsperiode is het is de taak van het netwerk volgens één of ander algoritme de gewichten zo aan te passen dat uiteindelijk na aanbieding van willekeurige input de juiste output verschijnt. Als input en output duidelijk gescheiden niveaus zijn noemt men dit type associatie wel ' patroon associatie ' of ook wel 'hetero associatie'. Als input en output dezelfde zijn wordt dit type associatie aangegeven door de term 'auto associatie'. Zoals de naam dat al aangeeft, wordt met deze laatste vorm van associatie een associatie van de input met zichzelf bedoeld. Het doel daarvan is patroon completering: de output is dan het onvolledige deel van het patroon dat na elke waarneming weer anders gelegen kan zijn. Iedere node is zowel input als output. Auto associatie is een speciale geval van patroon- of

hetero associatie. De laatste kan namelijk ook worden gebruikt als auto-associator door aan de input en aan de output gelijke patronen aan te bieden.

Omdat we in het geval van associatief leren een externe teacher nodig hebben die het netwerk vertelt wat bij wat hoort, wordt deze manier van leren ook wel 'supervised learning' genoemd.

Ter volledigheid volgt hier een derde type leren en wel het type “reinforcement” leren. Het netwerk krijgt nu geen afgerond leersignaal aangeboden, maar krijgt slechts een goed/fout signaal. Zolang het netwerk de juiste antwoorden geeft of goed handelt krijgt het een goed-signaal. Op het moment dat er iets misgaat wordt het foutsignaal geven, waardoor het netwerk kan leren dát, wat tot die fout heeft geleid, niet meer te herhalen.

Vooruitlopend op de volgende subparagrafen is het zo dat het Perceptron bij uitstek een voorbeeld is van een model van patroon associatie, terwijl het Hopfield Net een voorbeeld is van een auto-associatief model.

II. Regularity discovery.

Bij dit type leren is het de taak van het netwerk regelmatigheden te ontdekken in het inputpatroon. Zonder externe teacher is een netwerk van dergelijk type in staat bepaalde eigenschappen van de input te ontdekken. Omdat er geen externe teacher nodig is noemen we deze vorm van leren ook wel 'unsupervised learning' : de 'teaching function' wordt door het systeem zelf bepaald. Op basis van een algemene interne teaching function en het karakter van de trainingspatronen bepaalt het netwerk zelf op welke eigenschappen het zal leren te reageren.

Voorbeelden van modellen die gebaseerd zijn op een dergelijke manier van leren zijn 'Competitive learning ' en de "Harmony Theorie'.

III. Imitatie leren.

Voor deze vorm van leren hebben we niet echt een leraar, maar meer een soort ' controleur' . Deze controleur is een soort registratieapparaat die de resultaten van het netwerk registreert en terugkoppelt naar de input. Door aan de outputkant de omgeving te variëren, kan het systeem via deze manier leren van de handelingen die het doet. Beschouw hiervoor figuur 2.4.c.

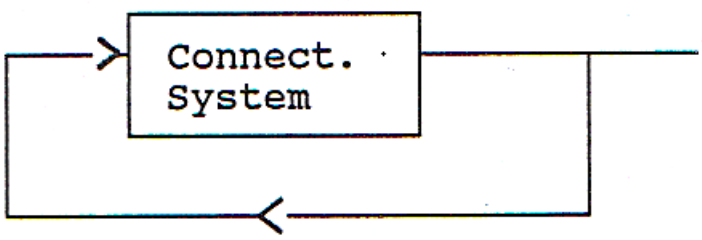


Fig. 2.4.c Imitatie leren

Tot zover deze indeling naar leertypen. Een laatste korte indeling kan er worden gemaakt op het type problemen dat we met Connectionist Systems kunnen oplossen. Deze indeling komt van [Bokhoven, 1990].

1. Associatieve geheugens.

Dit type is hiervoor besproken en wordt dus gebruikt voor patroonherkenning en patrooncompletering.

2. Classification problems.

Ook dit type is hiervoor besproken. Het leren van Classification Problems valt onder de Regularity Discovery modellen. Netwerken voor het oplossen van Classification problems kunnen bijvoorbeeld worden verkregen d.m.v. Competitive learning.

3. Feature extraction.

Dit valt ook onder het type Regularity Discovery. Nu gaat het echter niet om het indelen van patronen in verschillende categorieën, maar gaat het om het ontdekken van specifieke eigenschappen van die patronen. Het doel van feature extraction is meestal patroon completering, zoals bijvoorbeeld in de Harmony Theorie.

4. Optimaliseringsproblemen.

Een type problemen dat niet direct iets te maken heeft met patronen zijn optimaliseringsproblemen. Een klassiek optimaliseringsprobleem uit de theoretische informatica is het handelsreizigersprobleem ofwel het Traveling Salesman Problem (TSP) . Hierbij moet een handelsreiziger een aantal steden bezoeken. Dit moet zodanig gebeuren dat alle steden precies éénmaal worden aangedaan en de totale af te leggen afstand minimaal is. Er blijkt geen algoritme te bestaan dat dit probleem in een polynomiale tijd oplost. Voor dertig steden is het totaal aantal mogelijke routes in de orde 10^{30}. Het is ondoenlijk alle routes door te rekenen.

Voor dergelijke problemen blijken Connectionist Systems ook heel goed inzetbaar. J. J. Hopfield heeft voor het TSP probleem een Connectionist System gedefinieerd waarmee in korte tijd weliswaar niet de beste maar wel een zeer acceptabele oplossing kan worden gevonden.

Tot zover de categorisering van de verschillende Connectionist Systems. Het wordt nu hoog tijd dat we overgaan op de historische behandeling van de belangrijkste typen Connectionist Systems. Er zal regelmatig worden terugverwezen naar bovenstaande indelingscriteria.

2.4.2 McCulloch&Pitts

McCulloch & Pitts waren waarschijnlijk de eerste onderzoekers die zich bezig hielden met de modellering van de werking van de hersenen. In 1943 onderzochten zij het gedrag van 'neurale netwerken' met sterk geabstraheerde neuronen. Zij legden de basis voor de rekenelementen van Connectionist Systems zoals die heden ten dage nog worden gebruikt. Figuur 2.4.d is een copy van de schematische weergave van een node, zoals die in 2.4.1. is geïntroduceerd.

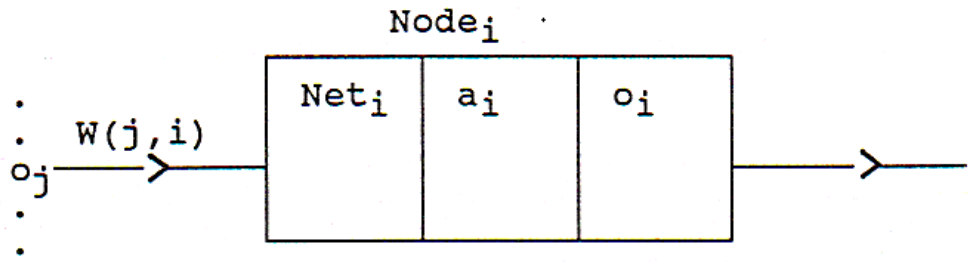


Fig. 2 . 4. d. McCulloch&Pitts cel

Het was het werk van McCulloch & Pitts de netfunctie te definiëren:

$Net_i = \sum_j o_j * W(j,i)$

Verder was hun activatiefunctie de niet-lineaire, binaire stapfunctie. In figuur 2.4.e is het algemene geval van die functie geschetst, met threshold Θ.

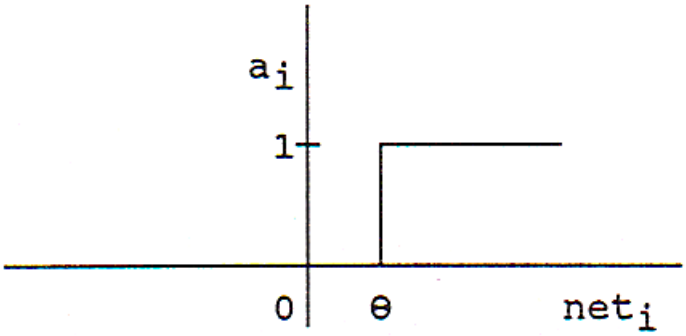


Fig. 2.4.e Activatie functie: binaire stapfunctie met threshold e

Andere benamingen voor deze functie zijn 'Hard limiter ' of gewoon 'Threshold functie '. Later hebben andere onderzoekers het belang van hiervan afwijkende activatiefuncties aangetoond. De belangrijkste worden hier alvast getoond, te beginnen met een lineaire activatiefunctie: de eenheidsfunctie. Zie figuur 2.4.f.

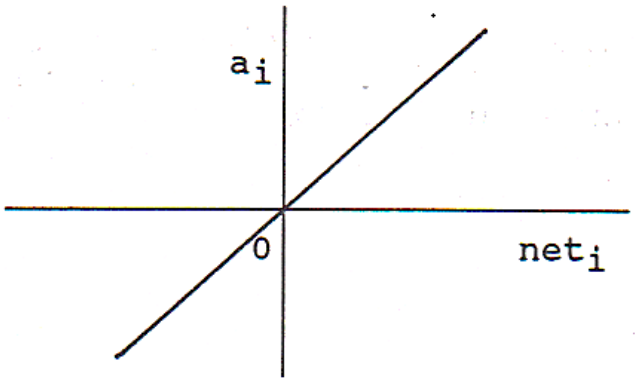


Fig. 2.4.f Activatie functie: de eenheidsfunctie

Omdat er geen enkele beperking wordt opgelegd aan de activatiefunctie wordt deze in de praktijk niet gebruikt. Later zullen de onderzoekers Minsky & Papert bewijzen dat de kracht van dergelijke lineaire rekenelementen niet groot is. Later wordt hier op terug gekomen.

Een vergelijkbare activatiefunctie die toch een bepaalde mate van niet-lineairiteit in zich heeft is de functie met de naam 'subthreshold sommatie' en is weergegeven in figuur 2.4.g.

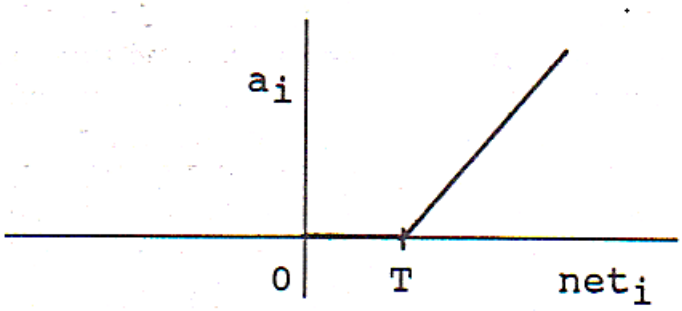


Fig. 2.4.g Activatie functie: subthreshold sommatie

Als een logisch gevolg van het gebrek aan grenzen die de activatiewaarde kan aannemen, ontstond er het volgende alternatief van figuur 2.4.h.

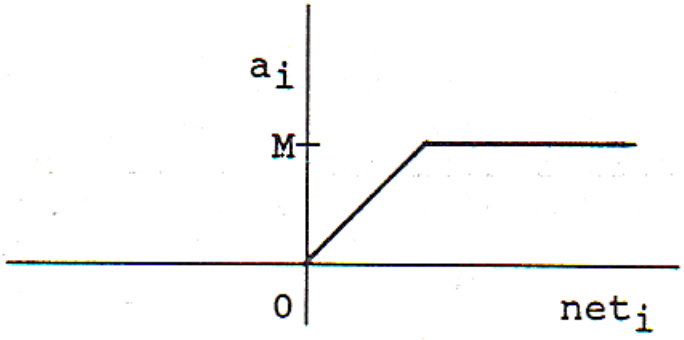


Fig. 2.4.h Activatie functie: Linear Saturating Limiter

Een andere veelgebruikte functie is de sigmoïde functie, welke een soort kruising is van de subthreshold sommatie en de Saturating Limiter. Zie hiervoor figuur 2.4.i.

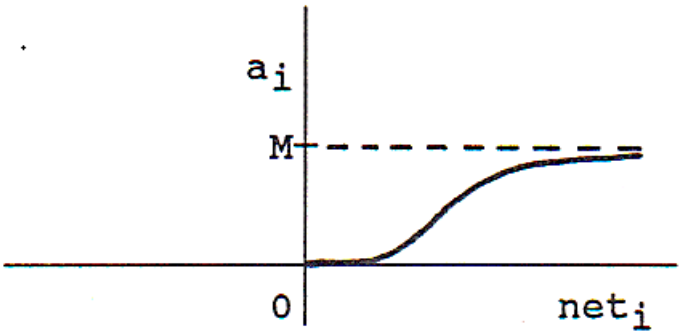


Fig. 2.4.i Activatie functie: Sigmoïde functie

De formule voor deze functie is: 1 / (1 + EXP (-net_i)) . Deze functie heeft een aantal fijne eigenschappen. Het is bijvoorbeeld continu en differentieerbaar. Voor de backpropagation regel die later besproken zal worden is dit namelijk een voorwaarde. Voor de afgeleide van een sigmoïde functie f(x) geldt: f'(x) = f(x) * (1-f(x)) en is dus zeer gemakkelijk uit te rekenen.

Tot zo ver de verschillende activatiefuncties. Voor de omzetting van activatiewaarden naar outputwaarden wordt vaak de eenheidsfunctie, zoals in figuur 2.3.f, gebruikt. Ook de andere hierboven geschetste activatiefuncties kunnen dienen als outputfuncties. De outputfunctie is ook vaak één of andere threshold-achtige functie, zodat de node pas een andere node kan beïnvloeden als zijn activationvalue een bepaalde waarde heeft overschreden. Soms is de outputfunctie een stochastische functie waardoor de output van een node op een probabilistische wijze afhangt van zijn activationvalue. Bij veel modellen echter worden activationvalues en outputvalues door elkaar heen gebruikt. Als de outputfunctie bijvoorbeeld een eenheidsfunctie is, steekt er geen kwaad in activatie en output onder één noemer te gooien en één functie over te houden. Het idee van net-activatie-output is ook maar een benadering van een neuron. Tot slot dient opgemerkt te worden dat, naast de activatiefunctie die een hard-limiter is, de outputfunctie van bovengenoemde McCulloch & Pitts cel inderdaad zo'n eenheidsfunctie is.

2.4.3 Het Perceptron

2.4.3.1 Rosenblatt

Het Perceptron, in 1958 reeds ontwikkeld door F. Rosenblatt, is één van de eerste Connectionist Systems. Het bestaat uit twee lagen McCulloch & Pitts cellen. De activationvalue is weer een hard-limiting functie van zijn netvalue, en de output volgt de activation.

Het Perceptron gaat echter verder dan de McCulloch & Pitts cellen. Het kan namelijk ook nog leren. Dat was iets waar McCulloch & Pitts nog niet mee overweg konden. Die hielden zich alleen bezig met die 'menselijke' manier van kennisopslag en het gebruik ervan. In figuur 2. 4.j is een schematische afbeelding van een Perceptron opgenomen.

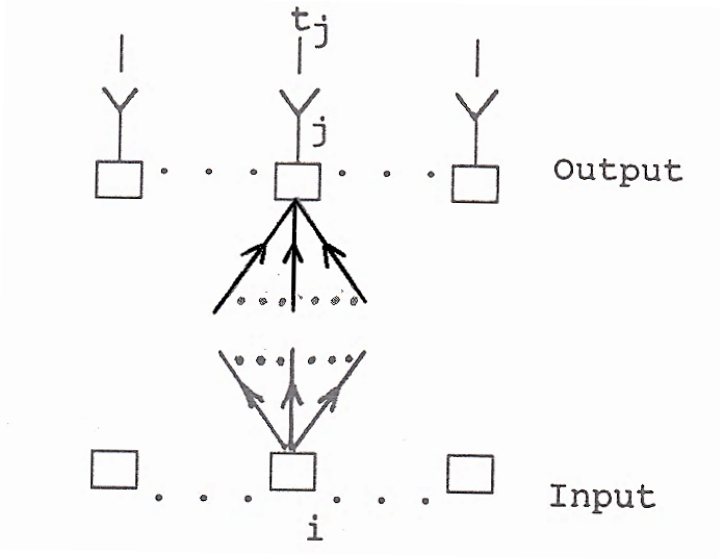


Fig. 2.4.j Het Perceptron

Alle nodes uit de inputlaag zijn verbonden met alle nodes uit de outputlaag en wel in de richting van input naar output. Omgekeerd liggen er geen verbindingen. Op dit punt wordt er al niet voldaan aan de lijst van Connectionist-eigenschappen zoals die in het begin van subparagraaf 2 - 4 - 1 zijn gedefinieerd. Het Perceptron van Rosenblatt is helemaal binair gericht. Aan de input kan een binair patroon worden aangeboden die geassocieerd wordt met een binair patroon aan de outputkant. Bij aanvang hebben alle gewichten een randomwaarde. Er is een training voor nodig om het netwerk de juiste associaties te kunnen laten maken. Het leeralgoritme dat hiervoor geschikt is, is de 'Hebbrule' ook wel gewoon de 'Perceptron leerregel' genoemd. Dit leeralgoritme maakt gebruik van een externe teacher (en is daarom een vorm van supervised learning) , in figuur 2. 4.j aangegeven door t. Voor elke outputnode j wordt na het aanbieden van een inputpatroon gespecificeerd wat zijn (binaire) waarde moet zijn. De Hebb-rule die dan toegepast wordt ziet er als volgt uit:

W_{ij}-nieuw - W_{ij}-oud = $\mu * (t_j - a_j) * a_i$

Hierin staat a voor de activationvalue en t voor de teacher en is μ de 'leersnelheid', of 'learning rate'. W_{ij}-nieuw - W_{ij}-oud is de verandering die de verbinding W_{ij} ondergaat. Na elk patroon dat wordt aangeboden, ondergaan de vérbindingen een verandering, die gelijk is aan

+ of - de leersnelheid μ (O <= μ <= 1) : doordat a_i, a_j en t_j binair zijn, is W_{ij}-nieuw - W_{ij}-oud gelijk aan +/- μ. De verandering in W_{ij} wordt bepaald door de vraag of a_i één is en in hoeverre het antwoord correct was, hetgeen wordt gespecificeerd door de term (t_j—a_j) die dus de waarden -1, 0 of 1 kan aannemen. Er kan worden bewezen dat als van een verzameling input/output patronen een tweelagig netwerk kan worden gemaakt, het dan zo is dat d.m.v. bovenstaande leerregel de juiste gewichten kunnen worden gevonden. Dit wordt wel het "Perceptron Convergentie Theorema" genoemd.

Het Perceptron kan, wanneer we slechts één node in de bovenste laag hebben, worden gebruikt als classificeringsnetwerk voor bijvoorbeeld eigenschappen van patronen. Als die ene node vuurt, dan bevat het patroon bijvoorbeeld wél een bepaalde eigenschap en als het niet vuurt, dan bevat het niet die eigenschap. Via een N-tal losse Perceptrons, die allen een deel van het totale patroon voor hun rekening nemen, kan er een systeem worden gebouwd die op grond van het al dan niet aanwezig zijn van de N bijbehorende eigenschappen beslist of het totale patroon wel of niet tot een bepaalde categorie behoort. Rosenblatt noemde deze toepassing het Mark I Perceptron. Het Mark I Perceptron is dus opgebouwd het een N-tal kleinere Perceptrons die eventueel weer opgebouwd kunnen zijn uit nog kleinere, enzovoorts. In figuur 2.4.k is een schema van zijn werking opgenomen, overgenomen uit [Forsyth, 1986].

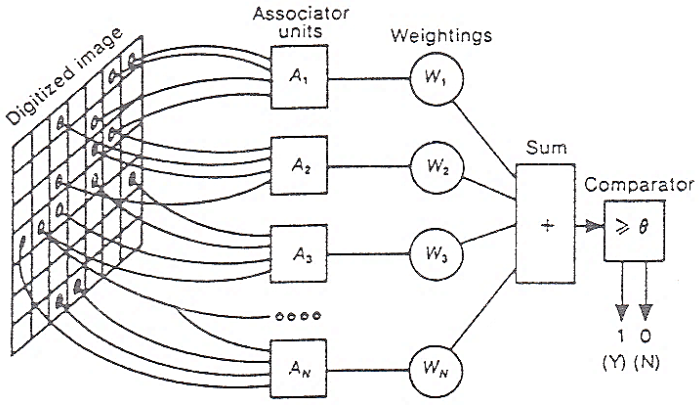


Fig. 2.4.k. Het Mark I Perceptron

Het is in staat beelden te scheiden in twee categorieën. Pixels van het gedigitaliseerde beeld worden aangeboden aan 'associator units' , die misschien beter aangeduid hadden kunnen worden met de term 'classifying units'. Iedere associator is verbonden met een willekeurig aantal pixels. De associator units zijn de feature detectors van het systeem. Deze units 'fire', d.w.z. geven aan hun uitgang een één, wanneer bijvoorbeeld vier of meer bijbehorende pixels aan (één) zijn. De output van unit A_i, aangegeven door A_i', is dus nul of één. Vervolgens

worden deze resultaten gewogen via de gewichten W_i>=0 en geeft de uitgang van de Perceptron 'Yes' als

$$\sum_{i=1}^{N}(A_i' * W_i) \geq \Theta$$ en 'No' als dit niet het geval is.

De gewichten W_i zijn in het Perceptron variabel. Als het antwoord van het systeem correct is, veranderen de gewichten niet van waarde. Als het niet correct is, wordt er een leerregel toegepast, die een afgeleide is van de hierboven besproken Hebbrule:

$W_j = W_j - A_j' * d$ met
d=+1, als het systeem 'Yes' als antwoord gaf, en de teacher zegt 'No' en
d=-1 als het systeem 'No' als antwoord gaf, en de teacher zegt 'Yes'.

Zoals U ziet is dit een vorm van reinforcement leren.

Deze manier van leren wordt ook wel 'Proportional Increment Training Strategy' genoemd. Het kan worden bewezen dat de gewichten convergeren naar een optimale set, mits de twee klassen van problemen lineair scheidbaar zijn. Het blijkt dat de resultaten bij niet-lineair scheidbare klassen ook zeer acceptabel zijn. Dit systeem kan ook worden uitgebreid zodat meer dan twee klassen onderscheidbaar zijn. Het associërende karakter van het Perceptron wordt dan volwaardiger gebruikt dan in het Mark I Perceptron het geval is.

2.4.3.2 Widrow-Hoff

In het jaar 1960 bracht de 'Widrow-Hoff' regel, ook wel de 'delta-rule' genoemd, een volgende golf van enthousiasme teweeg. Het Widrow-Hoff Perceptron is wat genuanceerder dan het Rosenblatt Perceptron. Het berekenen van net-, activation- en outputvalue geschiedt nog steeds op dezelfde manier als bij Rosenblatt, maar alleen de leerregel werkt niet met de discrete activationvalue, maar met de continue netvalue. Doordat nu ook de teachingvalue continu kan zijn, kan het netwerk nauwkeuriger worden afgeregeld. De Delta-rule:

W_{ij}-nieuw - W_{ij}-oud = $\mu * (t_j - net_j) * a_i$

In [Rumelhart&Hinton, 1986] kunt U het bewijs vinden van het feit dat via deze methode van leren, de som van de kwadraten van de fouten $\varepsilon_j = t_j - a_j$ worden geminimaliseerd als men maar genoeg trainingsvoorbeelden aanbiedt. Dan convergeren de waarden van de verbindingen naar dit optimum.

In het voorgaande is geschreven dat het Perceptron Convergentie Theorema geldt voor problemen die zich laten vangen door een twee-lagig netwerk. Minskey & Papert (1969) hebben aangetoond dat niet alle problemen beschreven kunnen worden door zo'n netwerk. Voor bepaalde problemen zijn meerdere lagen nodig. Het implementeren van een Exclusive-OR-functie d.m.v. een Connectionistisch netwerk bijvoorbeeld, kan pas geschieden als we drie

lagen tot onze beschikking hebben. Deze derde laag die tussen input en output gelegen is, wordt wel een "hidden layer" genoemd. De Hebb-rule en Delta-rule zijn niet bruikbaar voor netwerken met meer dan twee lagen omdat voor de hidden layers geen teaching-value bekend is. Door de onduidelijkheid over hoé meerlagige netwerken te kunnen laten leren, heeft het onderzoek naar Connectionist systems lange tijd in de koelkast gestaan: het probleem van de Exclusive-OR heeft geleid tot het bijna geheel stoppen van het onderzoek naar Connectionist Systems. Pas in de jaren 80 ontstonden er verschillende oplossingen voor het Exclusive-OR probleem. Eén daarvan wordt in de volgende subparagraaf behandeld.

2.4.3.3 De gegeneraliseerde Delta-rule

Dit type Perceptron is tot op heden het meest populaire Connectionist System gebleken. Dit type werd door Rumelhart, Hinton en Williams in 1986 opnieuw onder de aandacht gebracht [Rumelhart&Hinton, 1986), nadat Werbos in 1974 en Parker in 1982 deze regel al hadden beschreven. Inmiddels is er nog een andere naam hiervoor in omloop: "Error back propagation rule 1 - M. b.v. deze regel kan nu ook een Perceptron met meer dan twee lagen worden getraind, zonder dat de hiddenunits een teaching-signaal nodig hebben. Dit type Perceptron wordt ook wel een 'multi layer Perceptron' genoemd.

Het feit dat er meerdere lagen worden toegestaan zegt in ieder geval twee dingen: ten eerste betekent dit dat de nodes niet-lineaire activatiefuncties uitvoeren (Minsky & Papert, zie subparagraaf 2.4.1), ten tweede betekent dit dat de leerregel geen teachingvalue behoeft voor de hiddenunits. De leerregel ziet er als volgt uit:

W_{ij}-nieuw - W_{ij}-oud = $\mu * \delta_j * a_i$, met μ = leersnelheid

Hierbij wordt de δ voor output- en hiddenunits verschillend bepaald:

$\delta_j = f_j'(net_j) * (t_j - a_j)$ voor j=outputunit

$\delta_j = f_j'(net_j) * \sum_k(\delta_k * W_{jk})$ voor j=hiddenunit

Iedere training van een netwerk bestaat nu uit twee fasen. Tijdens de eerste fase wordt een patroon aan de input van het netwerk aangeboden. Propagatie vindt nu plaats van inputunits rechtstreeks naar de outputunits. Op de outputunits is een teaching signaal aanwezig. Aldaar worden de errors, en dat zijn de δ_j's, berekend d.m.v. het vergelijken van teachingvalue en activationvalue (zie bovenstaande eerste formule voor δ_j). De gewichten tussen de hoogste hiddenlaag en outputlaag kunnen worden aangepast. Nu kan de tweede fase beginnen. De errors die aan de outputlaag zijn berekend kunnen nu worden terug gepropageerd via hiddenunits naar de inputunits. Bij elke stap worden er nieuwe δ's berekend volgens bovenstaande tweede formule voor δ_j en kunnen er weer verbindingen worden aangepast.

Via deze methode van trainen worden na elk trainingsvoorbeeld de δ_j's, en dit zijn dus de fouten aan de outputlaag of de gepropageerde fouten in de hiddenlaag, kleiner gemaakt. We moeten de trainingsvoorbeelden net zo lang blijven aanbieden totdat de gewichten zich hebben gestabiliseerd. Hoewel elke training het netwerk een stap in de goede richting brengt, kan het toch zo zijn dat we via deze methode in een 'lokaal minimum' blijven hangen: de beste oplossing hoeft niet altijd te worden gevonden. In de praktijk blijkt dit probleem echter wel mee te vallen.

Er moet een keus worden gemaakt omtrent welke activatiefunctie moet worden gebruikt. In ieder geval moet deze functie continu en differentieerbaar zijn, anders kunnen we de leerregel niet uitvoeren. De discontinue thresholdfuncties vallen dan dus al af. Verder moet, zoals hiervoor al is vermeld, de activatiefunctie niet-lineair zijn anders heeft het meerdere lagen principe geen zin. De eenheidsfunctie valt hiermee ook af. Wat we overhouden is de sigmoïdefunctie, in dit verband ook wel de 'logistic activation function' genoemd:

$f_j(net_j) = a_j = 1 / (1 + EXP(-(net_j + \Theta_j)))$

Θ_j is een threshold voor de netwaarde. De afgeleide van f is voor déze functie zeer gemakkelijk: $f_j'(net_j) = a_j * (1 - a_j)$. Merk op dat het maximum voor $f_j'(net_j)$ ligt op a_j=0.5 en het minimum op a_j=0 of a_j=1. Dit impiceert dat de fout δ_j het grootst wordt voor a_j=0.5 en het kleinst voor a_j=0 en a_j=1. Dit betekent dus dat de gewichten de grootste verandering ondergaan als de activatiewaarden het onduidelijkst zijn.

Een ander aspect aan deze leerregel is de leersnelheid μ. In theorie is het zo dat pas met kans één de fout tijdens het trainen wordt verminderd als we oneindig kleine stapjes nemen. In de praktijk kan hij gelukkig groter worden gekozen. We moeten hem zó groot kiezen dat er net geen oscillatie optreedt. (Bij oscillatie zit er geen zinnige lijn meer in de gewichtsveranderingen. Er vindt geen convergentie van de gewichten plaats.) Dan verloopt het leren maximaal snel (met natuurlijk wel een kans dat we in een lokaal minimum terecht komen). Een manier om oscillatie tegen te gaan, om daarmee de leersnelheid op te kunnen voeren, is het invoeren van een 'momentum' term. De leerregel wordt dan:

$[W_{ij}\text{-nieuw}(n+1) - W_{ij}\text{-oud}(n+1)] = \mu * \delta_j * a_i + \alpha * [W_{ij}(n) - W_{ij}(n)]$

Er is nu bij de gewichtsverandering een tijdsindicatie n opgenomen om de $[W_{ij}\text{-nieuw} - W_{ij}]$'s van elkaar te kunnen onderscheiden.

$\alpha * [W_{ij}(n) - W_{ij}(n)]$ is de momentum term. De nieuwe waarde van $\alpha * [W_{ij} - W_{ij}]$ (óf eigenlijk: a $* [W_{ij}(n+1) - W_{ij}(n+1)]$ wordt nu dus afgeleid van de oude. Hierdoor wordt de ‘fouten-ruimte’ als functie van de gewichten , nauwkeuriger afgezocht. De gewichtsveranderingen kunnen niet meer kriskras plaats vinden. Het leren blijkt nu sneller te kunnen verlopen, hoewel een

kenmerk van de error back propagation Perceptron blijft dat het leren erg langzaam geschiedt. Dit type Perceptron heeft daarom vooralsnog geen real-time toepassingen.

2.4.4 Constraint Satisfaction Models

De Harmony Theorie [Smolensky, 1986], het Schema model [Rumelhart&Smolensky, 1986] en de Boltzmann machine [Hinton, 1986], allen 'Constraint satisfaction Models', zijn één van de eerste serieuze pogingen een werkelijk onderscheid te gaan maken tussen een subsymbolisch en een symbolisch niveau. In paragraaf 2.3 hebben we gezien dat de probleembeschrijving op het subsymbolisch niveau plaats vindt in termen van soft-constraints, terwijl op het symbolisch niveau toch iets als hard-constraints zijn te herkennen. Beschouw bijvoorbeeld een probleem waarbij aan vele constraints voldaan moet worden. Vaak is het zo dat niet aan alle constraints voldaan kan worden. In paragraaf 2.3 hebben we immers gezien dat het eigenlijk onmogelijk is de werkelijkheid waarop het probleem van toepassing is, consistent en ondubbelzinnig te beschrijven in termen van een grote verzameling onaantastbare regels. Nu is het dus de bedoeling dat, na een waarneming, bepaalde constraints kunnen worden verworpen en dat aan andere constraints wel wordt voldaan. Hoe beslis je nu welke constraints na deze waarneming belangrijk zijn en welke niet? Dat is de kernvraag die Minsky en Papert in 1969 de 'best match problem' noemden en in de PDP handboeken 'constraint satisfaction problem' wordt genoemd. De methodes waarmee men tracht dit probleem op te lossen worden 'constraint satisfaction models' genoemd. De Harmony Theorie, door P. Smolensky in 1986 gepubliceerd, is ontwikkeld voor het oplossen van dergelijke problemen. Het schema model en de Boltzmann machine zijn tegelijkertijd ontwikkeld met de Harmony Theorie en hebben daardoor veel met elkaar gemeen.

Een schema (meervoud: schemata) wordt door veel theoretici beschouwd als de basisbouwsteen voor het begrijpen van cognitie. Een schema is te vergelijken met begrippen uit de traditionele AI als scripts en frames. Schemata zijn geen losse concrete elementen in een netwerk, maar eigenschappen van het totale netwerk. Er kan bijvoorbeeld worden gesproken over schemata van kamers, van restaurants, van over straat lopen, etc. Bevindt men zich in een bepaalde situatie, dan zal een bepaald algemeen concept in werking treden en zich aanpassen aan die specifieke situatie.

In [Smolensky, 1986] wordt hier een concreet voorbeeld van gegeven. Wij hebben allen een soort algemeen idee van een restaurant. Als we een restaurant binnen stappen, hebben we een bepaalde verwachting van de omgeving en we passen ons gedrag daarbij aan. Dit noemt Smolensky het Restaurant-script. Stel nu dat we na het voorgerecht hoofdpijn krijgen. Nooit eerder hebben we bijvoorbeeld hoofdpijn in een restaurant gehad, dus een soort restaurant-hoofdpijn script is niet expliciet aanwezig. Toch zijn we prima in staat dit probleem op te lossen door aan de bediende een aspirientje te vragen. De twee afzonderlijke en algemene restaurant- en hoofdpijn scripts gaan in elkaar over, althans zo lijkt het, om een specifiek restaurant-hoofdpijn script te vormen.

Scripts of schemata zijn dus niet expliciet aanwezig. Op een subsymbolisch niveau worden constraints gedefinieerd, maar door de flexibiliteit in het gebruik ervan zijn er verschillende oplossingen van de problemen mogelijk. Deze verschillende oplossingen, die in principe onvoorspelbaar kunnen ontstaan en waarvan het zo is dat ze zo goed mogelijk bij de waarneming aansluiten, worden schemata genoemd. De Boltzmann machine en de Harmony Theorie zijn modellen voor het vinden van die oplossingen, dus die schemata, die zo goed mogelijk bij waarneming en constraints aansluiten.

In het Schema model, maar dat geldt ook voor de Harmony Theorie en de Boltzmann machine, is het zo dat een node voor een hypothese, en een verbinding tussen twee nodes voor de constraint betrekking hebbende op beide hypothesen staat. Het Schema model is verder continu: de nodes kunnen alle waarden in het interval [0,1] aannemen. Dit in tegenstelling tot de Harmony Theorie en de Boltzmann machine alwaar de nodes slechts discrete waarden kunnen aannemen. Verbindingen zijn symmetrisch, d.w.z. $W_{ij}=W_{ji}$, en nodes kunnen niet met zichzelf worden verbonden: $W_{ii}=0$. De updaterule voor het Schema model is de volgende:

$$a_i(t+1) = a_i(t) + net_i(1-a_i(t)) \text{ als } net_i >= 0$$
$$a_i(t+1) = a_i(t) + net_i * a_i(t) \text{ als } net_i < 0$$

Hier staat dus dat de activationvalue wordt verhoogd als de netvalue positief is en wordt verlaagd als de netvalue negatief is. Een stabiele situatie is bereikt als $net_i=0$ (a_i kan dan alle waarden in het interval [0,1] hebben aangenomen en er is blijkbaar evenveel stimulering als afremming op node i) of als ai=1 en neti20 (dus de netfunctie stimuleert het toenemen van de activationvalue, maar de bovengrens is bereikt) of als $a_i=0$ en $net_i>=0$ (de ondergrens is bereikt). Doordat de activationvalue zich altijd ontwikkelt in de richting van datgene wat de netvalue hem adviseert, dus datgene wat de aansturende nodes hem gemiddeld adviseren, streeft het systeem voortdurend naar toenemende consensus. Voor de mate van consensus hebben we al begrippen gezien als 'energie' , 'harmony' , maar er is er nog één: ‘goodness of fit’ , kortweg 'goodness'. Dit begrip wordt gebruikt voor het schema model en ook voor de Boltzmann machine. Het kan worden gedefinieerd als:

$$goodness_j = \Sigma_i W_{ij} * a_i * a_j$$

(Voor de volledigheid: Deze goodness-formule heeft deze vorm indien node j geen threshold heeft). De goodness van het totale netwerk is de som van de goodness van alle nodes afzonderlijk. Hoe sterker dus het gewicht, hoe meer het voor de goodness van belang is dat node i en node j zo sterk mogelijk worden geactiveerd.

Een belangrijk nadeel van het Schema model is het feit dat er rechtstreeks naar een zo hoog mogelijke goodness (of zo laag mogelijke energie) wordt gestapt zodat we gemakkelijk in een lokaal optimum kunnen geraken. In de Boltzmann machine wordt dit probleem aangepakt op

een manier zoals die ook in de Harmony Theorie is opgelost, namelijk via de methode van 'Simulated Annealing'. Later wordt hierop, in het kader van het afstudeeronderzoek, uitgebreid terug gekomen, maar nu wordt alvast de essentie van Simulated-Annealing aangestipt.

Iedere node moet op grond van lokale overwegingen beslissen hoe een, globaal gezien, zo hoog mogelijke goodness te bereiken. Dit lijkt onmogelijk als we er niet iets globaals bij halen. M.b.v. de fysica is hier het volgende op gevonden: een globale temperatuur T. Tijdens het zoeken naar een stabiele situatie laten we de temperatuur langzaam aflopen tot uiteindelijk T=0: het vriespunt. Bij hoge T bepalen de nodes onafhankelijk van hun input de activation en daarmee de output. Naarmate T afneemt, zal de invloed van de input op de activationvalue van de node toenemen. Uiteindelijk, bij T=0, wordt het gedrag van de activationvalue volledig bepaald door de input. We zitten dan in dezelfde situatie van redeneren als in het schema model: de beweging van de activationvalues is rechtstreeks in de richting van het dichtstbij zijnde lokale optimum. Doordat echter het systeem voor T> 0 de mogelijkheid heeft gekregen ook andere uithoeken van het 'goodness-landschap' te onderzoeken wordt de kans groter dat we nu het, of een, globaal optimum bereiken. Een lokaal optimum kan worden geïnterpreteerd als clusters van nodes die elkaar bevestigen in hun keuze zonder dat zij op elkaar nog invloed kunnen uitoefenen. Lokaal, dus binnen de clusters zelf, is een goede oplossing gevonden, maar globaal, dus op het niveau van de clusters onderling, hoeft dat niet zo te zijn. Het doel van de temperatuurregeling is nu dat bij hoge T de clusters worden 'losgeweekt': de activatiekeuzes van de nodes is volledig random. T moet vervolgens dermate langzaam afnemen dat elke lokale keuze de tijd krijgt dóór te werken in de rest van het netwerk, zonder dat de nodes meteen op hun keuze worden vastgepind. Bij lage T, zal het netwerk langzaam zijn positie innemen om uiteindelijk in T=O te bevriezen. De formule voor het wegen van het belang van de input als functie van de temperatuur is dan ook een stochastische functie:

$P(a_i=1) = 1 / (1 + EXP(-net_i/T))$.

Voor $T=\infty$ geldt $P(a_i=1)=0.5$, ongeacht de waarde van net_i.
Voor T=0 geldt $P(a_i=1)=0$, als $net_i<0$ en
$P(a_i=1)=1$, als $net_i>0$.

Zie ook figuur I.f uit bijlage I voor de schets van deze functie.

Een verschil tussen Harmony Theorie en Boltzmann machine is het feit dat binnen de eerste hardere eisen worden gesteld aan de netwerkstructuur. In de Harmony Theorie zijn er precies twee lagen van nodes: de 'representational features' en de 'knowledge atoms'. Dit is te zien in figuur 2.4.l.

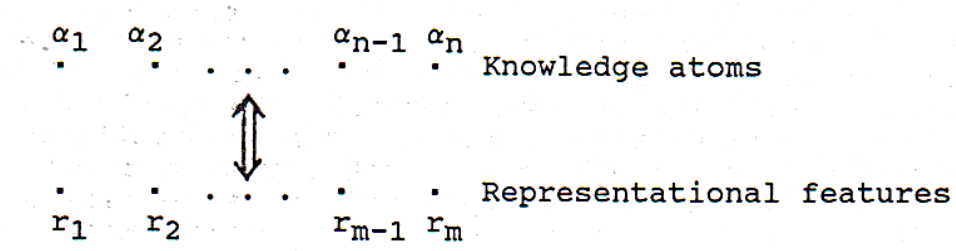


Fig. 2.4.l Nodestructuur in de Harmony Theorie

Op het niveau van de features vinden waarnemingen plaats, die als het ware worden gecodeerd binnen de atoms. Iedere atom staat voor een patroon van features. Voor welk patroon ieder atom specifiek staat, wordt gespecificeerd d.m.v. de verbindingen tussen features en atoms. De atoms dienen als schakel voor het completeren van onvolledige featurepatronen. Er ontstaat een ware abstractiehiërarchie als de gecompleteerde features via weer andere atoms op hun beurt dienen als ingang voor de completering van weer andere features. Zie figuur 2.4.m.

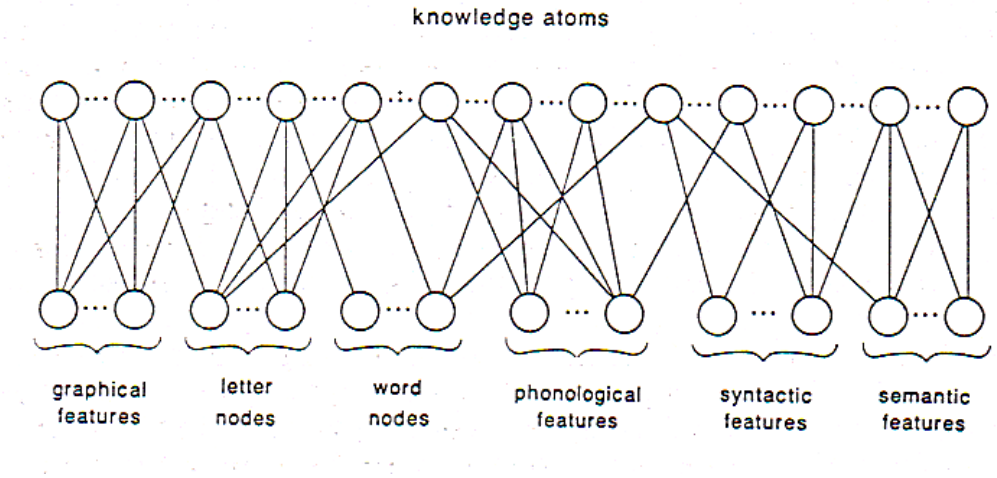


Fig. 2.4.m. Een netwerk voor het representeren van woorden in een abstractiehiërarchie.

Waarneming vindt nu plaats op de 'exogenous features' (in bovenstaande figuur zijn dit de line-segment nodes). Via de knowledge atoms leidt dit tot conclusies op de abstractie van de 'endogenous features' (de letter nodes). Via weer een laag knowledge atoms leidt dit de conclusie op het niveau van de word nodes. Exogenous features zijn features die in contact met de invoer van de buitenwereld staan. Endogenous features staan voor de features die de tussenresultaten en het eindresultaat representeren.

Zoals reeds vermeld zijn de verbindingen symmetrisch en ze lopen slechts tussen beide lagen en niet binnen de lagen zelf. Iedere redenatieslag bestaat daardoor uit twee stappen: op grond van de waarneming berekenen eerst alle atoms hun waarde, vervolgens alle features, dan weer alle atoms, etc. De structuur van een Boltzmann machine is niet zo scherp gedefinieerd. Bij de Boltzmann machine kan er ook gebruik worden gemaakt van hidden nodes (meestal: hidden units), waardoor ten eerste het netwerk zijn eigen interne representatie kan bouwen maar vooral ten tweede dat er complexere bewerkingen, die niet met twee lagen mogelijk zijn, uitgevoerd kunnen worden.

Binnen de Harmony Theorie zit het leren in het aanpassen van de sterktes o van de knowledge atoms. De sterkte van een atom is een maat voor de frequentie waarmee het patroon, waar het atom voor staat, is waargenomen. Functioneel gezien ligt die sterkte bij het atom, in de praktijk wordt die echter geïmplementeerd in de verbindingen naar de atom. Het leren bestaat uit twee stappen: een observatiestap en een simulatiestap. Als aan de exogenous features de waarneming is aangeboden en na verloop van de redenatie is gecompleteerd bij de endogenous features, vindt de observatie-leerstap plaats: alle atoms die de verkregen resultaten over de features ondersteunen worden versterkt met een kleine waarde dσ. Omdat aan deze leerregel geen bovengrens is verbonden t.a.v. de sterktes σ, vindt een tweede leerstap plaats: de simulatie-leerstap. Men laat het netwerk dan zijn instelling vinden, zonder dat er input van de buitenwereld is op de exogenous features. De resultaten die dan verschijnen produceren precies het tegenovergestelde effect, namelijk een afname met do van de actieve atoms. Het idee hierachter is het feit dat de gewichten zich zullen stabiliseren als ze evenveel worden geactiveerd te groeien met dσ in de observatiefase, als af te nemen met dσ in de simulatiefase. De simulator zal dan de observatie, dus de buitenwereld, precies beschrijven.

Voor de Harmony Theorie kan worden bewezen dat de optimale verzameling gewichten wordt verkregen als er geen endogenous features aanwezig zijn. Voor het geval dat ze wel aanwezig zijn, zoals in figuur 2.4.m, bestaat geen bewijs dat dan óók die beste verzameling wordt gevonden.

Het leerproces bij de Boltzmann machine lijkt hier veel op, maar bestaat niet uit twee maar één leerfase. De leerfase vindt echter pas plaats als er toch weer twee stappen zijn uitgevoerd. Eerst worden input- en outputnodes op een bepaalde waarneming *geclampt* en kan het annealing-proces aanvangen. Deze temperatuurdaling duurt voort tot het moment dat bijna het evenwicht is bereikt. Omdat het systeem nog niet is bevroren, veranderen de hidden nodes nog van waarde. Van de waarden die de nodes dan aannemen worden statistieken vastgelegd. Deze statistieken betreffen kansen voor de verbanden in activiteit tussen naburige nodes. Deze procedure wordt voor verschillende waarnemingen uitgevoerd. Alle kansverdelingen worden vervolgens gemiddeld en resulteert in een eindkansverdeling

voor alle verbindingen. Tijdens de tweede stap wordt op dezelfde manier ook deze kansverdeling bepaald als gevolg van een ongeclampte waarneming.

Kullback heeft in 1959 een formule opgesteld voor de mate waarin twee kansverdelingen met elkaar overeenkomen. Deze mate van overeenkomst wordt uitgedrukt in een getal G: hoe lager G, hoe meer overeenkomst in de kansverdelingen. Het gaat in dit verband te ver om die formule hier uit te werken maar er kan aan de hand van deze formule worden bewezen, en daar gaat het om, dat lokaal, dus op het niveau van de nodes, bekend is hoe de gewichten moeten worden veranderd om globaal een zo laag mogelijke G te bereiken. Welnu, we hebben nu twee kansverdelingen: het resultaat van de ongeclampte test en van de geclampte test. Voor iedere verbinding W_{ij} hebben we respectievelijk een kans p^+_{ij} en een kans p^-_{ij}. De verandering van G als functie van het gewicht W_{ij} blijkt nu evenredig te zijn met het verschil van p^+_{ij} en p^-_{ij}. Als we daar nog de temperatuur T aan toevoegen om de verbindingen niet rechtlijnig G te laten minimaliseren, en daarmee het eerste de beste lokale minimum trachten te ontlopen, ontstaat uiteindelijk de volgende formule:

$$\delta G/\delta W_{ij} = -1/T * [p^+_{ij} - p^-_{ij}]$$

Hiermee is de training voor alle verbindingen, dus ook voor de verbindingen tussen hidden nodes, gedefinieerd.

Via bovenstaande methodes voor het opbouwen van een netwerkstructuur kan op een zeer subtiele wijze zaken als schemata worden aangeleerd. Door de ingewikkelde kennisstructuur van zo'n schema is het ondoenlijk deze expliciet te definiëren, zoals dat binnen de traditionele AI nog wel gebeurt. Op basis van waarnemingen worden de schemata langzaam gewijzigd en kunnen ze perfect worden aangepast aan de observaties in de buitenwereld: symbolen zijn geen harde symbolen, maar zijn een zeer flexibel resultaat van de samenwerking van vele subsymbolen.

Voorgaande behandeling van Constraint Satisfaction Models is niet volledig. Aan vooral de Boltzmann machine en Harmony Theorie ligt een uitgebreide wiskunde ten grondslag. Binnen het kader van dit rapport heeft het weinig zin om daar op in te gaan. Dat neemt niet weg dat deze theorieën zeer belangrijk kunnen zijn voor verder onderzoek naar Connectionist Systems. Voor de geïnteresseerde lezer is in bijlage I een uitgebreidere behandeling opgenomen van de Harmony Theorie.

2.5 Een Connectionistisch Expertsysteem

In het artikel [Gallant, 1986] wordt een volwaardig, in principe commercieel toepasbaar 'Connectionist Expert system" gepresenteerd. De belangrijkste facetten ervan zullen worden besproken. Het zal blijken dat die volwaardigheid gerelateerd is aan wat van de huidige generatie expertsystemen verwacht wordt, waardoor er veel traditionele AI-technieken in dit

systeem zijn terug te vinden. De eis van commerciële toepasbaarheid heeft verder geleid tot, overigens zeer interessante maar soms, vergaande concessies t.a.v. het Connectionist AI-Paradigma. Dit type expertsysteem is, concluderend uit wat mij heeft bereikt, één van de weinigen in zijn soort.

Het systeem van Gallant is geïmplementeerd in een softwarepakket bestaande uit twee delen. Het eerste is een 'network knowledgebase generator' en het tweede deel is een 'stand-alone expert system inference engine' die de knowledgebase interpreteert. Het tweede deel heeft de naam MACIE, acroniem voor MAtrix Controlled Inference Engine, gekregen. Het is waarschijnlijk het meest eenvoudig om het systeem en de principes ervan toe te lichten aan de hand van een voorbeeld. Binnen het kader van dit rapport gaat het erom dat n.a.v. het hiernavolgende een indruk kan worden verkregen omtrent één van de mogelijke interpretaties van de vernieuwde inzichten in Connectionist Systems. Te veel technische details is zoveel mogelijk vermeden. In plaats daarvan is een eigen abstracte formulering gevonden die hopelijk meer tot de verbeelding zal spreken.

Het voorbeeld dat wordt besproken handelt over 'acute theoretical diseases of the sarcophagus'. In het artikel van Gallant wordt niet uitgelegd wat dat precies betekent, maar waarschijnlijk heeft het, hoe luguber ook, te maken met de ziektes die kunnen ontstaan door bacteriën die vrij komen na het openen van lang gesloten sarcofagen. Hoe dan ook, het is een leuk probleem voor expertsystemen. Het model dat voor deze ziektes is opgesteld bevat symptomen, ziektes en behandelingsmethoden. Hoewel het een vrij eenvoudig model is, schijnt het meer dan 85% van de kennis over het zeer gespecialiseerde domein te vangen.

Er zijn 6 symptomen, 2 ziektes en drie behandelingsmethoden. Aan de hand van een afhankelijkheidsgraaf wordt gespecificeerd wat van wat afhangt. In figuur 2.5.a is de afhankelijkheidsgraaf in matrixnotatie weergegeven. Een nul specificeert de onafhankelijkheid en een één de afhankelijkheid tussen twee units. Iedere unit staat voor een specifieke symptoom, ziekte of behandelingsmethode. Totaal zijn er dus 11 units, genummerd van u_1 t/m u_{11}. De betekenissen zijn, en voor het gemak blijven deze in het Engels staan, ook opgenomen in figuur 2.5.a.

Symptomen

u_1 : Swollen feet

u_2 : Red ears

u_3 : Hair loss

u_4 : Dizziness

u_5 : Sensitive aretha

u_6 : Placibin allergy

Ziektes

u_7 : Supercilliosis

u_8 : Namastosis

Behandelingsmethoden

u_9 : Placibin

u_{10} : Biramibio

u_{11} : Posiboost

Afhankelijkheidsgraaf:

u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}	
1	1	1	0	0	0	0	0	0	0	0	$\rightarrow u_7$
0	0	1	1	1	0	0	0	0	0	0	$\rightarrow u_8$
0	0	0	0	0	1	1	1	0	0	0	$\rightarrow u_9$
0	0	1	0	0	0	1	1	0	0	0	$\rightarrow u_{10}$
0	0	0	0	0	0	0	0	1	1	0	$\rightarrow u_{11}$

Grafisch kan dit worden weergegeven als in figuur 2.5.b.

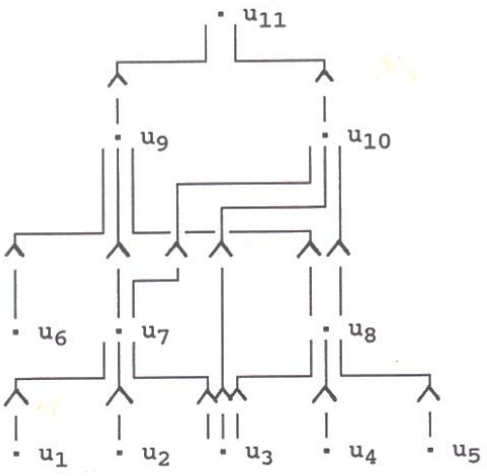


Fig. 2.5.b Het afhankelijkheidsnetwerk

u_1 t/m u_6 zijn de 'input cellen', u_7 en u_8 de ‘intermediate cellen' en de units u_9 t/m u_{11} de 'output cellen'. De begrippen ‘cellen’ en ‘units’ zijn synoniem.

Aan de hand van voorbeelden wordt het netwerk getraind. Iedere unit kan drie mogelijke waarden aannemen: +1, -1 of 0 voor respectievelijk waar, onwaar, of onbekend. Niet-Booleaanse data kan worden verkregen door Booleaanse variabelen te combineren. Het trainen van een netwerk berust op het principe van 'easy learning'. Daar wordt het volgende mee bedoeld. Aan het netwerk wordt kenbaar gemaakt welke combinaties van unittoestanden geoorloofd zijn: ieder trainingsvoorbeeld bevat correcte waardecombinaties. In het geval van easy learning worden daarbij ook de toestanden van de intermediate cellen (in dit geval de ziektes) gespecificeerd. Binnen de methode van 'hard learning' moet het systeem zelf uitzoeken wat geschikte waarden voor de intermediate cellen zijn: trainingen vinden alleen plaats in termen van input-output. Hierover is al gesproken in paragraaf 2.4. Intermediate cellen nemen nu de plaats in van hidden units. Easy learning resulteert in een wat gebruikersonvriendelijker systeem (de gebruiker moet nu immers in de trainingsvoorbeelden ook specificeren wat de waarden moeten zijn van de intermediate cellen), maar is eenvoudiger te implementeren en geeft meestal een sneller en betrouwbaarder systeem dan na hard learning. Een ander belangrijk argument voor de keuze van easy learning is er één die typisch is voor expertsystemen. De intermediate cellen hebben nu, in tegenstelling tot de hidden units uit de vorige paragraaf, betekenis. In de trainingsvoorbeelden willen we die dan ook kunnen specificeren.

In figuur 2.5.c is een voorbeeld gegeven van een training. In dit geval zijn 8 'training exempels' TE#1 t/m TE#8 aangegeven. Voor iedere unit is zijn waarde gespecificeerd.

u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}	
1	1	1	-1	0	-1	1	-1	1	-1	1	TE#1
-1	-1	-1	1	1	-1	-1	1	1	1	-1	TE#2
-1	-1	1	1	-1	1	1	1	-1	-1	-1	TE#3
1	1	-1	-1	1	-1	-1	-1	-1	-1	-1	TE#4
1	-1	0	1	1	1	1	1	-1	1	1	TE#5
1	-1	-1	1	1	-1	1	1	1	1	-1	TE#6
1	1	1	-1	-1	1	1	-1	-1	-1	-1	TE#7
-1	1	1	-1	1	1	-1	1	-1	-1	-1	TE#8

Fig. 2.5.c. Acht trainingsvoorbeelden TE#1 t/m TE#8

Iedere cel of unit is in staat tot het uitvoeren van een eenvoudige functie die Gallant noemt: de 'lineair discriminant functie' , en die in de vorige paragraaf is geïntroduceerd onder de naam 'hard limiter' of 'threshold functie'. In dit geval is de threshold nul:

$S_i = \Sigma_{j>=0} W_{i,j} * u'_j$

u'_j = +1 als $S_j > 0$
=1 als $S_j < 0$
-1 als $S_j = 0$

$W_{i,j}$ € [.., -1, 0, 1, ..]

$W_{i,j}$ is de sterkte van de verbinding van unit j naar unit i (let op: in de rest van dit rapport wordt de betekenis van de indices precies andersom gebruikt). In feite zijn de units dus niets anders dan kleine Perceptrons. De afspraak is dat de cellen worden genummerd vanaf u_1. U_0 heeft de vaste waarde +1. $W_{i,0}$ wordt daarom de 'bias' genoemd van cel i. Beschouw figuur 2.5.d.

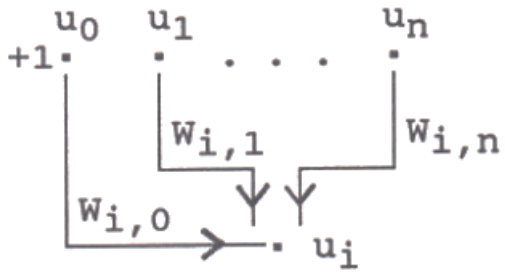


Fig. 2.5.d. Schematische weergave van de invloed van units j op unit i

Het netwerk mag geen gerichte loops bevatten. Dit impliceert dat er geen iteratieproces kan ontstaan. Vanaf de units op het laagste niveau worden de waarden naar boven toe doorgegeven om uiteindelijk te resulteren in een conclusie. Iedere unit hoeft dus slechts éénmaal zijn waarde uit te rekenen om die vervolgens door te geven aan de van hem afhankelijke unit(s).

De verbindingen moeten nu een dusdanige waarde krijgen opdat alle trainingsvoorbeelden op de juiste manier verwerkt zijn. D.w.z. dat wanneer bepaalde input, in dit geval symptomen, wordt aangeboden het netwerk na bovenstaand type redenatieproces ziektes en behandelingsmethoden concludeert als in de trainingsvoorbeelden gespecificeerd.

Het principe van dit redenatieproces moet verder onder de loep worden genomen. Beschouw het netwerk van figuur 2.5.e.

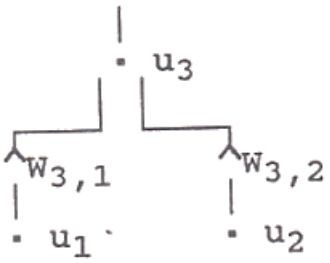
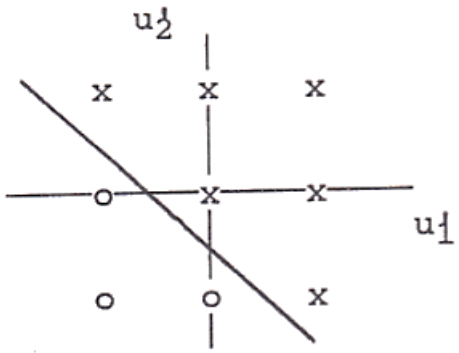


Fig. 2.5.e Een eenvoudig netwerk met drie units

Dan geldt dus (waarbij $W_{3,0}$ wederom de bias is):

$$u'_3 = \begin{cases} +1 \text{ als } W_{3,0} + W_{3,1} * u'_1 + W_{3,2} * u'_2 > 0 \\ -1 \text{ als } W_{3,0} + W_{3,1} * u'_1 + W_{3,2} * u'_2 < 0 \\ 0 \text{ als } W_{3,0} + W_{3,1} * u'_1 + W_{3,2} * u'_2 = 0 \end{cases}$$

Een unit verzorgt dus een classificeringsfunctie. In de figuur 2.5.f is een voorbeeld gegeven voor welke waarden van u'_1 en u'_2, u'_3 min één (aangegeven door een rondje) ofwel één (aangegeven door een kruisje) wordt.



Het snijpunt van deze lijn met de u'_1-as is: $-W_{3,0}/W_{3,1}$.
Het snijpunt van deze lijn met de u'_2-as is: $-W_{3,0}/W_{3,2}$.

Fig. 2.5.f De grafische weergave van welke waarden van u'_1 en u'_2, u_3 doen activeren

De lijn kan, doordat W_{ij} zowel positief als negatief kan zijn, op alle mogelijke manieren de relevante punten (dat zijn alle combinaties van u'_1 en u'_2, dus de kruisjes en de rondjes in figuur 2. 5.f) verdelen in twee categorieën (u'_3=-1: de rondjes en u'_3=+1: de kruisjes) en een twijfelcategorie (u'_3=0, punten die op de lijn liggen).

Als de kruisjes en rondjes zo verdeeld zijn als in figuur 2.5.f, dan kunnen we dus eenvoudig waarden voor $W_{3,0}$, $W_{3,1}$ en $W_{3,2}$ vinden opdat zij van elkaar worden gescheiden. Vaak echter zijn de punten niet lineair scheidbaar. Een voorbeeld hiervan is opgenomen in figuur 2.5.g.

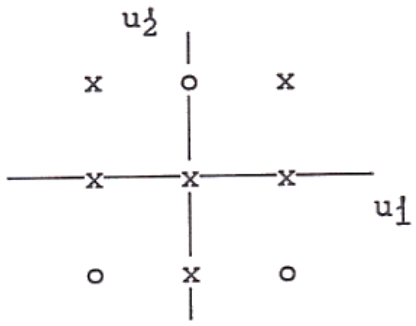


Fig. 2.5.g. Een voorbeeld van een niet lineair scheidbaar probleem

De kruisjes en rondjes in figuur 2.5.g zijn niet scheidbaar door één enkele lineaire functie. Dit probleem kan echter toch worden opgelost door bijvoorbeeld drie extra units (u_A,u_B,u_C) te introduceren. De constructie kan er dan als volgt uitzien. Hierbij is in de units zélf de bias weergegeven.

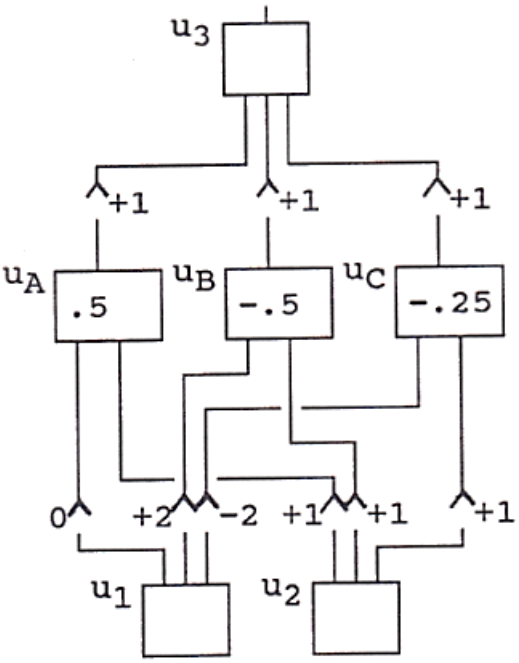


Fig. 2.5.h. Een oplossing van de scheiding van de kruisjes en rondjes van figuur 2.5.g

Ofwel:

$$u_A = \begin{cases} +1 \text{ als } u'_2 + .5 > 0 \\ -1 \text{ als } u'_2 + .5 < 0 \\ 0 \text{ als } u'_2 + .5 = 0 \end{cases}$$

$$U_B = \begin{cases} +1 \text{ als } u'_2 + 2 * u'_1 - .5 > 0 \\ -1 \text{ als } u'_2 + 2 * u'_1 - .5 < 0 \\ 0 \text{ als } u'_2 + 2 * u'_1 - .5 = 0 \end{cases}$$

$$U_C = \begin{cases} +1 \text{ als } u'_2 - 2 * u'_1 - .25 > 0 \\ -1 \text{ als } u'_2 - 2 * u'_1 - .25 < 0 \\ 0 \text{ als } u'_2 - 2 * u'_1 - .25 = 0 \end{cases}$$

Dat dit netwerk op de juiste manier de combinaties van u'_1 en u'_2 van elkaar scheidt is als volgt in te zien. Als functie van u'_1 en u'_2 zien u'_A, u'_B en u'_C er namelijk uit als in figuur 2.5.i.

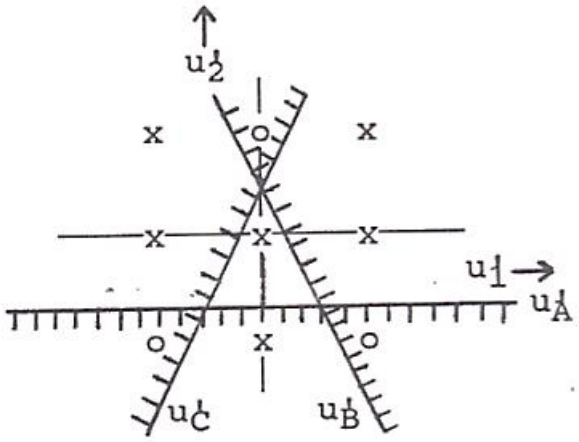


Fig. 2.5.i Een oplossing voor het niet lineair scheidbare probleem

Er zijn arceringen aangegeven aan de kant waar u'_A, u'_B en u'_C = +1. Op de lijnen zélf zijn ze nul en aan de andere kant ervan één. Aangezien de verbindingen van de units u_A, u_B en u_C naar de conclusie unit u_3 allen één zijn (zie figuur 2.5.h), is S_3 simpelweg de som van de outputs van u_A, u_B en u_C. Doordat we te maken hebben met drie scheidingslijnen ontstaan er 6 gebieden waarvoor we die waarde van S3 kunnen uitrekenen. Ze zijn in figuur 2.5.j vetgedrukt aangegeven.

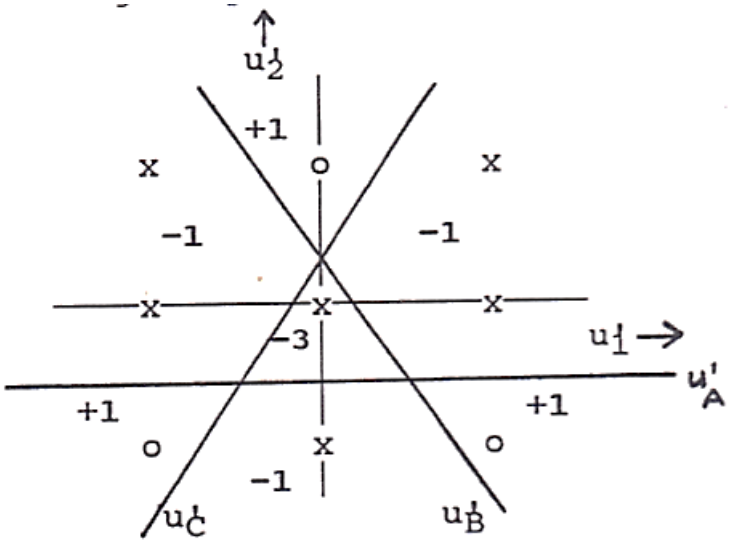


Fig. 2.5.j De waarden van S_3. Als $S_3>0$, dan bevindt zich daar een rondje, als $S_3<0$ een kruisje. Hiermee is het niet lineair scheidbare probleem opgelost d.m.v. drie extra units.

Aangezien:

$u'_3 =$ +1 als $S_3 > 0$
-1 als $S_3 < 0$
0 als $S_3 = 0$ (deze situatie doet zich echter niet voor)

is het inderdaad zo dat via bovenstaande keuze van u_A, u_B en u_C de kruisjes van de rondjes worden gescheiden: daar waar S_3 positief is, liggen de rondjes en daar waar negatief, de kruisjes.

Via deze methode van lineaire scheiding werkt ook het expertsysteem van Gallant. Uitgaande van de afhankelijkheidsgraaf en de eventueel toegevoegde extra units wordt er een leeralgoritme toegepast dat er voor moet zorgen dat alle trainingsvoorbeelden op bovenstaande manier worden verwerkt. Dit leeralgoritme, dat de naam 'pocket algoritme' draagt, is opgenomen in bijlage II. Dit is een eenvoudig iteratief algoritme dat net zo lang de trainingsvoorbeelden (in random volgorde en zo nodig meerdere malen) tot zich neemt, totdat alle voorbeelden zo goed mogelijk in de verbindingen zijn verwerkt. Om in bovenstaande terminologie te blijven: het leren berust op het geduldig verschuiven van de scheidingslijnen totdat alle mogelijke conclusies op grond van de input kunnen worden onderscheiden.

Dit leeralgoritme losgelaten op de trainingsvoorbeelden uit figuur 2.5.c binnen het netwerk als gespecificeerd door het afhankelijkheidsnetwerk van figuur 2.5.a blijkt te resulteren in een netwerk waarvan de grafische representatie is opgenomen in bijlage III.

De eerste stap die Gallant beschrijft is dus het genereren en trainen van het netwerk. De tweede stap is als volgt. Het netwerk wordt nu namelijk aangeboden aan MACIE, het redeneermechanisme. MACIE heeft verschillende taken:

1. Redeneren op grond van onvolledige informatie.
2. Het vinden van onbekende input variabelen die de sleutel zijn voor het heropstarten van een vastgelopen inferentieproces.
3. Produceren van justificaties voor de inferentie.

MACIE gaat procedureel te werk:

(1) Verkrijg de initiële informatie voor het netwerk. D. w. z. geef de inputs hun waarde.

(2) Start het inferentieproces op. Dit houdt in dat de cellen één voor één hun waarde uitrekenen. Dit dient te geschieden vanaf input- in de richting van de outputunits. Van te voren moet de expert een unit afhandelingsvolgorde definiëren opdat bij elke unitafhandeling alle benodigde informatie steeds klaar staat. Het inferentieproces berust op pure forward reasoning.

Bij de berekening van de outputfunctie van een unit i wordt ook bekeken of er niet te veel units de waarde 0 (onbekend) hebben: indien die units theoretisch het resultaat van unit i kunnen laten omslaan, dan wordt stap (4) uitgevoerd, anders (3).

(3) Stop. Alle outputvariabelen hebben een legitieme waarde gekregen.

(4) Er zijn te weinig gegevens aanwezig om het inferentieproces door te laten gaan. Er wordt nu een vorm van backward reasoning toegepast met als doel een 'onbekende' variabele te vinden waarvan het waarschijnlijk is dat die het redenatieproces weer kan opstarten. De waarde van deze variabele wordt aan de gebruiker gevraagd. Op verzoek van de gebruiker kan het systeem met een verklaring komen waarom betreffende variabele voor de inferentie van belang is. De verklaring wordt in een if-then regel gegoten.

(5) Het redenatieproces kan met deze informatie opnieuw worden opgestart. Ga dus naar (2).

Tot zover de principes van dit Connectionistische expertsysteem. Het is duidelijk dat vanuit de traditionele expertsysteem benadering de aanpak van Gallant zeer boeiend is. Deze aanpak komt op een bepaalde manier 'natuurlijk' over: in het voorgaande waren geen vreselijk

ingewikkelde algoritmes nodig om de principes ervan uit de doeken te doen. Het wiskundige fundament is minimaal, maar voldoende.

Het zal duidelijk zijn dat dit systeem van Gallant zowel Connectionistische als traditionele AI trekjes heeft. Allereerst moet worden opgemerkt dat de parallelle probleembeschrijving slechts dient als metafoor, maar geen praktische betekenis heeft. Het inferentieproces blijft sequentieel. Ook blijft er een soort besturing nodig die boven het geheel hangt om bijvoorbeeld de backward reasoning te ondersteunen: er liggen immers geen verbindingen terug. Dit ondersteund ook niet het idee van een parallel systeem.

Net als bij het Perceptron vindt er alleen forward reasoning plaats waardoor geen iteratieproces plaats vindt en dus geen stabiele situatie wordt ópgezocht. De symbolen (symptomen, ziektes en behandelingsmethoden) worden rechtstreeks gemanipuleerd: er is geen subsymbolisch niveau aanwezig. In het in dit rapport geschetste idee achter een Connectionist System is juist dit subsymbolisch niveau essentieel.

Toch springt het systeem wel al handig om met onvolledige, onzekere en inconsistente kennis. Waar bij een puur traditionele expertsysteem benadering nog allemaal ingewikkelde trucs voor dié vormen van kennis nodig zijn, is er binnen het basismodel van Gallant weinig extra aanpassing nodig: alleen een backward reasoning voor het oplossen van problemen met te onvolledige kennis.

Verbindingen tussen de units zijn, net zoals bij het Connectionist AI-paradigma, associaties tussen kenniseenheden waardoor een leeralgoritme eenvoudig is te implementeren. Toch zijn dit, net als bij het Perceptron, wederom hiërarchische associaties. De werking berust niet op associaties tussen kenniseenheden op een zelfde niveau waarbij er een totaalresultaat, dat wordt gevormd door al die kenniseenheden tezamen, als het ware naar boven komt borrelen.

Doordat Gallant dicht bij de traditionele benadering is gebleven zijn de gebruikersvragen nog steeds in termen van if-then regels te beantwoorden. Dit feit is vooral voor praktische toepassingen zeer belangrijk. Bij een puur Connectionistisch expertsysteem is het niet zo eenvoudig aan te geven waarom een inferentie eindigt bij een bepaalde oplossing. In ieder geval niet in termen van concrete symbolen.

Verder is het zo dat een Connectionistisch systeem altijd met een meest waarschijnlijke oplossing terugkomt, ook al is de kennis in principe ontoereikend. Gallant test hier voortdurend op en roept eventueel de hulp in van de gebruiker waardoor het probleem van onvoldoende kennis in zijn model niet kan voorkomen. Ook dit is een praktisch voordeel.

3 Probleemstelling

In het voorgaande hoofdstuk is getracht een beeld te schetsen van de huidige stand van zaken op het gebied van redeneersystemen. Daarin kwam, als het goed is, naar voren dat we voor wat betreft de benadering van cognitie op een keerpunt in de geschiedenis zitten. Misschien dat we in de toekomst een moment gaan bereiken dat we helemaal af zijn van het symbolisch paradigma, maar zover is het waarschijnlijk voorlopig nog niet. Op dit moment is er veel over het symbolisch- en nog erg weinig over het subsymbolisch paradigma bekend. Het laatste, zoals U in het vorig hoofdstuk heeft kunnen lezen, bevindt zich nog op een erg mistig, filosofisch vlak.

Het is grappig te zien dat nu het begrip 'Connectionist System' nog maar nauwelijks is geïntroduceerd, de commerciële applicaties ervan al over de toonbank rollen. De literatuur die de schrijver van dit rapport heeft bereikt was echter niet echt imponerend: het wezenlijke van het subsymbolisch paradigma zoals dat in hoofdstuk 2 is behandeld wordt niet of nauwelijks gebruikt. Het Connectionistisch Expertsysteem, behandeld in paragraaf 2.5, is daar een goed voorbeeld van.

In het licht van voorgaande is het volgende doel gecreëerd, namelijk het ontwerpen van een op de combinatie van het subsymbolisch en het symbolisch paradigma gebaseerd redeneersysteem. Het subsymbolisch paradigma wordt erin betrokken vanwege zijn vele, nieuwe en interessante aanknopingspunten voor redeneersystemen. Het symbolisch paradigma wordt niet verworpen on drie redenen. Ten eerste is er gewoon heel erg veel bruikbaars bekend geworden binnen het traditionele AI-onderzoek. Ten tweede moeten we niet de pretentie hebben om in één klap het symbolisch paradigma van tafel te vegen door een volwaardig subsymbolisch alternatief aan te bieden. Dat zou net iets te mooi zijn. Ten derde door het artikel [Feldman, 1988] waarin wordt beargumenteerd waarom de oplossing van onze technische problemen bij de huidige stand van zaken gezocht moet worden in de combinatie van beide paradigma's. Hij merkt bijvoorbeeld op dat de technische stand van zaken ver achterloopt op wat we nodig hebben voor een volwaardig Connectionist System. Feldman's conclusie geciteerd:

> *"De oplossing die de natuur in de vorm van massief parallelle rekenstructuren heeft gevonden zullen zonder twijfel bruikbaar zijn voor de computerwetenschappen, maar niet zonder modificatie."*

Feldman introduceert voor die modificatie (wat in feite neerkomt op de koppeling van beide paradigma's) het begrip: 'Het Gestructureerde Connectionist System'.

Zoals ook al in de inleiding vermeld, is in de doelstelling het woord 'expertsysteem' vermeden en daar het woord 'redeneersysteem' voor in de plaats gebruikt. Dat is gedaan om de eenvoudige reden dat 'expertsysteem' een nogal beladen term is. Met de reden in het achterhoofd waarom de traditionele AI-aanpak in een soort crisis is geraakt, namelijk doordat het zich heeft laten verleiden door de verschijning van iets moois als intelligentie, is het onverstandig doelen te leggen in een dergelijk esthetisch resultaat.

In [Hofstadter, 1988] wordt ook een aanval gedaan op de tendens binnen de AI naar opzichtige, modieuze domeinen. Hij merkt op dat dit niet het geijkte patroon is van fundamentele wetenschap. Het patroon is namelijk een verschijnsel proberen te isoleren en het tot zijn eenvoudigste vorm proberen terug te brengen. Hofstadter pleit dan ook voor wat hij noemt 'speelgoeddomeinen'. Hij zegt daar het volgende over:

> *"Naar mijn mening is de juiste domeinkeuze het belangrijkste besluit dat een AI-onderzoeker neemt wanneer deze aan het project begint. Als je je wilt mengen in de medische diagnostiek op expertniveau, verdrink je in een massa technische problemen die niets te maken hebben met de werking van de geest. Hetzelfde geldt voor andere zwaarwichtige domeinen die een rol spelen in de huidige expertsystemen. Maar als je zelf je domein bepaalt, het op maat kunt maken en bij snoeien zodat je de essentie van het probleem behoudt en bijkomstige aspecten kwijtraakt, heb je een goede kans iets fundamenteels te ontdekken."*

De doelstelling van dit rapport kan nu wat specifieker worden geformuleerd: op een abstract probleemdomein moet onderzoek worden gedaan naar de fundamenten van een gestructureerde Connectionist System, met als doel het maken van een model voor een redeneersysteem met een schuin oogje naar wat een volwaardig expertsysteem van zo'n model vraagt. De facetten die voor een volwaardig expertsysteem interessant zijn en ook in het model verwerkt zullen worden, zijn:

- Redeneren met onzekere waarnemingen
- Redeneren met onvolledige waarnemingen
- Redeneren met inconsistente waarnemingen
- Mogelijkheid tot kennisacquisitie
- Parallelle architectuur
- Kennis duidelijk voor handen, d.w.z. justificaties voor de inferentie

Voor dit model moet dus als eerste gedefinieerd worden een:

- Kennisrepresentatievorm
- Speelgoeddomein

Veel kritieken op huidige Connectionist Systems zoals de Harmonie Theorie is het feit dat de verbinding van neuron a naar neuron b dezelfde is als van b naar a: de verbindingen zijn unidirectioneel. Dit gebrek moet ook worden verholpen, dus:

- Verbindingen bi-directioneel

Het semantisch model zal als fundament gaan dienen voor het redeneersysteem. Het totaal krijgt de naam "Semantisch COnnectionistisch REdeneersysteem", afgekort tot SCORE. Uiteindelijk zal een sequentiële implementatie van SCORE onder de naam SCORE-S (SCORE-simulatie) worden losgelaten op een klein probleemdomein. Daarvoor wordt het 'theoretical diseases of the sarcophagus'-probleem gebruikt, dat ook werd gebruikt om het systeem van Gallant, besproken in paragraaf 2.5, te toetsen.

4 Het Speelgoeddomein

Indien waarnemingen consistent, volledig en zeker zijn is het ontwerpen van een redeneersysteem, dat in overeenstemming met zijn kennisbank de juiste conclusies moet trekken, meestal niet erg ingewikkeld. De traditionele expertsystemen kunnen dat prima. Lastig wordt het wanneer de waarnemingen inconsistent, onvolledig of onzeker zijn. Het afhandelen van, wat in het vervolg genoemd zal worden, 'zwakke waarnemingen' dient als uitgangspunt voor de definitie van het speelgoeddomein.

In traditionele expertsystemen worden allerlei ingewikkelde algoritmes toegepast om het inferentieproces bij zwakke waarnemingen niet te laten vastlopen. Onvolledige, inconsistente en onzekere waarnemingen vragen extra power van het redeneersysteem: het type waarneming is hem niet om het even. In SCORE is dat wel zo. Het verwerken van zwakke waarnemingen ligt intrinsiek opgesloten in de principes van het model. Het probleem van zwakke waarnemingen wordt hierin aangepakt door het netwerk te laten convergeren naar een volledige, consistente en zekere waarneming. Als die situatie is bereikt, is de inferentie in principe weer net zo eenvoudig als bij een traditioneel expertsysteem. Het herstellen van de waarneming en de daaruit afgeleide inferentie geschieden parallel. De (herstelde) waarneming beïnvloedt de inferentie, maar ook omgekeerd: de inferentie beïnvloedt de waarneming. Dit kan abstract worden toegelicht:

In SCORE, zoals we later zullen zien, zijn de eigenschappen actieve elementen. Ze proberen ieder voor zich hun positie te zoeken binnen de afleiding zelf. Stel dat we na een redenatie de afleiding van figuur 4.a hebben.

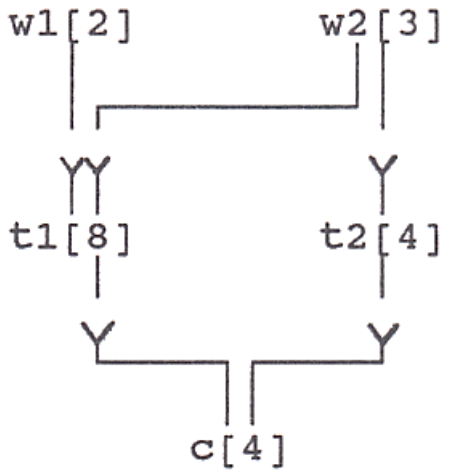


Fig. 4.a Een willekeurige afleidingsboom

Hierin stellen w1, w2, t1, t2 en c eigenschappen binnen een bepaald probleemdomein voor. w1[2] is de instantie 2 van eigenschap w1. Concreet kan dit bijvoorbeeld Ziekte[Geelzucht] zijn. In een traditioneel redeneersysteem zal een waarneming w1[2], w2[3] slaafs leiden tot tussenresultaten t1[8] en t2[4] om uiteindelijk c[4] te concluderen. Ook in het systeem van Gallant vindt dat nog steeds op die manier plaats. In SCORE echter, strijdt iedere eigenschap voor zijn aandeel in de oplossing. Iedere eigenschap zoekt, afhankelijk van de informatie die het van zijn omgeving krijgt, z'n beste instantie. (Voor t1 bleek dit uiteindelijk 8 te zijn en bijvoorbeeld geen 6.) Iedere keuze heeft weer invloed op de overige eigenschappen die daar op hun beurt ook weer een zo goed mogelijke instantie bij zoeken. Dit gaat zo door tot alle eigenschappen het ééns zijn. De afleiding geeft zich op deze manier niet willoos over aan de waarneming, maar zoekt actief mee naar eigenschap-instantie koppels die uiteindelijk in principe zelfs de waarneming zélf kunnen corrigeren. Door dit actieve element van het inferentieproces worden inconsistente waarnemingen gecorrigeerd en onvolledige waarnemingen gecompleteerd.

Het volledig en consistent maken van een waarneming is iets wat ook binnen patroonherkenningsproblemen aangetroffen wordt. In het hiernavolgende wordt dat toegelicht.

Onvolledige waarneming

Stel dat het volgende wordt waargenomen.

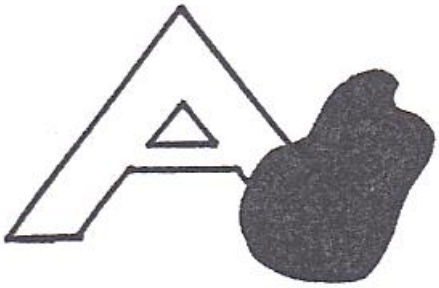


Fig. 4.b De letter 'A' met inktvlek

Wij mensen zijn prima in staat deze letter te reconstrueren. In gedachte zien wij:

Fig. 4.c De letter 'A' met inktvlek in gedachten gecompleteerd

Onzekere waarneming

Als het bijvoorbeeld te donker is, kan het zo zijn dat we niet met zekerheid in staat zijn te zeggen hoe bepaalde lijnfragmenten lopen. Een onzekere waarneming slaat dus op de (geschatte) onzekerheid in de stap van werkelijkheid naar zintuigen.

Inconsistente waarneming

Een inconsistente waarneming zou bijvoorbeeld een schrijffout kunnen zijn:

één, twee, die , vier, …

In deze zin mist het woordje 'drie' de letter 'r'. Toch hebben mensen geen enkele moeite met het herkennen van het woord. Door de context kan de inconsistentie worden hersteld.

Dit probleem van patroonherkenning is impliciet hetzelfde probleem als wat we in expertsystemen ook hebben na een vraag van de gebruiker. De vraag van een gebruiker kan gezien worden als een onvolledige waarneming (die daarnaast nog onzeker of inconsistent kan zijn) die gecompleteerd (of hersteld) dient te worden om uiteindelijk te kunnen leiden tot een conclusie. Het speelgoeddomein binnen SCORE kan worden opgevat als een speelgoeddomein van een patroonherkenningssysteem. Dit is als volgt opgevat.

Stel we hebben een ziekte Z die geïdentificeerd wordt door twee symptomen A en B. A en B kunnen zowel aan- als afwezig zijn. Als functie van deze combinaties zijn er drie ziektes bekend:

A	B	Z	
0	0	0	= geen ziekte
1	0	1	= Ziekte_1
1	1	2	= Ziekte_2

Symptoom X = 0 : Symptoom X is afwezig.
Symptoom X = 1 : Symptoom X is aanwezig.
Xϵ{A,B}

In een grafische weergave ziet dit er uit als in fig. 4.d. Op dit moment zal nog niet duidelijk zijn waarom voor een dergelijke grafische weergave is gekozen, later zal dit duidelijk worden.

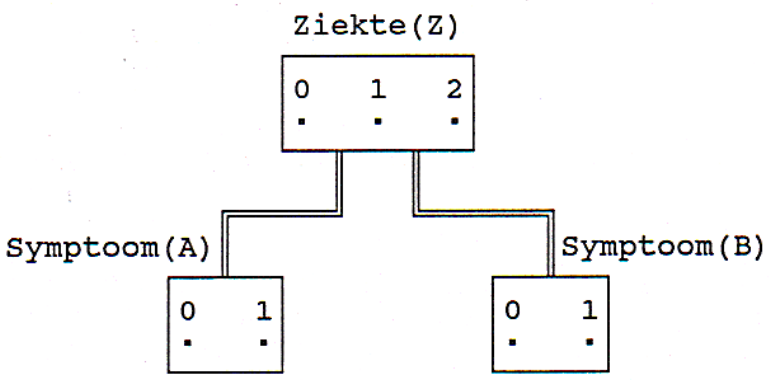


Fig. 4.d Grafische weergave eenvoudig ziektemodel

De dubbele lijnen geven de relaties aan tussen de eigenschappen. Binnen de eigenschappen is aangegeven welke instanties mogelijk zijn.

Als de waarneming volledig is (A en B worden waargenomen) en de waarneming is consistent (geen onmogelijke combinatie van A en B wordt waargenomen) en de waarneming wordt met zekerheid gedaan, dan is de redenatie erg eenvoudig. In feite is dit namelijk gewoon het uitlezen van de waarheidstabel. In een traditioneel redeneersysteem wordt die vastgelegd d.m.v. drie produktieregels:

If A[0] AND B[O] Then Z[0]
If A[1] AND B[O] Then Z[1]
If A[1] AND B[1] Then Z[2]

Echte problemen ontstaan pas bij zwakke waarnemingen.

Onvolledige waarneming

Bijvoorbeeld alleen A[0] wordt waargenomen. Geen van bovenstaande produktieregels definieert voor dit geval de waarheidswaarde van Z. De waarde van B moet immers ook bekend zijn. Toch zien wij, als we naar de waarheidstabel van Z kijken, in één oogopslag dat alleen de instantie B[0] daarbij kan horen en dat vervolgens Z[0] geconcludeerd moet worden. De combinatie A[0], B[1] bestaat immers niet in de werkelijkheid van dit kleine ziektemodel. Hierin is dus duidelijk te zien dat het patroon gecompleteerd wordt aan de hand van kennis over welke ziektes überhaupt kunnen optreden. De gecompleteerde waarneming bepaalt naar één kant de bijbehorende ziekte, maar naar de andere kant schrijven de ziektes ook voor welke combinaties van waarnemingen mogelijk zijn, d.w.z. hoe het waarnemingspatroon gecompleteerd moet worden. Attributen A en B beïnvloeden Z, maar ook attribuut Z beïnvloedt A en B.

Natuurlijk zouden we het inferentiemechanisme van het traditionele expertsysteem zo kunnen ombouwen dat Z[0] inderdaad wordt geconcludeerd als A[0] wordt waargenomen. Het probleem met die aanpak is dat voor elk type probleem waarvoor het systeem nog niet optimaal functioneert, nieuwe heuristieken aan het inferentiemechanisme moeten worden toegevoegd. Met deze aanpak mag men niet verwachten dat er een intrinsiek 'intelligent' systeem ontstaat. M.a.w. natuurlijke problemen, waarvan we er net één zagen, liggen niet intrinsiek opgesloten in het probleemoplossend vermogen van het systeem. Het gebruik van actieve elementen resulteert in een natuurlijke afhandeling van die natuurlijke problemen.

Inconsistente waarneming

Stel we nemen waar: A[0], B[1]. Deze combinatie bestaat niet binnen de werkelijkheid van ons ziektemodel. Nu zijn er twee mogelijkheden:

1. De waarneming is correct, maar deze ziekte is nog niet gecodeerd binnen ons ziektemodel. De oplossing van deze inconsistente waarneming ligt dan in het toevoegen van een vierde ziekte.

2. De waarneming is niet correct en moet dus worden gecorrigeerd. Correctie kan plaatsvinden als we weten welke combinatie: A[0], B[0] of A[1], B[1] of eventueel zelfs A[1],B[0] het meest voor de hand ligt. Of A óf B óf beiden moeten worden gecorrigeerd. Voor welke gekozen moet worden kan afhangen van de context (het model is natuurlijk meestal groter dan 3 mogelijke afleidingen of symbolen) of kan afhangen van welke combinatie het meest bekend is voor het systeem. Als bijvoorbeeld A[0], B[O] (dus Z[0]) veel vaker wordt waargenomen dan A[1], B[1] (dus Z[2]) dan is het aannemelijk te veronderstellen dat de waarneming naar A[0], B[0] gecorrigeerd moet worden. Kennis over de bekendheid van de ziektes Z kunnen een inconsistente waarneming over A en B herstellen om vervolgens een

consistente conclusie over Z te kunnen trekken. Ook hier heeft Z informatie nodig van A en B én hebben A en B informatie nodig van Z.

Onzekere waarneming

Een onzekere waarneming manifesteert zich concreet meestal als een onvolledige en/of inconsistente waarneming. Daar is in het voorgaande over gesproken. Toch kan het zo zijn dat de waarneming de vorm heeft van: “B[1] en waarschijnlijk ook A[0], maar het kan ook A[1] zijn”. A[0] wordt dan met wat minder overtuiging gesteld dan B[1]. Deze onzekerheid in de inconsistentie van de combinatie A[0], B[1] kan nu oplossen naar de combinatie A[1], B[1], i.p.v. A[0], B[0] zoals in het voorgaande voorbeeld van een inconsistente waarneming.

Het model, dat in dit rapport uiteengezet wordt, is gebaseerd op het parallel vergelijken van afleidingen, die de symbolen uit het Connectionist AI-Paradigma voorstellen, om daar uiteindelijk de beste uit te zoeken. In het ziektemodel hebben we dan drie symbolen, die voor de duidelijkheid tussen haakjes worden genoteerd:

1. (A[0], B[0], Z[0])
2. (A[1], B[0], Z[1])
3. (A[1], B[1], Z[2])

Dit geschiedt zonder dat er één of ander algoritme boven hangt dat alle mogelijke combinaties afgaat. Dat is wezenlijk voor een Connectionist Redeneersysteem. Op het niveau van de attributen wordt dié instantie geselecteerd die het beste in de afleiding past. Alle attributen zijn daar parallel mee bezig en zien slechts de informatie die van de naburige attributen binnenkomt. Bij het ziektemodel ziet Z, A en B en omgekeerd, maar A en B zien elkaar onderling weer niet. In hoofdstuk 5 wordt de methode besproken waarbij globaal naar de beste oplossing wordt gestreefd, terwijl de attributen slechts lokaal zicht hebben.

5 Technische Realisatie

5.1 Netwerkstructuur

5.1.1 Functioneel Niveau

Uitgaande van het waarnemingsniveau moeten, eventueel via subconclusie-niveaus, uiteindelijk op een conclusieniveau conclusies getrokken kunnen worden. Binnen een redeneersysteem willen we de waarnemingen, subconclusies en conclusies expliciet voor handen hebben. Binnen SCORE is gekozen voor een hiërarchische kennisstructuur. Dit idee sluit goed aan bij de probleembenadering zoals die is besproken in hoofdstuk 4. Denk bijvoorbeeld aan het ziektemodel.

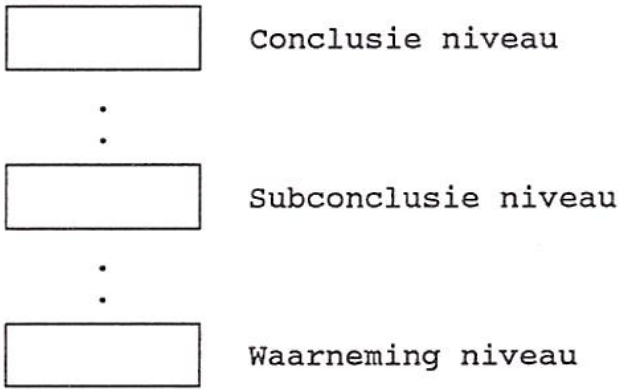


Fig. 5.1.a Hiërarchische kennisordening

Beschouw figuur 5.1.a. Alle niveaus worden expliciet geprogrammeerd. Dus op het subsymbolisch niveau zijn waarnemingen, subconclusies en conclusies expliciet aanwezig. Binnen SCORE worden deze beschouwd als de subsymbolen. De symbolen zijn, zoals in hoofdstuk 4 gedefinieerd, geoorloofde combinaties van subsymbolen: de 'afleidingen'. Deze symbolen zijn niet expliciet aanwezig maar worden als zodanig geïnterpreteerd of herkend. In deze subparagraaf wordt het substraat; het subsymbolisch niveau van SCORE, gedefinieerd teneinde aan de besproken symbolische eisen te kunnen voldoen.

Het doel van een redeneersysteem is het verstrekken van informatie aan de gebruiker. Voordat een redeneersysteem ontwikkeld kan worden moet eerst het begrip 'informatie' duidelijk zijn. Het begrip 'informatie' moet niet worden verward met het begrip 'gegevens' (of 'data'). Gegevens zijn primitieven (atomair). Ze kunnen bijvoorbeeld bestaan uit: gevoel, smaak, klanken en beelden. Het hebben van gegevens impliceert niet automatisch dat er ook informatie is. De gegevens 'boom', 'stam', 'takken', 'wortels' als losse objecten geven ons, op zich zelf, geen informatie: de namen bevatten geen enkele kwaliteit of kenmerk van het reële

object. Informatie ontstaat pas als de samenhang tussen de gegevens wordt aangegeven, d.w.z. wanneer er betekenis (semantiek) aan de gegevens wordt toegekend. Als we bijvoorbeeld zeggen dat een 'boom' een samenstel van 'stam', 'takken' en 'wortels' is, dan is er pas sprake van informatie: er wordt semantiek aan het gegeven 'boom' toegekend. Informatie, in tegenstelling tot gegevens, is dus een samengesteld begrip (moleculair). Onsamenhangende gegevens leiden nooit tot informatie. Voordat we tot het ontwerp van een redeneersysteem kunnen overgaan moeten de gegevens dus eerst worden gestructureerd.

In [Zeppenfeldt, 1988] wordt een idee geopperd voor het gebruik van neurale netwerken in expertsystemen. Zijn speelgoeddomein kent een vijftal objecten en relaties ertussen: Leo heeft een fiets. Kees heeft een fiets, en auto. Toevallig is Kees ook ziek.

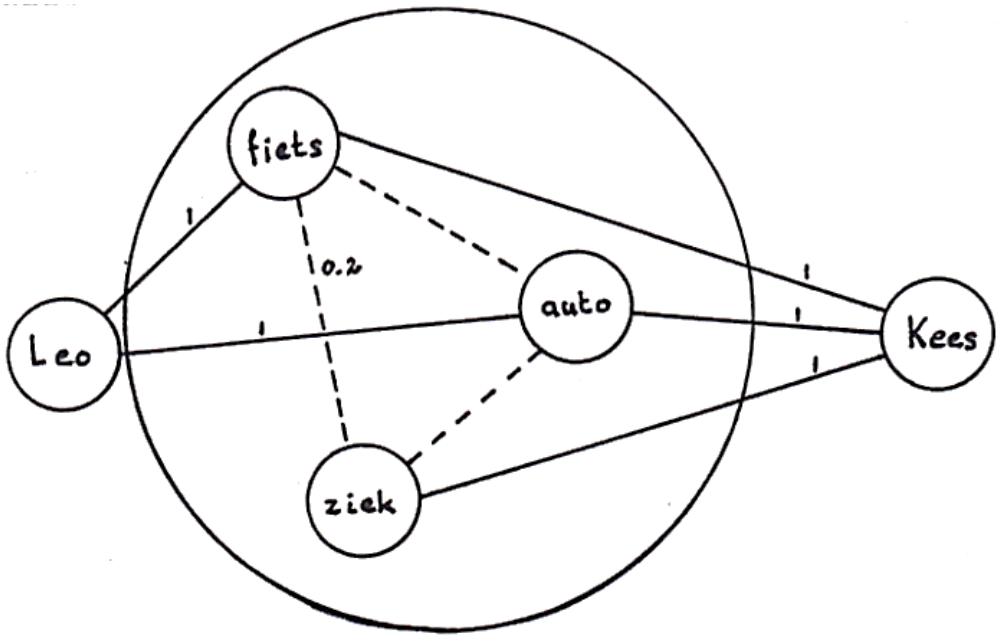


Fig. 5.1.b En klein model van een stukje werkelijkheid

Beschouw hiervoor figuur 5.1.b. De getrokken verbindingen of associaties zijn een afspiegeling van bovenstaand verhaaltje over Kees en Leo. Als bijvoorbeeld Leo wordt waargenomen, wordt ook aan zijn fiets gedacht en omgekeerd. De sterktes van deze verbindingen zijn allen maximaal (binnen zijn model is dat één). De gestippelde lijnen zijn verbindingen of associaties tussen objecten, zoals die zijn ontstaan nadat het netwerk een tijdje in een praktijkomgeving heeft gefunctioneerd. Die verbindingen zijn neveneffecten: ze waren oorspronkelijk niet van belang. De sterkte van één gestippelde verbinding is aangegeven. De overige zijn niet van belang. Op het eerste gezicht is het heel aardig als het

netwerk zelf zijn relaties gaat leggen tussen objecten, zonder dat de expert die zelf expliciet bewust was. Dit is zeker een bijzondere eigenschap van neurale netwerken: in de praktijk ingezet zal het netwerk zelf leren. Het probleem wat hier echter naar voren komt is het feit dat er via bovenstaande aanpak weliswaar een hoop conditionele relaties kunnen worden gelegd, maar daar geen absolute uitspraken aan kunnen worden ontleend. Wat zeggen de vier met 'fiets' in verbinding staande objecten met hun verbindingen nu in absolute zin over 'fiets'? Hoe wegen we hun invloeden? Er moet een zodanige basisstructuur worden opgezet waarbinnen relaties tussen de objecten vrij kunnen worden gelegd. Deze structuur moet die uitspraken dwingen zinvol te zijn. Als we een redeneersysteem opbouwen uit zinvolle uitspraken is het geheel ook zinvol.

Nu moet worden uitgezocht wat zinvolle uitspraken zijn. Een zinvolle uitspraak moet ons informatie geven en moet dus waar zijn. De beantwoording van voorgaande vraag is nu verschoven naar de vraag: wanneer geeft een uitspraak ons informatie? Hoe moet een netwerk worden opgezet opdat we kunnen komen tot uitspraken over een subject? Er moet een consistente aanpak worden gevolgd om te komen tot zinvolle uitspraken. Dan en slechts dan blijft de weg open naar een zinvol redeneersysteem. Er is geen zinvol redeneersysteem mogelijk als het netwerk zélf inconsistent is of dubbelzinnigheden bevat. Pas als het netwerk consistent en ondubbelzinnig is kan er een systeem op worden ontworpen dat inconsistentie en dubbelzinnigheid in de waarneming op de juiste manier te lijf gaat.

We beschouwen een object uit de werkelijkheid, bijvoorbeeld een boom. Als ik over straat loop en ik kijk naar die boom, het object is nu het subject van mijn aandacht geworden, dan vallen voor mij stam en takken op. De werkelijkheid van de boom bevat meer details, maar voor mij zijn die niet van belang. Onbewust laat ik details weg om zo een hanteerbare collectie over te houden. Datgene wat dan overblijft wordt een "abstractie' genoemd. Mijn abstractie van de boom is zijn stam met zijn takken. Voor een bioloog is veel meer van belang. Zijn abstractie van de werkelijkheid van de boom zal daarom dichter bij die werkelijkheid zelf liggen. De details die in een abstractie worden beschouwd, worden bepaald door het doel dat men voor ogen heeft. Als we de werkelijkheid op een bepaald abstractieniveau willen vangen, dan is het van belang dat de uitspraken die we erover plegen wáár zijn. Dergelijke uitspraken worden 'asserties' genoemd. Binnen mijn werkelijkheid van een boom is een boom niets anders dan de som van zijn stam en takken. Dat is mijn assertie over de boom.

Een assertie bestaat uit twee delen: het 'subject' (onderwerp) en het 'predicaat' (gezegde). Dit wordt op de volgende manier genoteerd: boom = stam,takken. Informatie over stam en takken levert ons nu informatie over de boom. Informatie over de stam kan bijvoorbeeld in een assertie worden weergegeven als: stam = bast,kruin. D.m.v. naamgeving ontstaat zo structuur in de beschrijving van het subject 'boom'. Informatie over 'bast', 'kruin' en 'takken' levert ons nu informatie over 'boom'.

Voor een assertie geldt, en dat is belangrijk, de 'omkeerbaarheidseis' die als volgt luidt. Enerzijds bepaalt subject éénduidig het predicaat: subject -> predicaat, anderzijds wordt het subject éénduidig door het predicaat bepaald: subject <- predicaat. Dat dit wordt geëist is in het licht van voorgaande niet verwonderlijk meer omdat, het voorbeeld van de boom er weer bijhalende, een boom de som van zijn takken is. Een zelfde stam met andere takken is een andere boom en dezelfde takken met dezelfde stam is dezelfde boom. Asserties zijn algemene uitspraken over objecten. Concrete objecten worden 'instanties' of 'instantiaties' genoemd: die ene boom daar voor het huis van de Cornelis Anthoniszstraat 66, is dus een instantie. Dat is dié boom met dié stam en dié takken. Dit begrip is straks ook van belang. Meer over voorgaande zaken, maar dan echter puur toegesneden op databases, kunt U terugvinden in [Bekke, 1988].

D.m.v. een systeem dat beschreven is in termen van asserties wordt gegarandeerd dat de beschrijving van de werkelijkheid consistent is waardoor zinvolle afleidingen mogelijk worden. Het grote verschil tussen de redeneersysteem benadering d.m.v. asserties en de redeneersysteem benadering die door Zeppenfeldt geschetst werd is het feit dat een assertie iets zegt over subject enerzijds en een aantal eigenschappen tezamen (het predicaat) anderzijds. Er worden dus niet relaties gelegd tussen subject en losstaande eigenschappen, maar het gehéél van subjectpredicaat wordt in beschouwing genomen. Dat maakt deze benadering zeer krachtig.

In SCORE wordt uitgegaan van een door de expert gedefinieerde tijdsinvariante, op asserties gebaseerde, netwerkstructuur. Vervolgens wordt het netwerk getraind om toegespitst te geraken op de praktijkomgeving waardoor oplossingen gevonden kunnen worden voor onvolledige, inconsistente en onzekere waarnemingen. Trainingen vinden plaats op het niveau van de instanties binnen de goed gedefinieerde netwerkstructuur, om uiteindelijk te resulteren in relaties tussen de instanties. Als deze relaties zijn gelegd is het redeneersysteem klaar voor praktijkgericht, zinvol redeneren. Waarnemingen vinden plaats op het niveau van gegevens; van instanties, om d.m.v. de relaties te leiden tot informatie over het specifieke object waarover de waarneming plaatsvindt. SCORE, dat zoals U weet is geïnspireerd door het Connectionist AI-Paradigma, zal trachten de gegevens zó te selecteren dat aan de asserties zo goed mogelijk wordt voldaan. De asserties zijn dan ook te beschouwen als de hard constraints! (zie paragraaf 2.3.3)

Met het oog op het uiteindelijke model worden de asserties in unitvorm weergegeven met daarbinnen de instanties van de eigenschappen waaruit de assertie bestaat. Beschouw figuur 5.1.c voor de weergave van de assertie C=A,B. De verbindingen tussen de units staan voor de relaties die er liggen: de instanties van unit C zijn afhankelijk van de instanties van de units A en B en omgekeerd. Tussen de units A en B onderling, ligt geen relatie. A en B kunnen immers onafhankelijk van elkaar hun waarde aannemen. Unit C wordt de 'aggregatie' (=samenvoeging) genoemd van de units A en B.

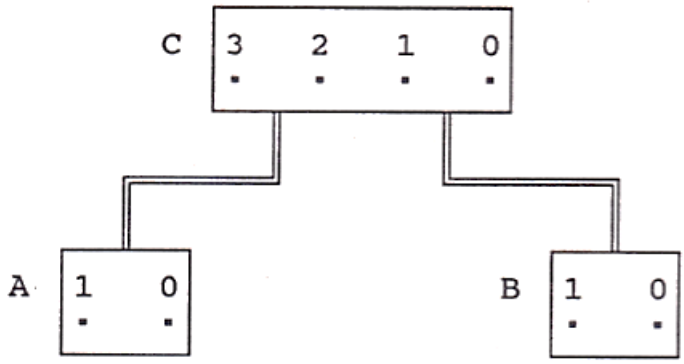


Fig. 5.1.c De grafische weergave van een assertie in netwerkvorm met bijbehorende instanties.

Hoe de relaties feitelijk liggen tussen de instanties van de eigenschappen kan in tabelvorm worden weergegeven. Als gevolg van de omkeerbaarheidseis kan C uit maximaal vier instanties bestaan omdat A en B maximaal vier verschillende combinaties kunnen aangaan. De 'waarheidstabel' van bovenstaand volledig gevuld netwerkje ziet er dan uit als in tabel 5.1.a.

A	B	C
0	0	0
0	1	1
1	0	2
1	1	3

Tabel 5.1.a De waarheidstabel behorende bij figuur 5.1.c

In het vervolg wordt de instantie i van een eigenschap E als volgt aangeven: E[i]. Er geldt dan bijvoorbeeld: C[1]=A[0],B[1]. Ook dergelijke uitspraken worden asserties genoemd. C=A,B is een assertie op eigenschapniveau, terwijl bijvoorbeeld C[2]=A[1],B[0] een assertie is op instantieniveau. In dit rapport wordt de naam 'assertie' voor beide niveaus gebruikt omdat het steeds duidelijk zal zijn of het 't eigenschap- of instantieniveau betreft.

Deze benadering heeft gevolgen voor de kennisstructurering van praktische problemen. Stel weer dat A en B symptomen zijn en C is de ziekte, die in het vervolg Z wordt genoemd. Het is

niet ondenkbaar dat er relaties als gespecificeerd in tabel 5.1.b tussen de eigenschappen liggen.

A	B	Z
0	0	0
0	1	0
1	0	1
1	1	2

Tabel 5.1.b Waarheidstabel van een ziekte Z, als functie van twee symptomen A en B.

Aan de omkeerbaarheidseis is hier niet voldaan omdat bij één ziekte, in dit geval Z[0] , meerdere symptoomcombinaties horen. Toch zijn dit soort situaties in de praktijk heel goed mogelijk. Om toch aan de omkeerbaarheidseis te blijven voldoen moeten we Z[0] opsplitsen in $Z[0_1]$ en $Z[0_2]$, die beiden voor de buitenwereld ziekte Z[0] voorstellen maar intern gescheiden worden behandeld. Beschouw hiervoor tabel 5. 1. c.

A	B	Z
0	0	0_1
0	1	0_2
1	0	1
1	1	2

Tabel 5.1.c. De waarheidstabel van tabel 5.1.b, waarbij de omkeerbaarheidseis is hersteld.

Op deze manier gaat er geen semantiek verloren. Stel bijvoorbeeld ook dat we een onvolledige waarneming hebben: A[O]. Als we geen onderscheid maken tussen $Z[0_1]$ en $Z[0_2]$ dan is het ook niet mogelijk de instantie van B te completeren. Z dient als brug voor het ondubbelzinnig kunnen maken van associaties tussen A en B.

Een andere praktijksituatie die kan optreden is weergegeven in tabel 5.1.d.

A	B	Z
0	0	0
0	1	1
1	0	2
1	1	3
1	1	4

Tabel 5.1.d. Een mogelijke praktijksituatie voor de ziekte Z als functie van de symptomen A en B.

In dit voorbeeld wordt de omkeerbaarheidseis naar de andere kant overtreden. Dit wordt ook niet toegestaan. In dit soort situaties schiet de kennisstructuur dus blijkbaar te kort. We hebben nog een unit H (later zal dit deze de naam 'hidden unit' krijgen) nodig om de twee varianten van de waarneming A[1], B[1] van elkaar te onderscheiden. Een mogelijkheid is weergegeven in tabel 5.1.e.

H	A	B	Z
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	1	1	4

Tabel 5.1.e Oplossing van de omkeerbaarheidsfout van tabel 5.1.d.

Nu zijn niet alle mogelijke combinaties van H, A en B gedefinieerd, maar dat hoeft ook niet. SCORE zal convergeren naar één van deze vijf, wat we in termen van het Connectionist AI-Paradigma genoemd hebben, symbolen. Dat is precies wat we willen.

Beschouw tabel 5.1.f.

A	B	C
0	0	0
0	1	1
1	0	2
1	1	3

Tabel 5.1.f. Een algemene waarheidstabel, waarbij alle mogelijke combinaties van de eigenschappen A en B zijn beschreven in C.

In deze waarheidstabel stellen A, B en C eigenschappen voor. Eigenschappen A en B kunnen enkel aan- of afwezig zijn, gerepresenteerd door een 1 respectievelijk een 0. Alle mogelijke (bepaald door de omkeerbaarheidseis die aan de asserties wordt opgelegd) instantiecombinaties over A en B zijn weergegeven. C bevat dus vier instanties. De vier asserties die dan ontstaan stellen afleidingen voor: als bijvoorbeeld A[0], B[1] wordt waargenomen dan moet dit resulteren in de conclusie C[1]. In hoofdstuk 4 is al opgemerkt dat in SCORE deze vier asserties niet *expliciet* worden geprogrammeerd. De redenen daarvan zijn uitgebreid besproken in paragraaf 2.3. In SCORE worden deze asserties *impliciet* geprogrammeerd. Programmering vindt plaats in termen van soft constraints. Op het symbolisch niveau verschijnen toch de asserties maar nu echter in een meer flexibele vorm, waardoor inconsistente of onvolledige waarnemingen op een natuurlijke, d.w.z. 'ongetructe' , manier kunnen worden afgehandeld. Het netwerk zal altijd convergeren naar één van zijn legitieme asserties of legitieme afleidingen.

Zoals hiervoor opgemerkt, is er pas sprake van informatie als er relaties worden aangegeven tussen de eigenschappen. Deze relaties bepalen de semantiek van het systeem. Zonder relaties hebben de eigenschappen geen betekenis. Op symbolisch niveau wordt de semantiek gedefinieerd in termen van asserties, maar als we deze niet expliciet programmeren, hoé komt het dat er in SCORE tóch die hard constraints aanwezig zijn? Om dit te begrijpen moeten we het subsymbolisch niveau verder uitdiepen.

Het subsymbolisch niveau is in twee aparte niveaus onderverdeeld: het functioneel- en het implementatie niveau. Het functioneel niveau is het niveau van de eigenschappen, instanties en relaties tussen de eigenschappen en het implementatie niveau is het niveau van de units of processoren, nodes en verbindingen tussen de nodes:

Subsymbolisch niveau	
Funktioneel Niveau	Implementatie Niveau
Eigenschap	Unit/Processor
Instantie	Node
Relatie	Verbindingen

In deze subparagraaf is uitgebreid op het functioneel niveau ingegaan. Daarbinnen werd ook de kennisstructuur opgezet. In de volgende subparagraaf wordt verder afgedaald: naar het implementatieniveau.

5.1.2 Implementatie Niveau

Een node staat voor een instantie van een eigenschap. De naam van een node is gelijk aan de naam van de instantie. Een naam van een node kan bijvoorbeeld C[2] zijn. In het voorbeeld van de vorige subparagraaf (figuur 5.1.c) hebben we dus 8 nodes. De nodes zijn logisch gegroepeerd binnen een unit. De naam van een unit is dezelfde als de naam van bijbehorende eigenschap. In datzelfde voorbeeld zijn er drie units, te weten A, B en C. De relatie tussen twee eigenschappen A en C wordt weergegeven door twee stelsels van verbindingen. Een stelsel voor de verbindingen van unit A naar unit C en een stelsel voor de verbindingen in de omgekeerde richting: vanuit nodes van unit A lopen verbindingen naar nodes van unit C en omgekeerd. Er kunnen alleen verbindingen worden gelegd tussen nodes van units waar ook een relaties tussen is. Er liggen dus geen verbindingen tussen de nodes van unit A en de nodes van unit B.

Binnen deze subparagraaf wordt de unit, node en verbinding formeel behandeld. De begrippen en notaties worden geïntroduceerd die voor het vervolg van dit technische hoofdstuk van belang zijn.

Unit

Een unit is een logisch samenhangend geheel van nodes. Een unit is geen passief, maar actief element. Deze activiteit wordt ondersteund d.m.v. de term 'unitbesturing'. Informatie afkomstig van andere units gaat eerst naar deze unitbesturing.

In het voorgaande is gesteld dat SCORE en hiërarchische kennisbenadering heeft. Iedere unit kan verbonden zijn met units die hoger- en units die lager in de kennishiërarchie liggen. De

eerste type units worden 'parentunits', de tweede 'childunits', genoemd. Als er m Parentunits zijn, worden die genummerd van P1 t/m Pm. De n Childunits krijgen op dezelfde manier de namen C1 t/m Cn. Verder kan op iedere unit een waarneming plaatsvinden. Voor een waarneming, op een unit U, wordt een extra unit met de naam U' (naam oorspronkelijke unit met een toegevoegd accent) geïntroduceerd. Zo'n type unit noemen we in het vervolg een 'Perceptorunit'. In figuur 5.1.d zijn de relaties van U met perceptor-, parent- en childunits weergegeven d.m.v. dubbele lijnen.

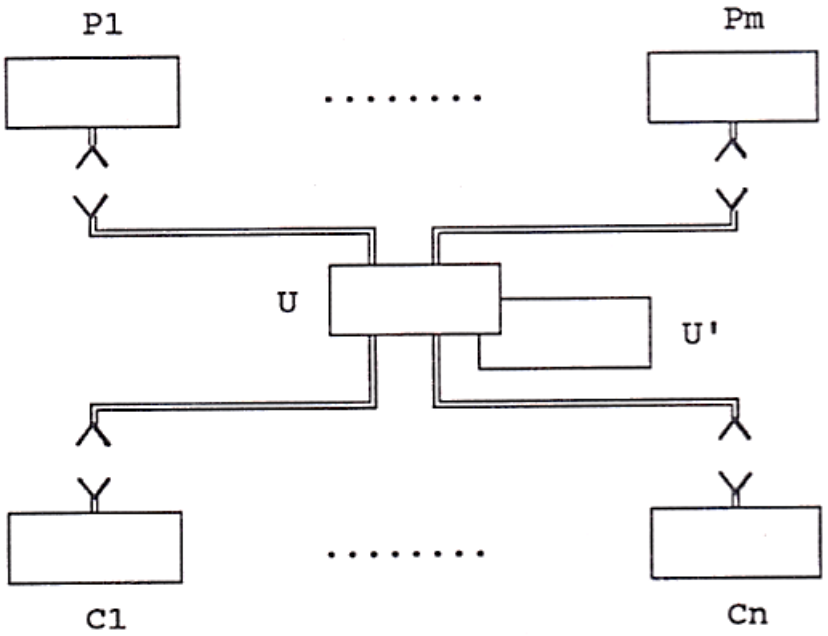


Fig. 5.1.d De relaties van perceptor- (U'), child- (C1 t/m Cn) en Parentunits (P1 t/m Pm) met unit U.

De unit is het actieve element van het netwerk en dient op grond van de informatie afkomstig van andere units te beslissen welke onderliggende node geactiveerd moet worden. De unit moet daartoe weten hoe de adviezen van child-, parent en perceptorunit(s) te interpreteren. De functies van deze drie zijn namelijk verschillend. Hier wordt later op terug gekomen. De adviezen van parent-, child- en perceptorunits worden aangeduid met respectievelijk de namen; $r_{1,\alpha}$, $r_{2,\alpha}$, $r_{3,\alpha}$. $r_{1,\alpha}$ is de resultante van de parentunits-invloeden op node α. $r_{2,\alpha}$ staat voor de resultante van de childunits en $r_{3,\alpha}$ voor de resultante van de perceptor. Iedere unitbesturing heeft als doel het zodanig activeren van nodes dat zo goed mogelijk rekening wordt gehouden met deze resultanten opdat uiteindelijk een situatie zal worden bereikt dat alle adviezen dezelfde zijn geworden: alle units zijn het eens.

Een ander aspect van de unitbesturing is het garanderen dat het in zijn lokale streven naar een globaal zo goed mogelijke oplossing, niet in een lokaal optimum blijft steken. In

subparagraaf 2.3.4 zijn de begrippen lokale en globale optima reeds genoemd. Binnen SCORE is als oplossing voor dit probleem voor een methode met de naam 'simulated-annealing' gekozen. In subparagraaf 2.4.4 is deze methode beschreven, zoals het wordt toegepast binnen de Harmony Theorie. In subparagraaf 5.3.2.2 wordt deze methode besproken zoals deze binnen SCORE is toegepast.

Omdat de unit alle berekeningen nog kan doen op grond van de lokaal binnenkomende informatie, is de unit het laagste niveau waaraan we een processor, als we het hebben over een parallelle implementatie van SCORE, kunnen toebedelen. Een processor kan dan natuurlijk ook worden toebedeeld aan clusters van units of aan het complete netwerk. Van het laatste geval is gebruik gemaakt binnen SCORE-S. In het verdelen van het netwerk in lokale rekeneenheden kunnen we niet verder gaan dan een verdeling in units. Instanties van eigenschappen, dus de nodes, voeren geen lokale berekeningen meer uit zoals de units dat nog wel doen. De unit heeft het overzicht over al zijn nodes en de unit beslist dan ook welke node actief moet worden. Bij de node is geen lokaal beslissingsalgoritme aanwezig. Aan de nodes kunnen dus geen processoren worden toegekend.

Node

Een node staat voor een instantie van een eigenschap. Als die instantie U[α] is, dan is de naam van bijbehorende node ook U[α]. Als het duidelijk is om welke eigenschap het gaat, dan kunnen we volstaan met de specificatie van de instantienaam: α.

Een node heeft geen werkelijk fysieke betekenis. Als metafoor wordt het apart van de unit gezien, maar in werkelijkheid is het niet meer dan bijvoorbeeld een register van de bovenliggende unitprocessor. Nodes kunnen slechts discrete waarden aannemen en wel zodanig dat één node actief is (de waarde één heeft) en de overige passief (d.w.z. de waarde nul). Dit geldt niet voor de nodes van de perceptor die wel continue waarden kunnen aannemen. Dit laatste heeft te maken met de eis dat waarnemingen onzeker mogen zijn.

Tijdens het redenatieproces waar de nodes (of eigenlijk dus de units, maar we gebruiken hier de metafoor) naar hun juiste waarde zoeken kan het op dat moment actief zijn van een bepaalde node worden geïnterpreteerd als het 'voorlopig geloven' in het wáár zijn van de bijbehorende instantie. Dit geloof heeft gevolgen voor de andere units, die op grond daarvan ook weer wat gaan geloven. Teruggekoppeld naar de eerste unit kan dit weer leiden tot het feit dat toch maar weer een andere node wordt geloofd. Als uiteindelijk alle units bij hun geloof blijven en dus niet meer 'voorlopig geloven', maar 'definitief geloven' hebben we het antwoord van het probleem te pakken.

De output van een node U[α] wordt aangegeven d.m.v. o(U[α]). Er geldt dus:

$$O(U[\alpha]) = \begin{cases} 1, \text{ voor één node } \alpha \\ 0, \text{ voor de overige nodes} \end{cases}$$

De input aan de unit voor een specifieke node U[α], afkomstig van een specifieke unit V € {P1, … , Pm, C1, … , Cn, U'} wordt in het vervolg i(V, U[α]) , of kortweg i(V, α), genoemd. i(V, α) is dus de complete invloed van unit V op node U[α]. De waarde van i(V, α) is afhankelijk van de verbindingen tussen de nodes (of eigenlijk de units). Onder het volgende punt 'Verbinding' wordt i(V, α) nauwkeuriger gedefinieerd.

Verbinding

Verbindingen tussen nodes worden weergegeven d.m.v. verbindingen tussen de units. De dubbele lijnen uit de voorgaande figuren stellen stelsels van verbindingen voor. We beschouwen in figuur 5.1.e de verbindingen van een unit U met een unit V.

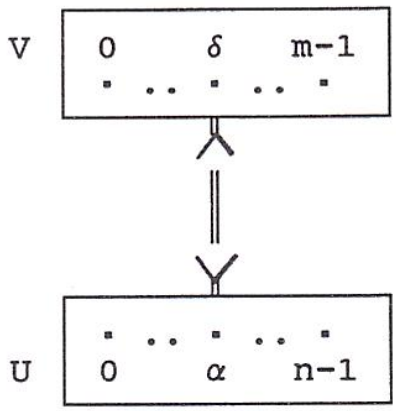


Fig. 5.1.e Units U en V met hun nodes en onderlinge verbindingen

Vanuit alle nodes van unit U liggen verbindingen met alle nodes van unit V, en omgekeerd. De sterkte of gewicht ('Weight') van een verbinding van een node V[δ] naar een node U[α] wordt aangegeven d.m.v. de variabele W(V[δ],U[α]). Als bekend is welke unitrelatie het betreft kunnen we volstaan met de notatie: W(δ,α). De verbindingen van node V[δ] naar de nodes van unit U zijn aangegeven in figuur 5.1.f.

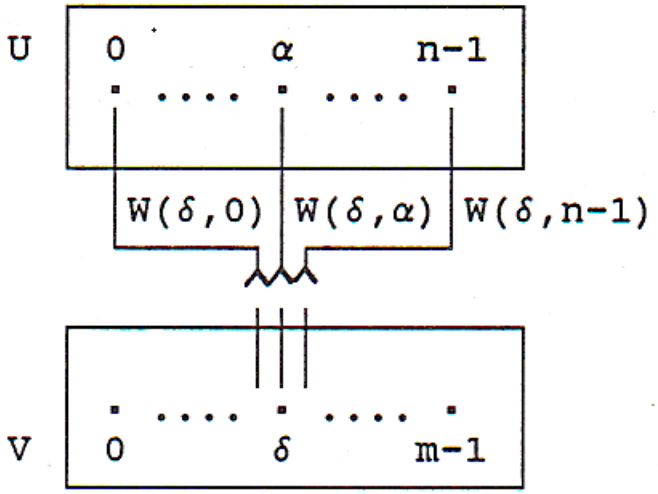


Fig. 5.1.f Verbinden van V[δ] naar U[α], α € {0, … , n-1}

En bij U[α] komen de verbindingen aan als in figuur 5.1.g.

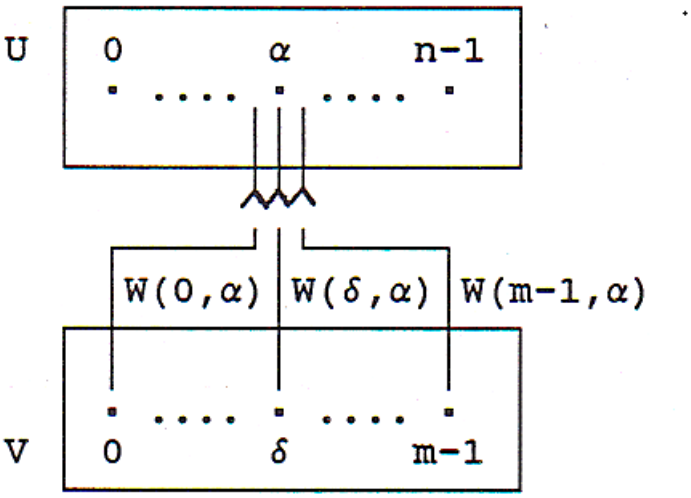


Fig. 5.1.g Verbindingen van V[δ], δ € {0, … , m-1} naar U[α]

De verbindingen tussen een perceptorunit U' en de unit U zelf liggen in slechts één richting, namelijk van perceptorunit U' naar de unit U zelf. Alleen overeenkomstige nodes zijn met elkaar verbonden. De verbindingen tussen perceptorunit U' en de unit U zijn allen één. In subparagaraaf 5.2.1 alwaar de semantiek van de verbindingen uit de doeken wordt gedaan, zal duidelijk worden waarom dit zo is.

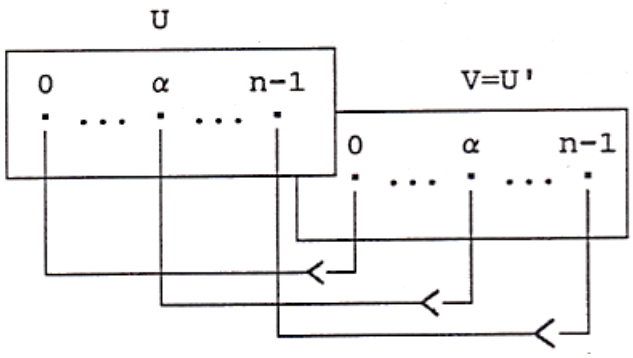


Fig. 5.1.h De verbindingen tussen perceptorunit U' en unit U

i(V,α) met V € {P1, … ,Pm, C1, … ,Cn, U' } wordt nu als volgt gedefinieerd:

$$i(V,\alpha) = \sum_{\delta} o(V[\delta]) * W(V[\delta],U[\alpha])$$

M.a.w. de input voor unit U, van unit V, wordt bepaald door de actieve nodes uit V en zijn verbindingen met U. Aangezien er maar één node uit V actief zal zijn zal er op ieder moment ook maar één verbinding van unit V naar node U[α] functie hebben.

Aangezien voor de verbindingen van unit V=U' naar unit U geldt dat W(U'[τ],U[τ])=l en W(U'[α],U[τ])=l voor de overige nodes α<>τ, kunnen we deze formule voor de perceptor reduceren tot: i(U',α)=o(U'[α]). De perceptor zal echter niet als een speciaal geval behandeld gaan worden. Vandaar dat in het vervolg ook voor de perceptor i(V,α) zal worden gebruikt en niet o(U'[α]).

5.2 Netwerk Initialisatie

5.2.1 Semantiek van de verbindingen tussen de nodes

Voordat het netwerk, zoals in paragraaf 5.1 beschreven, als redeneersysteem gebruikt kan worden moet er eerst betekenis aan de verbindingen worden toegekend. We hebben al gezien dat nodes staan voor instanties van eigenschappen en verbindingen voor relaties hiertussen. Op deze kwalitatieve aspecten is uitgebreid ingegaan. Wat ons nu interesseert is de kwantitatieve betekenis van verbindingen en nodes.

Beschouw nogmaals figuur 5.1.d. Hierin zijn alle mogelijk units V getekend V € {P1, ... , Pm, C1, . . Cn, U'} die met U in verbinding staan. Iedere unit V heeft suggesties voor unit U. In figuur 5.2.a is er één unit V uitgelicht.

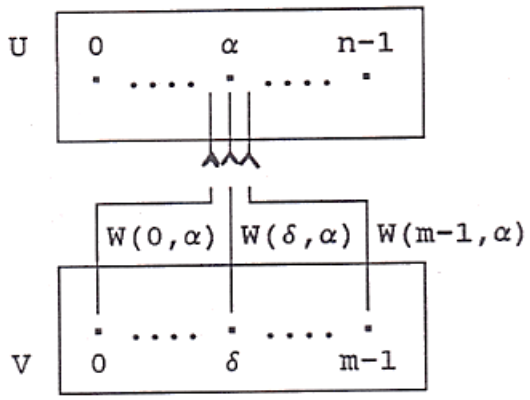


Fig. 5.2.a Invloed van unit V op unit U

De verbindingen zijn rechtstreeks van node tot node getekend. Nogmaals: de besturing van unit U zit er nog tussen. De invloed die unit V op unit U uitoefent is herhaald in formule 5.2.a.

$$i(V,\alpha) = \sum_{\delta} o(V[\delta]) * W(V[\delta],U[\alpha])$$

Formule 5.2.a

Met deze formule in het achterhoofd wordt nu eerst de Stelling van de Totale Waarschijnlijkheid geïntroduceerd.

De Stelling van de Totale Waarschijnlijkheid, vertaald naar de terminologie zoals in dit rapport gebruikt, luidt als volgt:

> Definieer Ω als de verzameling van alle relevante instanties van een bepaalde eigenschap. Als nu geldt dat alle instanties van V disjunct zijn, d.w.z. $V[i] \cap V[j] = \emptyset$ voor i<>j én gezamenlijk Ω opspannen, d.w.z. $U_{\delta} V[\delta] = \Omega$, dan geldt voor instantie U[α] de volgende :
>
> $$P(U[\alpha]) = \sum_{\delta} P(U[\alpha] \mid V[\delta]) * P(V[\delta])$$
>
> **Formule 5.2.b**
>
> Het bewijs van deze stelling kunt U vinden in ieder kansrekeningboek, bijvoorbeeld [Grimmett, 1982].

In woorden betekent dit dat als een eigenschap U volledig wordt bepaald door een eigenschap V, de kans op een instantie van U volgens formule 5.2.b kan worden bepaald als de instanties van V elkaar niet overlappen.

Als we deze formule voor de kans op U[α] vergelijken met de formule voor i(V, α) dan valt de analogie onmiddellijk op. Naar aanleiding van deze constatering is besloten in SCORE de kansrekening zo goed als mogelijk toe te passen: alle waarden van verbindingen en activiteiten van nodes zijn te interpreteren als een kans. Het bleek echter niet mogelijk het redeneermechanisme volledig te baseren op de analogie tussen de formules 5.2.a en 5.2.b, waardoor enige concessie t.a.v. de wiskunde is gedaan en waar vervolgens een meer intuïtieve strategie tegenover is gezet. Er zal later een voorbeeld worden besproken waarin duidelijk is te zien hoe dat dan precies mis gaat. De analogie vormt wel een belangrijke basis voor het redeneren binnen SCORE. In het hiernavolgende wordt daarom eerst ingegaan op deze analogie om daarna te kunnen komen tot een beschrijving hoe deze is ingepast in het uiteindelijke redeneerproces.

Verschillende units V beïnvloeden de unit U en geven daarmee hun invloed i(V, α) aan unit U af. i(V, α) kan nu worden geïnterpreteerd als een kansadvies van unit V aan unit U. Dit kansadvies is afkomstig van een unit V, maar kan afkomstig van een andere unit weer anders zijn. Er kan daarom niet worden gesteld dat i(V, α) = P(U[α]). P(U[α]) laten we daarom overgaan in een $P_V(U[\alpha])$, d.i. de kans op U[α] voor wat betreft unit V. Formule 5.2.b. gaat daarom over in formule 5.2.c, als aan de daarna vermelde drie voorwaarden is voldaan.

$$P_V(U[\alpha]) = \sum_\delta P(U[\alpha] \mid V[\delta]) * P(V[\delta])$$

Formule 5.2.c

<u>Voorwaarde 1</u>

o(V[δ]) = P(V[δ])

We hebben gezien dat per unit steeds één node actief kan zijn, d.w.z.: o(V[δ])=1 voor één node V[δ] en o(V[τ])=0 voor alle nodes τ<> δ. Dit houdt dus in dat voor die ene actieve node geldt: P(V[δ])=1 en voor de overige passieve nodes P(V[τ])=0 voor τ<>δ. In de Stelling van de Totale Waarschijnlijkheid zagen we als eis dat $U_\delta V[\delta]=\Omega$. Dit houdt niets meer in dan $\sum_\delta$ P(V[δ])= 1. Ω is immers de totale kansruimte. Als we willen dat o(V[δ])=P(V[δ]), dan moet formule 5.2.d gelden.

$$\sum_{\delta=0}^{m-1} o(V[\delta]) = 1$$

Formule 5.2.d

Met één actieve node δ (o(V[δ])=l) en passieve nodes τ <> δ

(o(V[τ])=0) geldt dat inderdaad. Conclusie: binnen SCORE kunnen we o(V[δ]) beschouwen als P(V[δ]). De betekenis van de outputwaarde van een node is dus eenvoudigweg de kans op bijbehorende instantie.

Voorwaarde 2

W(V[δ],U[α]) = P(U[α]|V[δ])

Via deze relatie wordt er kwantitatieve betekenis gegeven aan de verbinding van node V[δ] naar U[α]: die verbinding is de conditionele kans op U[α], gegeven dat V[δ] wáár is. Dit sluit goed aan bij het idee dat Smolensky heeft van verbindingen in een Connectionist System, namelijk het idee van de 'Statistical Connection' (zie subparagraaf 2.2.3). Deze conditionele kansen worden beschouwd als de absolute waarheden waaraan de units zich tijdens het redenatieproces moeten conformeren. De verbindingen, samen met de waarnemingen die ook absoluut zijn, schrijven het systeem impliciet de eindoplossing voor. De verbindingen hebben tijdens het redenatieproces dan ook een constante waarde.

Een volgende eis die nu ontstaat is weergegeven in formule 5.2.e.

$$\sum_{\alpha} P(U[\alpha] \mid V[\delta]) = 1$$

Formule 5.2.e

Voor het behoud van de semantiek in het netwerk wordt immers geëist (omkeerbaarheidseis, zie subparagaraaf 5.1.1) dat eigenschap V noodzakelijk is voor de beschrijving van de instanties van eigenschap U. Alle implicaties van V[δ] t.a.v. eigenschap U moeten bestaande instanties van U zijn omdat er domweg niet meer bestáát binnen de werkelijkheid van het netwerk. $\sum_{\alpha} P(U[\alpha] \mid V[\delta])$ is dan ook 1. Daar waar geen semantische relatie ligt zijn deze conditionele kansen nul.

Voorwaarde 3

Als voorgaande geldt, dan geldt dus ook: $i(V, \alpha) = P_V(U[\alpha])$. Van verschillende kanten krijgt unit U adviezen omtrent de kansverdeling over zijn nodes. Het kansverdelingsadvies dat unit U van unit V krijgt, was i(V,α). Iedere unit V€{P1, … Pm, C1, … Cn, U'} brengt dat kansverdelingsadvies uit aan unit U. De besturing van unit U combineert alle parentadviezen in $r_{1,\alpha}$, alle childadviezen in $r_{2,\alpha}$ en het perceptoradvies in $r_{3,\alpha}$, op een manier die zodadelijk wordt besproken. via dat algoritme wordt aan de hand van $r_{1,\alpha,}$ $r_{2,\alpha}$ en $r_{3,\alpha}$ door de unitbesturing de nieuwe actieve node berekend.

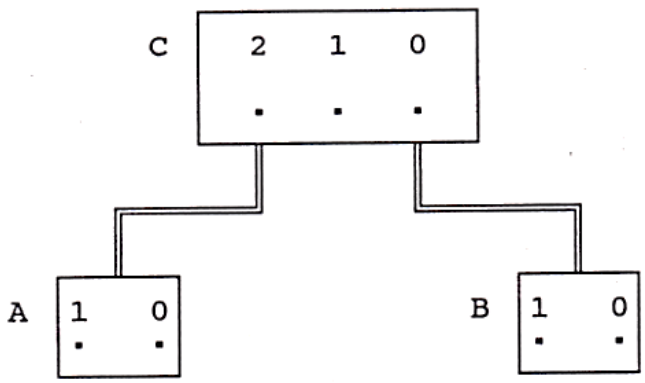
Er zijn dus blijkbaar meerdere $r_{i,\alpha}$'s, i€{1,2,3} mogelijk voor één node U[α] (namelijk meerdere adviesresultanten afkomstig van meerdere units V). Deze discrepantie tussen de verschillende adviezen omtrent P(U[α]) is nu juist datgene wat SCORE voor ons moet oplossen! Het

redeneerproces is gebaseerd op het zoeken naar dié unitinstellingen (activeringspatroon van unitnodes) waarbij voor iedere unit uiteindelijk geldt dat zowel parents, childs als perceptor de unitbesturing hetzelfde advies geven t.a.v. activering van een node. M.a.w. als voor unit U na een redenatie geldt dat $r_{1,\tau}$ de hoogste is, dan moeten -voor zover aanwezig- ook $r_{2,\tau}$ en $r_{3,\tau}$ de hoogste zijn: er is consensus.

Hetzelfde geldt als we het geheel bekijken vanuit de waarneming. Als op unit U bijvoorbeeld de waarneming gepleegd wordt (V=U'), dan moeten $r_{1,\tau}$ en $r_{2,\tau}$ van unit U uiteindelijk hetzelfde adviseren als $r_{3,\tau}$: het systeem heeft zich dan geschikt naar de waarneming. Als er geen waarneming op de unit U wordt gedaan, dan kan de unit zich vrijer bewegen en heeft het alleen te maken met child- en parentadviezen.

Verwerking van de asserties in de verbindingen

Voordat $r_{1,\alpha}$, $r_{2,\alpha}$ en $r_{3,\alpha}$ formeel kunnen worden gedefinieerd en we toe zijn aan de bespreking van het redenatieproces, is het eerst van belang dat duidelijk is hoe uitspraken in termen van eigenschappen kunnen worden verwerkt in de verbindingen tussen de units. Beschouw de volgende figuur met bijbehorende waarheidstabel:



A	B	C
0	0	0
0	1	1
1	0	2

Fig. 5.2.b Netwerk met bijbehorende waarheidstabel

Puur op grond van de waarheidstabel kunnen we conditionele kansen tussen de instanties opstellen. Voor de relaties tussen de instanties van de eigenschappen A en C geldt:

P(C[O]|A[0]) = 0,5 P(C[O]|A[1]) = 0
P(C[1]|A[0]) = 0,5 P(C[1]|A[1]) = 0
P(C[2]|A[0]) = 0 P(C[2]|A[1]) = 1

en omgekeerd:

P(A[0]|C[0]) = 1	P(A[0]|C[1]) = 1	P(A[0]|C[2]) = 0
P(A[1]|C[0]) = 0	P(A[1]|C[1]) = 1	P(A[1]|C[2]) = 0

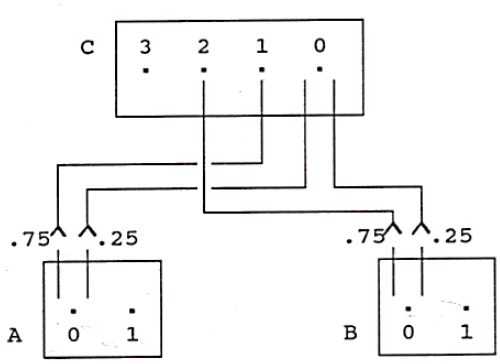
Dergelijke conditionele kansen zijn ook op te stellen voor de unitcombinatie B, C. De verbindingen tussen de nodes krijgen op deze manier hun gewicht. Er zijn geen verbindingen nodig tussen de units A en B.

In bovenstaand voorbeeld kunt U ook zien dat, gegeven een instantie, met kans één aan te geven is wat de bijbehorende childinstanties moeten zijn. Omgekeerd echter: gegeven een instantie, dan is niet met kans één aan te geven wat de bijbehorende parentinstantie is. Dit is een belangrijk aspect waar later op wordt terug gekomen.

Nu blijkt $r_{1,\alpha}$ met i€{1, 2, 3 } de drie resultanten van de adviezen van respectievelijk parent-, child en perceptorunits, niet altijd rechtstreeks uit i(V,α) te bepalen te zijn. Hoe $r_{1,\alpha}$ wel bepaald kan worden, wordt in het hiernavolgende in drie delen uitgelegd. Er wordt begonnen met $r_{2,\alpha}$ om via $r_{1,\alpha}$ te eindigen bij $r_{3,\alpha}$.

Berekening $r_{2,\alpha}$

Beschouw figuur 5.2.c. Hierin ziet U een eenvoudig netwerk met drie units en een aantal verbindingen, die zo ontstaan zijn na aandiening van de erbij afgedrukte trainingen. Subparagraaf 5.2.2 gaat uitgebreid in op trainingen, maar deze figuur zal duidelijk zijn: in bijvoorbeeld de vier gevallen van A[0], hoort daar maar één keer C[0] en drie keer C[1] bij. De kansen volgen hier rechtstreeks uit. In datgene wat nu besproken zal worden, zijn alleen de verbindingen vanuit A[0] en B[0] interessant en zijn daarom alleen dié verbindingen getekend.



A	B	C
0	0	0
0	1	1
1	0	2
1	1	3
0	1	1
0	1	1
1	0	2
1	0	2

Fig. 5.2.c Netwerk met bijbehorende trainingsvoorbeelden waardoor de verbindingen vanuit A[0] en B[O] hun waarde krijgen zoals aangegeven.

Stel namelijk dat A[0] en B[O] actief zijn, d.w.z. o(A[0])=o(B[O])=1 en dus o(A[1])=o(B[1])=0. In figuur 5.2.d is aangegeven wat de waarden van i(V,C[α]) als functie van α dan worden, met V€{A,B}.

α	C[0]	C[1]	C[2]	C[3]
i(A,α)	.25	.75	0	0
i(B,α)	.25	0	.75	0
gem(α)	.25	.375	.375	0
prod(α)	.125	0	0	0
$r_{2,\alpha}$	.5	.25	.25	0

Fig. 5.2.d i(A,α), i(B,α), gem(α), prod(α) en $r_{2,\alpha}$ als functie van C[α], waarbij gem(α) = (i(A,α) + i(B,α))/2 en prod(α)=i(A,α)*i(B,α). $r_{2,\alpha}$ is de resultante waarvan de definitie in de tekst is beschreven.

In figuur 5.2.c ziet U ook wat er gebeurt als we de i(V,α)'s middelen: Als A[0] en B[O] gelden, worden C[1] en C[2] geadviseerd (daarvan is het gemiddelde het hoogste). Dit is niet de bedoeling. Als we naar de trainingsvoorbeelden kijken zien we dat C[O] bij (A[0],B[O]) hoort. Dit komt dus blijkbaar niet tot uitdrukking in het gemiddelde gem(0). Waar dit wel tot uitdrukking komt is het product prod(α), zoals U ook kunt zien in figuur 5.2.c. Het product is voor C[0] het hoogste, of sterker nog: de enige die ongelijk is aan nul.

Een probleem is dat prod(α) erg weinig informatie bevat. Het enige dat C nu kan vernemen is het feit dat A[O] en B[O] actief zijn. Toch willen we unit C informatie verschaffen omtrent redelijke alternatieven voor de activering van C[0]. Want tijdens het inferentieproces willen we ook dié alternatieven voor C[0] kunnen proberen. Want stel dat unit C een groot aantal childunits heeft die allen op één unit na, node C[α] adviseren. Via het product alleen is niet te achterhalen dat node C[α] heel dicht bij de oplossing ligt. Door de foute instelling van die ene unit is het product prod(α) dan namelijk nul.

Binnen SCORE is gekozen voor de combinatie van product en gemiddelde. En wel zó dat die node waarvoor geldt dat prod(α) ongelijk is aan nul, minimaal de kans O.5 krijgt toebedeeld en dus zeker de grootste wordt. De resterende kans van 0.5 wordt over alle nodes verdeeld via een weging die is afgeleid van gem(α). Dit geschiedt op de volgende manier. Stel dat voor node τ van unit U geldt dat prod(τ) > 0. Node τ is dan dus ook de enige binnen U waarvoor dat

geldt. Deze krijgt dan de waarde van 0.5 (waardoor het in ieder geval winnaar van de unit wordt) plus een bonus van gem(τ) - 1/n, waarbij n het aantal nodes is van unit U. Het idee hierachter is dat hoe hoger gem(τ), hoe meer het (boven op het feit dat het volledig door zijn childs wordt ondersteund en dus minimaal kans 0. 5 krijgt) beloond moet worden. Het hoog zijn van gem(τ) wordt gerelateerd aan het aantal nodes van de unit. Hoe meer nodes, hoe lager de kansen gem(α), zullen uitvallen. 1/n is ook de gemiddelde kans die bij de nodes van unit U binnenkomt. Dit zal daarom verder de verwachting E(U) worden genoemd: E(U)=1/n. De bonus is dan dus het verschil van gem(τ) met de verwachting E(U): gem(τ) -E(U). Het resultaat $r_{2,\tau}$, de resultante van de childunits, wordt dus als in formule 5.2.f.

> $r_{2,\tau}$ = 0.5 + (gem(τ)-E(U)), waarbij τ die node is waarvoor geldt dat prod(τ) <> 0, met een maximum van 1 omdat het kansen betreft. In dat geval wordt gem(τ) -E(U) op 0.5 gesteld.
>
> **Formule 5.2.f.**

Totaal valt er nu nog (0.5 - (gem(τ)-E(U))) te verdelen over de overige nodes, dus de nodes α waarvoor geldt dat prod(α)=0, van unit U. Als $r_{2,\tau}$ 1 was geworden is gem(τ)-E(U) dus gelijk aan 0.5 en is er nog (0.5-0.5)=0 te verdelen. Totaal komt er nog (1-gem(τ)) aan gemiddeldes bij die overige nodes α binnen. $r_{2,\alpha}$ wordt nu als in formule 5.2.g.

> r2,α = gem(α) * [(0.5 - (gem(τ)-E(U))) / (1-gem(τ))], waarbij α die nodes zijn waarvoor geldt dat prod(α) =0 en τ die node is waarvoor geldt prod(τ) > 0.
>
> **Formule 5.2.g**

Voor zowel $r_{2,\tau}$ als $r_{2,\alpha}$ geldt ook dat wanneer (gem(τ)-E(U))<0, voor deze uitdrukking toch de waarde 0 wordt ingevuld. Hierdoor kan r2,τ nooit kleiner dan 0.5 worden. Dat was immers onze eis.

Het is in principe ook mogelijk dat er geen enkele node τ bestaat waarvoor geldt dat prod(τ)>0. De childunits hebben dan een inconsistente combinatie van waarden te pakken. De oplossing hiervoor is eenvoudigweg dié node te activeren die het dichtste ligt bij de adviezen i(V,α) van de childs. Hiervoor is gem(α) gebruikt. Dus geldt formule 5. 2. h.

> $r_{2,\alpha}$ = gem(α), als er geen enkele node τ is waarvoor geldt dat prod(τ) >0.
>
> **Formule 5. 2. h**

Deze invulling aan $r_{2,\alpha}$ in het geval van een inconsistente waarneming is in wezen een bijzonder geval van de formule 5.2.g. Er geldt nu namelijk niet meer dat na node τ er nog (0.5 - (gem(τ)-E(U))) is te verdelen, maar nog de totale kans één. Verder is gem(τ) =0, waardoor 1-

gem(τ)=l. Hierdoor gaat formule 5.2.g voor een inconsistente situatie al eigenlijk automatisch over in 5.2.h.

Als deze formules worden toegepast op figuur 5.2.c, krijgen we de resultaten voor $r_{2,\alpha}$ zoals aangegeven in figuur 5.2.d. A[0], B[1] betreft een consistente situatie dus gelden de formules 5.2.f en 5.2.g. E(C) is hierbij gelijk aan 1/4. Unit C bevat immers vier nodes. Bij het actief zijn van A[0] en B[0] is prod(C[0]) ongelijk aan nul en voor die node geldt dan ook formule 5.2.f, waarbij τ=C[0]. Voor de overige drie nodes geldt formule 5.2.g. Omdat gem(C[0]) precies gelijk is aan E(C), wordt de bonus nul en blijft $r_{2,\alpha}$ gelijk aan 0.5. In figuur 5.2.d kunt U zien dat de kansen nu logischer zijn verdeeld dan een product of gemiddelde dat alleen kan veroorzaken.

In het vervolg zal de formule voor het berekenen van $r_{2,\alpha}$ worden aangeduid met de naam: f_{par}. f_{par} beschrijft dus alle bovenbeschreven situaties met bijbehorende oplosmethodes.

Berekening $r_{1,\alpha}$

De berekening van de parentresultante $r_{2,\alpha}$ kan eenvoudiger dan de berekening van de childresultante $r_{1,\alpha}$. Beschouw figuur 5.2.e.

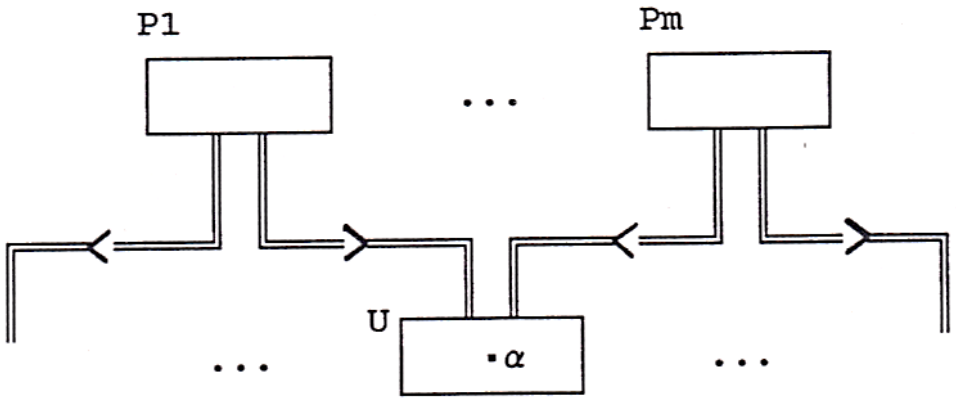


Fig. 5.2.e De invloed van Parentunits P1 t/m Pm op U[α].

Unit U is nu childunit voor alle units P1 t/m Pm. Wegens de omkeerbaarheidseis geldt dat als een instantie Pj[δ] j€{1, ... ,m} geldt, daar een eenduidige combinatie van childinstanties bij hoort. Units P1 t/m Pm zullen, gegeven hun instellingen, dan ook stuk voor stuk een ondubbelzinnig advies i(Pj,α) uitbrengen aan de nodes α van U. In tegenstelling tot de childs van U spannen de parents P1 t/m Pm nu niet U op. We hebben het product daarom nu niet nodig. De adviezen die iedere unit Pj aan U in de vorm van i(Pj,α) geeft, staan los van elkaar

en moeten met elkaar in overeenstemming worden gebracht. $r_{1,\alpha}$ wordt daarom berekend door middeling van die adviezen. Dit is weergegeven in formule 5.2.i.

$r_{1,\alpha} = 1/m * \sum_j i(Pj,\alpha)$, waarbij m het aantal parentunits Pl t/m Pm

Formule 5.2.i.

In het vervolg zal de formule voor het berekenen van $r_{1,\alpha}$ worden aangeduid met de naam: f_{chi}.

Berekening $r_{3,\alpha}$

Beschouw figuur 5.2.f.

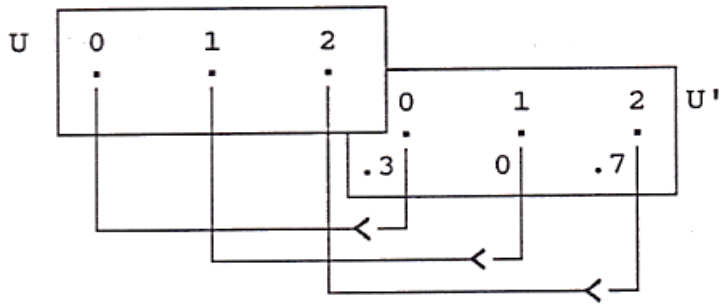


Fig. 5.2.f Een voorbeeld van de clamping van de perceptor

In deze figuur ziet U dat er een onzekere waarneming wordt aangeboden aan de perceptor. De perceptor heeft de waarden als in figuur 5.2.f aangegeven, gekregen. Dit wordt ook de output van de perceptor, dus o(U'[0])=0.3, o(U'[1])=0 en o(U'[2])=0.7. De functie van de perceptor is het adviseren van unit U, net zoals parentunits en childunits dat bij unit U doen. Doordat de verbindingen tussen de nodes van U[α] en U'[α], met α € {0,1,2}, één zijn is de input bij U[α] afkomstig van U', i(U',U[α])=o(U'[α]). Nu kan $r_{3,\alpha}$ heel eenvoudig worden gedefinieerd als in formule 5.2.j.

$R_{3,\alpha} = i(U',U[\alpha])$

Formule 5.2.j

Hoe hoger de onzekere waarneming op U[α]: o(U'[α]), hoe evenredig hoger het advies $r_{3,\alpha}$. Het is de taak van het inferentiemechanisme ervoor te zorgen dat de node met een zo groot mogelijke $r_{3,\alpha}$ wordt geactiveerd. Als dat niet kan wordt een met lagere genoegen genomen, enzovoorts.

In het vervolg zal de formule voor het berekenen van $r_{3,\alpha}$ worden aangeduid met de naam: f_{per}.

$r_{1,\alpha}$, $r_{2,\alpha}$, $r_{3,\alpha}$: kansmaat

Nu $r_{1,\alpha}$, $r_{2,\alpha}$ en $r_{3,\alpha}$ zijn gedefinieerd, moet voor de volledigheid worden bewezen dat zij, zoals eerder gesuggereerd, ook inderdaad een kansmaat voorstellen. Dit bewijs verloopt in een aantal stappen.

Stelling:

$$\sum_{\alpha} i(V,\alpha) = 1.$$

Formule 5.2.k.

Bewijs:

$i(V,\alpha) = \sum_{\delta} W(V[\delta],\alpha) * o(V[\delta])$ (zie formule 5.2.c)

Er is van unit V slechts één node τ actief, d.w.z. $o(V[\tau])=1$ en $o(V[\delta])=0$ voor $\delta <> \tau$. Hieruit volgt:

$i(V,\alpha) = W(V[\tau],\alpha) = P(U[\alpha]|V[\tau])$. Dus:

$\sum_{\alpha} i(V,\alpha) = \sum_{\alpha} P(U[\alpha]|V[\tau]) = 1$, op grond van formule (5. 2. e).

Einde bewijs

Alle parentadviezen i_{Pk},α worden gemiddeld in $r_{1,\alpha}$ (zie formule 5.1.a). Ook $r_{1,\alpha}$ blijft een kansmaat.

Stelling:

$$\sum_{\alpha} r_{1,\alpha} = 1$$

Formule 5.2.l.

Bewijs:

$r1,\alpha = 1/m * \sum_k i(Pk,\alpha)$

$\sum_\alpha r_{1,\alpha} = \sum_\alpha 1/m * \sum_k i(Pk,\alpha) = 1/m * \sum_k \sum_\alpha i(Pk,\alpha) =$

$1/m * \sum_k 1 = 1/m * m = 1$ (Op grond van het feit dat er m parents zijn en formule 5.2.k)

Einde bewijs

Een ander verhaal geldt voor $r2,\alpha$: de resultaten van de childunits.

Stelling:

$\sum_\alpha r2,\alpha = 1.$

Formule 5. 2. m

Bewijs

Op grond van de formules 5.2.f en 5.2.g kan het volgende worden gesteld waarbij $A=gem(\tau)-E(U)$.

$\sum_\alpha r_{2,\alpha} = r_{2,\tau} + \sum_{\alpha<>\tau} gem(\alpha) * [(0.5 - A)/(1 - gem(\tau))]$

$= 0.5 + A + (0.5 - A)/(1-gem(\tau)) * \sum_{\alpha<>\tau} gem(\alpha)$

$= 0.5 + A + (0.5 - A)/(1-gem(\tau)) * (1-gem(\tau))$

$= 1$

Einde bewijs

Aangezien de waarneming een consistente kansverdeling over U' moet zijn geldt ook automatisch voor $r_{3,\alpha}$, dat het een kansmaat is.

Stelling:

$\sum_\alpha r_{3,\alpha} = 1$

Formule 5.2.n.

Bewijs

$$\sum r_{3,\alpha} = \sum_{\alpha} o(U'[\alpha]) = 1$$

Einde bewijs

In het algemeen geldt dus:

$\sum_{\alpha} r_{i,\alpha} = 1$ met i € {1, 2, 3}

$r_{3,\alpha}$ kan trouwens alleen maar gelijk zijn aan o(U'[α]) als de verbindingen tussen U[α] en U'[α] één zijn en de overige nul. Dit klopt ook want natúurlijk geldt: P(U[α]|U'[α]) = 1. Immers, als we U[α] waarnemen willen we dat U[α] geconcludeerd wordt en niets anders.

De unitbesturing bepaalt vervolgens welke ene node α actief, dus één, mag worden en zorgt daarmee dat $\sum_{\alpha} o(U[\alpha]) = 1$.

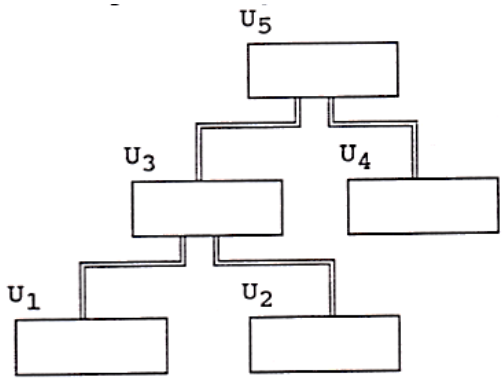
Conclusie

Het kringetje is nu keurig rond: het hele redenatieproces is een consistente manipulatie van kansen. Door de heuristieken zijn de kansen echter een deel van hun wiskundig fundament kwijt geraakt. Wat er van over is gebleven, is een kansmaat die niet gelijk op loopt met de wiskunde. Toch liggen er wel belangrijke relaties tussen de heuristische en de wiskundige kansmaat. Dit geldt sowieso voor $r_{1,\alpha}$ en $r_{3,\alpha}$. Als het redeneermechanisme namelijk probeert overeenstemming te bereiken in deze twee resultanten, probeert het volledig volgens de Stelling van de Totale Waarschijnlijkheid, die hiervoor is besproken, overeenstemming te brengen in puur wiskundige kansadviezen. Als dat is gelukt mag men puur wiskundig stellen dat geldt: o(U[α])=P(U[α]). We hebben nu echter nog een $r_{2,\alpha}$, die wat dat betreft roet in het eten gooit. $R_{2,\alpha}$, was ingewikkelder gedefinieerd en was niet één of andere middeling van i(V,α)'s (waarbij V€{C1, ... , Cn}). Met de afwijkende definitie van $r_{2,\alpha}$ is er voor gezorgd dat het belangrijkste criterium voor het unitadvies d.m.v. $r_{2,\alpha}$, een volledige childunit ondersteuning is. Daar bovenop, om het inferentiemechanisme de mogelijkheid te geven ook alternatieven te onderzoeken, werd een bonus-advies gedefinieerd die rechtstreeks was afgeleid van de Stelling van de Totale Waarschijnlijkheid. Hiermee is een $r2_{,\alpha}$ gedefinieerd waardoor we niet meer mogen stellen o(U[α])=P(U[α]), maar wel doet wat het doen moet, namelijk op basis van zijn childunits een consistente node selecteren, en er verder voor zorgt dat de zuiver wiskundige definitie van de verbindingen een belangrijke rol spelen in het zoeken naar die consistente selectie.

5.2.2 Training

Voordat er geredeneerd kan worden moet er eerst een kennisstructuur worden opgezet. Zoals we in subparagraaf 5.1.2 hebben gezien geschiedt dit aan de hand van een afhankelijkheidsgraaf. Als de structuur is gedefinieerd is er nog niets bekend over de instanties van de eigenschappen en hun onderlinge relaties. De structuur is een afhankelijkheidsbeschrijving op eigenschapniveau. In deze subparagraaf wordt beschreven hoe, uitgaande van trainingsvoorbeelden, de nodes en hun onderlinge relaties kunnen worden afgeleid.

Om te beginnen: hoe ziet een training in SCORE eruit? Beschouw figuur 5.2.g.



U_1	U_2	U_3	U_4	U_5	
0	0	0	0	0	U_1
0	0	0	0	0	U_2
1	1	0	0	0	U_3
0	0	0	0	0	U_4
0	0	1	1	0	U_5

Fig. 5.2.g. Een netwerkstructuur met bijbehorende Afhankelijkheidsgraaf

Trainingen zijn gebaseerd op het type 'easy learning': voor elke unit U_i moet worden gespecificeerd wat z'n waarde is [Gallant, 1988]. Iedere trainingsvoorbeeld bestaat uit een consistente set unitwaarden. Stel verder dat de 5 eigenschappen U_1 t/m U_5 uit figuur 5.2.g binair, d.w.z. tweewaardig, zijn. M.a.w. iedere U_i, waarbij $1 \leq i \leq 5$, kent slechts twee instanties. In de praktijk mogen de eigenschappen meerwaardig zijn, maar het hiernavolgende wordt er duidelijker van als we voorlopig tweewaardige eigenschappen beschouwen. Mogelijke trainingsvoorbeelden (Training Examples TE) zijn:

U_1	U_2	U_3	U_4	U_5	
0	0	1	0	0	TE#1
1	0	0	1	0	TE#2
1	1	1	0	1	TE#3
1	1	1	1	1	TE#4
1	1	0	1	1	TE#5

Tabel 5.2.a Waarheidstabel met trainingsvoorbeelden (Training Examples TE).

In subparagraaf 5.1.1. is aangegeven dat asserties aan de omkeerbaarheidseis moeten voldoen. SCORE herstelt automatisch omkeerbaarheidsfouten. Dit wordt gedaan door ten eerste het door de gebruiker gedefinieerde netwerk te transformeren naar een netwerk, uitgebreid met 'hidden units'. Ten tweede door een consistente node representatie te genereren. Deze twee zijn rechtstreekse gevolgen van de ideeën zoals die in subparagraaf 5.1.1 zijn beschreven: als de eigenschappen van het predicaat van een assertie niet volledig zijn in de beschrijving van het subject wordt er een node toegevoegd en als het subject dubbele instanties voor verschillende predicaatinstanties heeft, worden er nodes toegevoegd. De toegevoegde units zijn de hidden units. Het verschil tussen de door de gebruiker gedefinieerde instanties en de uiteindelijk nodeconfiguratie wordt tot uitdrukking gebracht in de begrippen 'externe nodes' respectievelijk 'interne nodes' .

Aan de hand van het voorbeeld van figuur 5.2.g kan dit worden verduidelijkt. Op verschillende punten wordt daar de omkeerbaarheidseis geschonden. Bijvoorbeeld $U_1[1]$, $U_2[1]$ wordt zowel afgebeeld op $U_3[0]$ als $U_3[1]$. Dit betekent dat U1 en U2 alleen, blijkbaar onvoldoende zijn voor de éénduidige beschrijving van alle instanties van U_3. Dit moet worden opgelost door een extra unit, de hidden unit, toe te voegen. Binnen SCORE wordt gesteld dat dit automatisch, dus zonder dat de gebruiker daar notie van heeft, kan gebeuren. Binnen SCORE-S is dit nog niet geïmplementeerd. Onder unit U_3 moet een extra unit worden toegevoegd, zodat $U_3[0]$ en $U_3[1]$ van elkaar kunnen worden onderscheiden. Zie hiervoor figuur 5.2.h. U_H is de naam van de hidden unit.

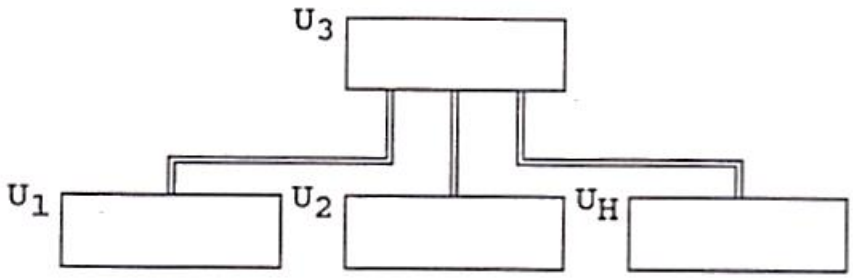


Fig. 5.2.h. Een deel van het netwerk van figuur 5. 2.g, met een unit U_H toegevoegd om de omkeerbaarheidseis naar één kant toe te herstellen.

De waarheidstabel kan bijvoorbeeld worden gedefinieerd als in tabel 5.2.b.

U_1	U_2	UH	U_3
0	0	0	1
1	0	0	0
1	1	0	1
1	1	0	1
1	1	1	0

Tabel 5.2.b Waarheidstabel behorende bij figuur 5.2.h. Hiermee is de omkeerbaarheidseis in dié zin hersteld dat bij iedere combinatie van U_1, U_2 en U_H precies één waarde voor U_3 hoort.

De waarde die U_H heeft gekregen voor de eerste twee combinaties van U_1 en U_2 had ook één mogen zijn. Het gaat erom dat de twee verschillende waarden van U_3 voor $U_1[1]$ en $U_2[1]$ van elkaar worden onderscheiden door voor de ene de waarde $U_H[0]$ en voor de andere de waarde $U_H[1]$ te introduceren.

Nu wordt nog steeds niet in zijn geheel aan de omkeerbaarheiseis voldaan: $U_3[1]$ wordt nu nog steeds beschreven door twee combinaties van U_1, U_2 en U_H, namelijk ($U_1[0]$, $U_2[0]$, $U_H[0]$) en ($U_1[1]$, $U_2[1]$, $U_H[0]$). Via een extra node in U_3 moeten deze twee van elkaar worden onderscheiden. Ook t.a.v. $U_3[0]$ is de omkeerbaarheidseis niet gewaarborgd. Ook hiervoor dient dus een extra node te worden geïntroduceerd. Binnen SCORE-S geschiedt het op deze manier herstellen van de omkeerbaarheidseis automatisch. In tabel 5.2.c ziet U het resultaat. Hierbij wordt onderscheid gemaakt tussen interne en externe nodes. Intern wordt er altijd aan de omkeerbaarheidseis voldaan. Deze interne nodes bepalen hoe het netwerk fysiek is opgebouwd. De externe nodes bepalen wat er zichtbaar is naar de gebruiker toe. De externe nodes zijn op te vatten als de instanties. Voor één instantie kunnen dus meerdere interne nodes aanwezig zijn. Tussen interne en externe nodes ontstaan nu bepaalde relaties zoals die zijn gespecificeerd in dezelfde tabel 5.2.c.

U_1	U_2	UH	U_3-e	U_3-i
0	0	0	1	0
1	0	0	0	1
1	1	0	1	2
1	1	0	1	2
1	1	1	0	3

Tabel 5.2.c Waarheidstabel behorende bij figuur 5.2.h. Hiermee is de omkeerbaarheidseis in beide richtingen hersteld.

Als bijvoorbeeld na het redeneren U_3-i[1] wordt geconcludeerd zal naar de gebruiker toe de bijbehorende externe node of instantie $U_3[0]$ zichtbaar worden.

Doordat U_3-e[0] is onderscheiden in U_3-i[1] en U_3-i[3] wordt het mogelijk de onvolledige waarneming U_1[1] via U_3 te completeren tot ofwel (U_1[1], U_2[0]), ofwel (U_1[1],U_2[1]). Via de sterktes van de verbindingen met U_3 kan dit worden geregeld. De hiddenunit maakt het mogelijk de voorkeur te regelen voor U_3-e[0], ofwel U_3-e[1] bij de waarneming (U_1[1], U_2[1]).

U_1, U_2 en U_3 uit figuur 5.2.g zijn verwerkt. Vervolgens kan U_4 erbij worden gehaald. Voor deze unit hoeft, net als de units 1 en 2, geen onderscheid gemaakt te worden tussen interne en externe waarden omdat voor hen de omkeerbaarheidseis naar beneden toe automatisch is gewaarborgd: zij bevatten geen childunits. Zie hiervoor tabel 5.2.d.

U_1	U_2	U_H	U_3-e	U3-i	U_4
0	0	0	1	0	0
1	0	0	0	1	1
1	1	0	1	2	0
1	1	0	1	2	1
1	1	1	0	3	1

Tabel 5.2.d. Waarheidstabel met getransformeerde trainingsvoorbeelden

Tot slot moet U_5 nog aan de waarheidstabel worden toegevoegd. Om te bepalen of t.a.v. U_5 de omkeerbaarheidseis is gewaarborgd moeten we kijken naar U_3 en U_4, en wel naar hun interne waarden! Een instantie die wordt gerepresenteerd door twee nodes is immers uiteengevallen in twee instanties. Aan de buitenkant is er maar één instantie zichtbaar, maar dat is voor de omkeerbaarheidseis natuurlijk niet van belang. Voor U_5[0] geldt dat deze wordt vastgelegd door zowel U_3-i[0], U_4[0] als U_3-i[1], U_4[1]. Hierin is de omkeerbaarheidseis dus ook niet gewaarborgd. Met het introduceren van een extra node binnen U_5 kan dit euvel worden verholpen. Ook t.a.v. U5[1] blijkt de omkeerbaarheidseis niet gewaarborgd. Daar dient dus nog een node te worden geïntroduceerd. Verder zien we nog iets vreemds t.a.v. U5. De omkeerbaarheidseis naar de andere kant toe lijkt ook te worden overtreden: U_3[1], U_4[0] leiden zowel tot U_5[0] als U_5[1]. Toch is dit een schijnprobleem omdat U_3[1] precies voor dié asserties onderverdeeld is in een U_3-i[0] en een U_3-i[2]. En zoals zojuist is opgemerkt moet er naar de interne waarden worden gekeken. De omkeerbaarheidseis blijkt die kant op dus toch te worden gewaarborgd. De complete waarheidstabel, inclusief U_5, is tabel 5.2.e geworden.

U_1	U_2	U_H	U_3-e	U3-i	U_4	U_5-e	U_5-i
0	0	0	1	0	0	0	0
1	0	0	0	1	1	0	1
1	1	0	1	2	0	1	2
1	1	0	1	2	1	1	3
1	1	1	0	3	1	1	4

Tabel 5.2.e Complete waarheidstabel met getransformeerde trainingsvoorbeelden

Deze waarheidstabel is een transformatie van de voorbeelden waar we deze subparagraaf begonnen. Deze tabel is consistent en voldoet aan alle eisen. Als nu bijvoorbeeld U_5-e[0] wordt waargenomen is het netwerk in staat onderscheid te maken tussen de externe completeringen (U_1[0], U_2[0], U_3-e[1], U_4[0], U_5-e[0]) en (U_1[1], U_2[0], U_3-e[0], U_4[1], U_5-e[0]). Bij aanvang zal het netwerk beide completeringen even goed vinden, maar als het ene patroon vaker wordt waargenomen dan het andere zal (als gevolg van het leeralgoritme dat wordt besproken in paragraaf 5.4) onderscheid gemaakt kunnen worden tussen beide. Als er geen extra interne nodes werden geïntroduceerd voor U_5-e[0], had dat niet gekund.

Het netwerk van figuur 5.2.g ziet er aangevuld met de interne nodes en de hidden unit uit als in figuur 5.2.i.

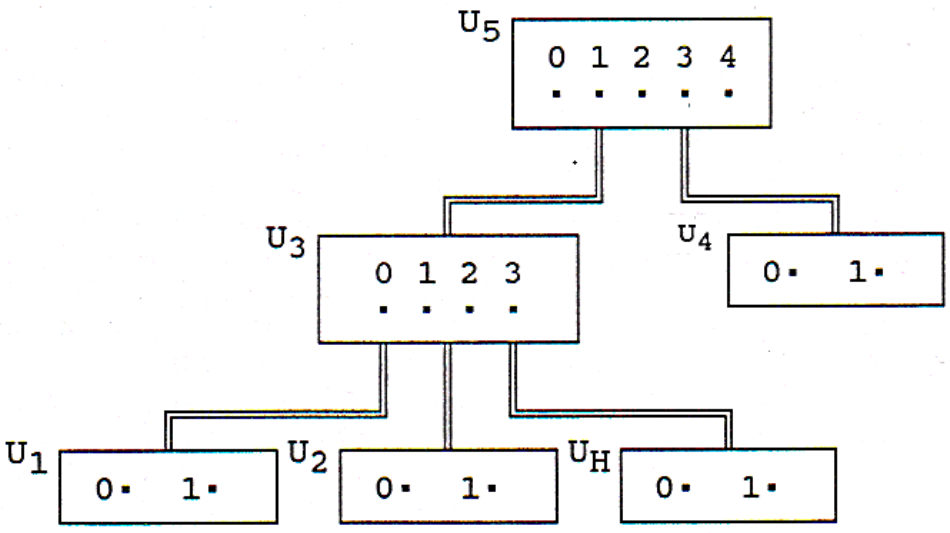


Fig. 5.2.i. De netwerkstructuur van figuur 5.2.g met nodes en hidden unit na de training.

De volgende stap is het vinden van waarden voor de verbindingen tussen de nodes. Dit kan heel eenvoudig. In subparagraaf 5.2.1 is uitgelegd dat de verbindingen tussen de nodes voor

conditionele kansen staan. Uitgaande van tabel 5.2.e zijn die kansen voor bijvoorbeeld U_3 (en dan gaat het natuurlijk om U_3-i):

$P(U_3[0]|U_1[0])=1$
$P(U_3[1]|U_1[0])=0$
$P(U_3[2]|U_1[0])=0$
$P(U_3[3]|U1[0])=0$

$P(U_3[0]|U_2[0])=0.5$
$P(U_3[1]|U_2[0])=0.5$
$P(U_3[2]|U_2[0])=0$
$P(U_3[3]|U_2[0])=0$

$P(U_3[0]|U_1[1])=0$
$P(U_3[1]|U_1[1])=0.25$
$P(U_3[2]|U_1[1])=0.5$
$P(U_3[3]|U_1[1])=0.25$

$P(U_3[0]|U_2[1])=0$
$P(U_3[1]|U_2[1])=0$
$P(U_3[2]|U_2[1])=0.66$
$P(U_3[3]|U_2[1])=0.33$

De overige conditionele kansen worden hier niet gegeven, maar kunnen ook eenvoudig worden bepaald uit onze waarheidstabel. U ziet dus dat alle trainingsvoorbeelden op deze manier kunnen worden verwerkt in het netwerk: ongeacht hoeveel het er zijn! Mits ze voldoen aan de eisen als hierboven besproken.

Als nu bijvoorbeeld U_1[0] wordt waargenomen, wordt node U_1[0] geactiveerd en moet het redeneerproces er dus voor zorgen dat intern (U_1, U_2, U_3, U_4, U_5) = (0,0,0,0,0) wordt gevonden. De gebruiker krijgt (0, 0, 1, 0,0) te zien. Als instantie U_3[1] door de gebruiker wordt waargenomen moeten dus zowel node U_3[0] als U_3[2] worden geactiveerd en kunnen de afleidingen (0, 0, 0, 0, 0) , (1, 1, 2, 0, 1) én (1, 1, 2, 1, 1) worden gemaakt. Hoe waarnemen, inferentie en concludering precies geschiedt, komt in paragraaf 5.3 uitgebreid aan de orde.

Wat zijn nu precies de kenmerken van een netwerk dat op deze manier ontstaat vanuit een afhankelijkheidsgraaf en trainingsvoorbeelden? Zoals we in het voorgaande voorbeeld hebben kunnen zien bevat een unit zoveel nodes als nodig om alle in de trainingsvoorbeelden aanwezige childcombinaties te beschrijven. Zo bevatte U_3 uiteindelijk vier nodes omdat over U_1, U_2 en U_H vier van de acht mogelijke combinaties voorkwamen. De combinatie (U_1[0], U_2[1]) kwam niet voor en wordt daarom binnen U_3 ook (nog) niet gecodeerd. Verder kwam de combinatie (U_1[1], U_2[1]) dubbel voor en werd maar één keer gecodeerd binnen U_3. Op het niveau van U_5 gebeurt hetzelfde: alle in de trainingsvoorbeelden aanwezige combinaties van U_3 en U_4 worden gecodeerd binnen U_5. Nu zijn dat echter geen combinaties van de waarden zoals die in de trainingsvoorbeelden expliciet aanwezig zijn (de externe instanties), maar combinaties van de uiteindelijke nodes (de interne instanties). U_3 bevat vier nodes, U_4 twee. U_5 kan daardoor maximaal acht nodes bevatten. Uiteindelijk zijn het er in ons voorbeeld vijf geworden: voor ieder uniek trainingsvoorbeeld één. Iedere node op het hoogste niveau staat voor een eigen combinatie van waarden van units op het laagste niveau. Dit houdt dus in dat iedere node binnen U_5 voor een eigen trainingsvoorbeeld staat: tijdens het trainingsproces zal elk nieuw type voorbeeld, dubbele trainingsvoorbeelden uitgesloten, leiden tot de aanmaak

van een node op het hoogste niveau. Op lage niveaus geldt hetzelfde verhaal voor subvoorbeelden. Het trainingsalgoritme zorgt dus voor een compacte opslag van de trainingsvoorbeelden om ze vervolgens zó te kunnen gebruiken dat onvolledige, inconsistente of onzekere waarnemingen over deze kennis kunnen worden gepleegd. Het is dan de taak van het redeneermechanisme het zo bijpassend mogelijk trainingsvoorbeeld terug te zoeken.

5.3 Consultatie

Als netwerkstructuur, dat wil dus zeggen units, nodes en verbindingen, zijn gedefinieerd kunnen we beginnen aan een 'consultatie sessie'. De expert is nu niet meer nodig en we hebben alleen nog te maken met de gebruiker die zijn vragen aan het systeem stelt. De consultatiesessie heeft tot doel gebruikersvragen zo goed mogelijk te beantwoorden. Een consultatie sessie bestaat uit drie fasen:

1. Waarneming
2. Inferentie
3. Concludering

Aan de hand van het netwerk dat in de vorige paragraaf is gecreëerd (zie figuur 5.2.i), worden deze drie fasen in deze paragraaf besproken.

Als in deze paragraaf wordt gesproken over 'de waarde van $U_i[j]$' , wordt daar de output van $U_i[j]$ mee bedoeld. Dus $U_i[j]=o(U_i[j])$. Bijvoorbeeld $U_1[0]=0.5$, houdt in dat de output van $U_1[0]$, die staat voor de kans op $U_1[0]$, 0.5 is.

5.3.1 Waarneming

Waarnemingen kunnen worden uitgevoerd door het op de juiste manier activeren van bepaalde nodes. Als bijvoorbeeld $U_1[0]$ wordt waargenomen, moet op de één of andere manier $U_1[0]$ worden geactiveerd en daarmee $U_1[1]$ worden gedeactiveerd, d.w.z. $U_1[0]=l$ en $U_1[1]=0$.

Een waarneming kan ook onzeker zijn, bijvoorbeeld $U_1[0]=0.75$, $U_1[1]=0.25$. Zoals in subparagraaf 5.2.1 opgemerkt, moeten de som van de kansen per unit één zijn. Dat geldt ook voor de waarneming.

Verder moet er een transformatie van kansen plaatsvinden als de interne nodes niet overeenkomen met de externe (zie subparagraaf 5.2.2). Als $U_3[1]$ bijvoorbeeld wordt waargenomen, moeten de nodes U_3-i[0] en U_3-i[2] worden geactiveerd en U_3-i[1] en U_3-i[3] worden gedeactiveerd. We kunnen de waarneming van de instantie $U_3[1]$, m.a.w. de externe node U_3-e[1], gelijkelijk verdelen over de interne nodes U_3-i[0] en U_3-i[2]: d.w.z. U_3-i[0]=U_3-i[2]=0.5. Het is de taak van de unitbesturing de externe waarnemingen op deze manier te distribueren over zijn interne nodes.

Op elke unit of iedere combinatie van units kunnen waarnemingen worden gepleegd. Vaak zullen waarnemingen op het laagste niveau (daar wordt het niveau van units mee bedoeld die niet nog eens childunits bevatten) worden gedaan, maar dat is niet noodzakelijk. Stel dat we drie lagen units hebben met op het laagste niveau 'symptomen', op het middelste niveau 'ziektes' en op het hoogste niveau 'behandelingsmethoden'. Het ligt dan voor de hand dat de waarneming op het niveau van de symptomen wordt gedaan om vervolgens te leiden tot conclusies omtrent ziektes en uiteindelijk behandelingsmethoden. Toch kan het handig zijn om uitgaande van een ziekte de bijbehorende symptomen te weten, of zelfs uitgaande van de behandelingsmethode de ziekte terug te zoeken. Ook zou een combinatie van waarnemingen over verschillende niveaus gewenst kunnen zijn.

Binnen SCORE zijn alle units, technisch gezien, hetzelfde. Verder worden alle units tijdens het redenatieproces gelijkwaardig behandeld. Een waarneming kan daardoor op iedere unit of iedere combinatie van units worden ingeprikt.

Zoals reeds eerder vermeld vindt de waarneming niet rechtstreeks plaats op de unit zelf, maar gebeurt dit op een aparte unit: de perceptorunit. Deze methode van waarnemen wordt 'clampen' genoemd. Als bijvoorbeeld instantie $U_3[0]$ wordt waargenomen adviseert de perceptorunit U_3' de unit U_3 tot het activeren van de interne nodes U_3-i[1] en U_3-i[3]. Tijdens het inferentieproces oefent niet alleen U_3' invloed uit op U_3, maar ook de childunits U_1, U_2 en U_H en de parentunit U_5. In de keuze welke node uiteindelijk wordt geactiveerd, worden de invloeden van parents en childs even zwaar geteld. Via een parameter β kan de kracht van de perceptorunit U' op de unit U worden geregeld. Als β=1, dan heeft de perceptor evenveel invloed op de unit als de parent- en childunits tezamen. Als β<1, dan hebben parent- en childunits meer invloed op de unit dan de perceptor en als β>1 dan heeft de perceptor meer invloed. In de vorm van een formule ziet I(α), want zo wordt de uiteindelijke input voor node α verder genoemd, er uit als in formule 5.3.a.i. P_{par} staat voor 'Present parent(s)', ofwel 'er zijn parents aanwezig', die de waarde nul, als er geen parents aanwezig zijn, of de waarde één, als ze er wel zijn, kan aannemen. Met gelijksoortige betekenis is er ook een P_{chi} (Present child(s)) en een P_{per} (Present perceptor). D.m.v. de term $(1/(P_{par} + P_{chi})) * [P_{par}*r_{1,\alpha} + P_{chi}*r_{2,\alpha}]$ worden parent- en child invloeden ($r_{1,\alpha}$ respectievelijk $r_{2,\alpha}$), voor zover aanwezig, gemiddeld. Deze term wordt vervolgens weer gemiddeld met de eventuele perceptorinvloed ($\beta*r_{3,\alpha}$). In deze formule wordt er vanuit gegaan dat er voor iedere unit óf een parent- en/óf een childunit aanwezig is. Een losse unit, met eventueel alleen een perceptorunit, heeft geen praktische betekenis.

$$I(\alpha) = \{1/(1+\beta*P_{per})\} * \{ (1/(P_{par}+P_{chi})) *[P_{par}*r_{1,\alpha} + P_{chi}*r_{2,\alpha}] + \beta*P_{per}*r_{3,\alpha} \}$$

(5.3.a.i)

Deze formule wordt gebruikt voor de berekening van de I(α)'s van alle units, ook voor de units waar geen waarneming op wordt gepleegd. P_{per} is dan nul en I(α) wordt daarmee gedefinieerd als pure middeling van parent- en childinvloeden. Door deze vorm zal altijd gelden dat

$\sum_\alpha I(\alpha) = 1$, waardoor I(α) ook als een kansmaat kan worden gezien.

Voor verschillende situaties gelden verschillende argumenten voor de keuze van β. In het hiernavolgende zal worden uitgelegd hoe β in SCORE is toegepast. Ten eerste is het in SCORE-S zo dat β kan worden ingesteld. Als 3 op bijvoorbeeld 2 wordt ingesteld, zal bij iedere unit waarop een waarneming is gepleegd, deze waarneming twee maal zo zwaar worden meegerekend als de invloeden van de parent- en childunits. Dit oogt nogal willekeurig. Voor ieder type probleem zouden we moeten uitzoeken wat de beste keuze voor β is. Stel dat in figuur 5. 2. i enkel U1. I0j wordt waargenomen. Dan ligt het voor de hand 3 zeer groot te kiezen om er voor te zorgen dat binnen unit U1, tijdens het redenatieproces voortdurend U1. I0J wordt geactiveerd waardoor de invloed van de waarneming goed door het netwerk kan doordringen. Als we aan de andere kant de waarneming (U3-e [1] , U4-e [1]) doen (beschouw nogmaals waarheidstabel 5. 2. e), dan zou een hoge 6 voor problemen zorgen. Bij U3-e [1] horen namelijk twee interne nodes die dus beiden even sterk door de waarneming worden aangestuurd, namelijk U3-i I0J en U3-i[2 J. Vanuit de buitenwereld, in dit geval parentunit U5, moet aan U3 duidelijk gemaakt kunnen worden dat U4-i[1j ook is waargenomen, en dus voor de bijbehorende U3-is 2) en niet voor U3-i[0j moet worden gekozen. Als β te groot is kan dit gegeven niet meer tot U3 doordringen.

Een andere situatie waarbij een hoge β weinig zin heeft is in het geval van een onzekere waarneming, bijvoorbeeld in de in het begin van deze paragraaf geschetste situatie van $U_1[0]=0.75$ en $U_1[1]=0.25$. Als β hoog is, zal altijd worden gekozen voor $U_i[0]$ omdat de unitbesturing uiteindelijk dié node activeert die het hoogste advies I(α) krijgt (hierover later meer) en de hoogste in dit geval altijd $U_1[0]$ zal zijn. Het onzekere in de waarneming zal daarom bij hoge β automatisch als een zekere waarneming worden geïnterpreteerd. Daarom moeten we er voor zorgen dat bij een onzekere waarneming ook childs en parents adviezen kunnen uitbrengen aan de unit om uiteindelijk eventueel $U_1[1]$ te activeren, ook al heeft deze waarneming niet de hoogste kans.

In SCORE-S is daarom de volgende optie ingebouwd. Deze is gebaseerd op de constatering dat wanneer het een zékere waarneming op de interne nodes betreft, β hoog moet zijn en wanneer het een onzekere waarneming op de interne nodes betreft, β gelijk aan één moet zijn. Een zekere waarneming U_4-e[1] wordt hiermee, zonder invloed van buiten, hard op U_4-i[1] geclampt. Een zekere waarneming U_3-e[1], resulterend in een onzekere interne waarneming U_3-i[0]=U_3-i[2]=0.5, óf rechtstreeks een onzekere externe waarneming, worden niet hard geclampt. β wordt in dat geval automatisch één, waardoor invloeden van buiten even zwaar worden geteld als de invloed van de perceptor.

Wellicht ten overvloede, maar met het oog op de volledigheid van deze paragraaf over waarnemen worden de drie typen zwakke waarnemingen op rij gezet: onvolledige, onzekere en inconsistente waarnemingen.

Beschouw nogmaals figuur 5.2.i. We hadden daar 5 units. Op elke unit kan een waarneming worden gepleegd. Binnen elke legale afleiding, die allen worden gedefinieerd door de trainingexamples, heeft élke unit een waarde. Een volledige waarneming, d.w.z. een waarneming over alle 5 units tegelijkertijd, is in dit verband niet interessant. Het antwoord is dan door de gebruiker al gegeven en er valt niets meer te redeneren, behalve als het een inconsistente waarneming betrof, waardoor een deel van de waarneming aangepast zou moeten worden. Op het moment dat op minder dan 5 units een waarneming wordt gepleegd, hebben we te maken met een onvolledige waarneming. Het is de taak van het systeem deze waarneming te completeren door de waarden van de overige units erbij te vinden. In de literatuur wordt dit wel de 'completion task' genoemd. Een voorbeeld van een onvolledige waarneming is: (U_1[0], U_2[0], U_4[0]) . Als we kijken naar waarheidstabel 5.2.e moet deze waarneming door het inferentieproces intern worden gecompleteerd met (U_3-i[0], U_5-i[0]) en dus extern met (U_3-e[1], U_5-e [0]). Als slechts de onvolledige waarneming U_5[0] wordt gedaan zijn er meerdere legale afleidingen mogelijk. Welke uiteindelijk moet worden gevonden hangt af van de bekendheid, gedefinieerd in de verbindingen, van een patroon.

Onzekere waarnemingen hebben we ook inmiddels uitgebreid besproken. Een voorbeeld ervan is: P(U_1[0]=0.6), P(U_1[1]=0.4). Via de methode van clampen met β=1 kan deze informatie in het systeem worden gebracht.

Een inconsistente waarneming is een waarneming die niet in de trainingsvoorbeelden van het systeem voorkomt. Bijvoorbeeld de combinatie: (U_1[0], U_2[1]). U_1[0] is een onderdeel van U_3-i[0] en U_2[1] is een onderdeel van U_3-i[2] en U_3-i[3]. Nu willen we graag dat de waarneming hersteld wordt tot (U_1[1], U_2[1]) omdat U_3-i[2] bekender voor het systeem is dan U_3-i[0] en U_3-i[3]. Voor het afhandelen van inconsistente waarnemingen is de volgende oplossing bedacht. Een waarneming (U_1[0], U_2[1]) kan niet resulteren in een legale afleiding. Het systeem zal dan blijven hangen in een zo redelijk mogelijk alternatief. Als β<1, dus als child- en parentunits meer invloed op de unit hebben dan de perceptor dat heeft, kunnen de waarnemingen in principe worden gecorrigeerd. Dan kan bijvoorbeeld toch (U_1[1], U_2[1], U_3-i[2], U_4[0], U_5-i[2]) worden geconcludeerd. U_1 is in dat geval gecorrigeerd. In de praktijk blijkt het niet verstandig de β te laag te maken, omdat het belangrijk is dat de waarneming het netwerk doordringt. De waarneming is tenslotte datgene waar aan de hand van geredeneerd moet gaan worden.

Nu voorziet SCORE-S in de mogelijkheid tot het, onmiddellijk volgend op het redenatieproces, uitvoeren van extra afleidingsstappen zónder waarneming. Dit gebeurt dan totdat er een stabiele situatie is ontstaan. Het effect hiervan is dat tijdens deze toegevoegde fase het vinden van een legale afleiding wordt opengebroken. Omdat er geen storing van een

inconsistente waarneming is, zal het netwerk vervallen naar een dichtbij liggende (en hiermee wordt gedoeld op het 'energielandschap') oplossing. Omdat, in het inconsistente geval van hiervoor, U_3-i[1] geldt, zal U_1[1] worden gecorrigeerd tot U_1[0] en wordt de legale oplossing bereikt. Later zal hier uitgebreider op worden teruggekomen.

5.3.2 Inferentie

5.3.2.1 Inleiding

Beschouw figuur 5.3.a. De nodes aangegeven door het teken o zijn de actieve nodes van de units. De nodes met het teken • zijn de passieve.

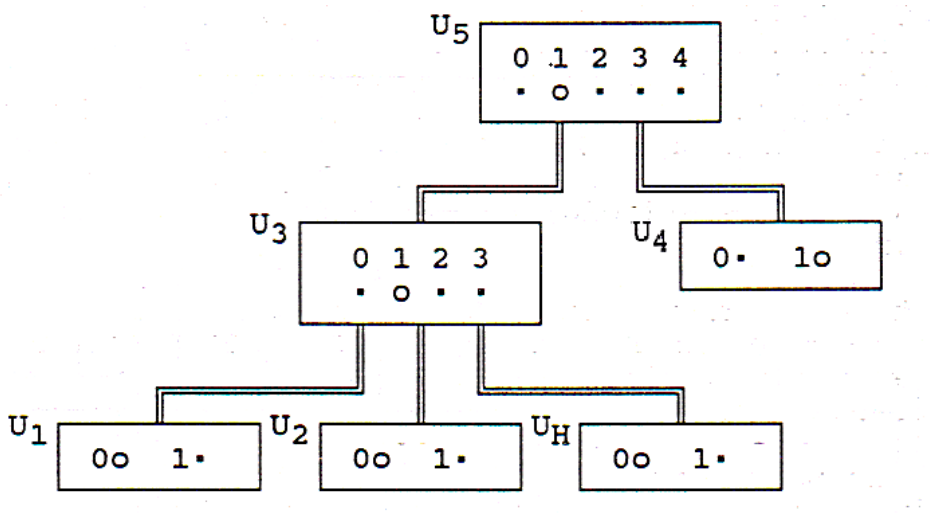


Fig. 5.3.a. Een netwerkinstelling corresponderend met een illegale afleiding

Hiervoor is gesproken over een unitbesturing die een niet-lineaire functie toepast op de I(α)'s van de nodes uit de betreffende unit. Stel nu dat we dat algoritme wegdenken en dat de unitbesturing enkel de node met de hoogste I(α) activeert en de overige deactiveert. De bovenstaande situatie van figuur 5.3.a. zou dan een stabiele zijn die echter niet in één van de trainingsvoorbeelden voorkomt. Dit noemen we een lokaal optimum. De situaties als in de trainingsvoorbeelden zijn de globale optima. Dat bovenstaande wél een stabiele situatie is, is als volgt in te zien. U_3[1] en U_4[1] concluderen samen U_5[1].

(Eigenlijk wordt U_3-i[1], U_4-i[1] en U_5-i[1] bedoeld, maar omdat het interne nu uit de context duidelijk is wordt die specificatie achterwegen gelaten). Hier zit niets vreemds: dit is overeenkomstig de node definities zoals we die hebben opgesteld in tabel 5.2.e. (U_1[0], U_2[0], U_H[0]) concluderen verder U_3[1]. Dit klopt echter niet: (U_1[0], U_2[0], U_H[0]) zou U_3[0] moeten concluderen. Wat is de informatie die U_3 binnen krijgt van U_1, U_2, U_H en U_5? Om dit te

onderzoeken hoeven we alleen de informatie van $U_1[0]$, $U_2[0]$, $U_H[0]$ en $U_5[1]$ te bekijken omdat niet-actieve nodes van U_1, U_2, U_H en U_5 geen invloed hebben op de berekening van U_3.

In figuur 5.3.b zijn alle van belang zijnde verbindingen getekend. De verbindingen zijn niet rechtstreeks van node tot node getekend. Vanuit de nodes $U_1[0]$, $U_2[0]$, $U_H[o]$ en $U_5[1]$ moeten de verbindingen naar de nodes $U_3[0]$, $U_3[1]$, $U_3[2]$ en $U_3[3]$ worden gedacht.

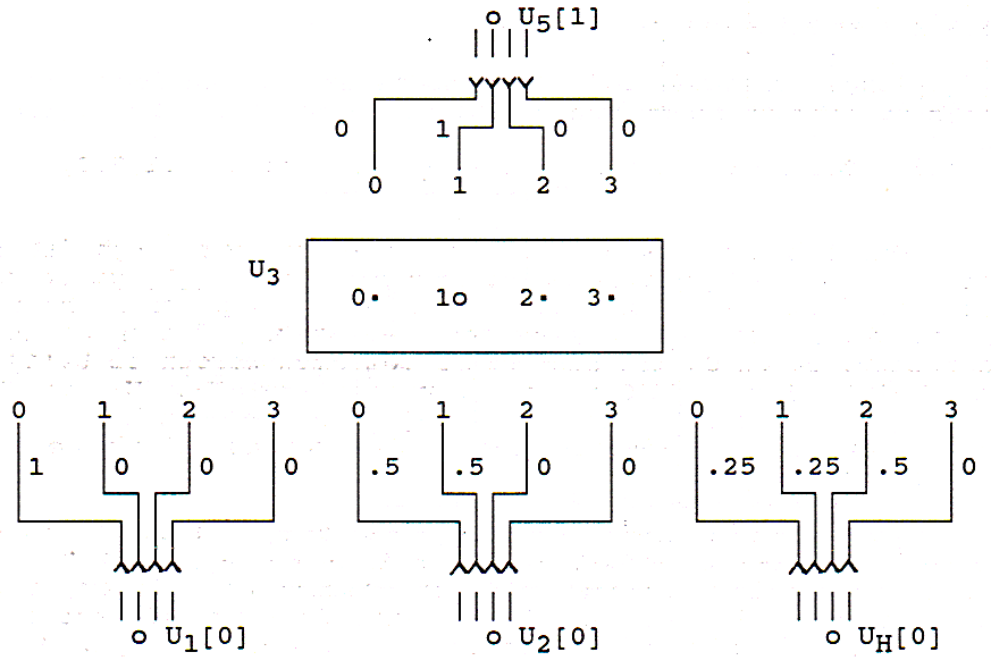


Fig. 5.3.b Unit U_3 uit figuur 5.3.a met zijn binnenkomende verbindingen in detail

Hierna volgt tabel 5.3.a waarin de berekening van $I(U_3[\alpha])$ is weergegeven. Dit geschiedt d.m.v. de formules zoals die beschreven zijn in subparagraaf 5.2.1.

α	0	1	2	3
$i(U_1,\alpha)$	1	0	0	0
$i(U_2,\alpha)$	.5	.5	0	0
$i(U_H,\alpha)$	.25	.25	.5	0
gem(α)	.58	.25	.17	0
prod(α)	.13	0	0	0
$r_{2,\alpha}$	.83	.1	.07	0
$i(U_{5,\alpha})$	0	1	0	0
$r_{1,\alpha}$	0	1	0	0
$I(U_3[\alpha])$	.42	.55	.03	0

Tabel 5.3.a De berekening van $I(U_3[\alpha])$ voor $\alpha \in \{0,1,2,3\}$

Omdat het unitbesturingsalgoritme er voor zorgt dat de node met de hoogste I(α) wordt geactiveerd, wordt $U_3[1]$ geactiveerd en de overige gedeactiveerd. Deze wordt op zijn beurt weer doorgegeven aan U_1, U_2, U_H en U_5. Omdat, laten we dat even aannemen, U_1 en U_2 met een hoge β geclampt worden op $U_1[0]$, $U_2[0]$ en $U_H[0]$, kan $U_3[1]$ de activeringen van U_1, U_2 en U_H niet corrigeren. Verder houdt $U_3[1]$, U_5 [1] in stand en het hele zaakje is keurig in evenwicht. Toch hebben we duidelijk geen legale oplossing te pakken.

In energietermen gesproken, is een laag energieniveau bereikt, alleen niet het laagste. In subparagraaf 2.2.4 is daar in abstracte bewoordingen al over geschreven. Aldaar is energie gedefinieerd als een maat in hoeverre de verschillende neuronen, in SCORE zijn dat nodes, het met elkaar 'eens' zijn. De laagste energie heeft dat patroon van nodeinstellingen waar de nodes met zo groot mogelijke zekerheid worden geactiveerd. Dit wordt weer aan de hand van voorgaand voorbeeld toegelicht.

In de nodeinstelling van figuur 5.3.a liggen, zoals U hiervoor heeft gezien, de I(α)'s van de nodes van U_3 erg dicht bij elkaar:

$I(U_3[0]) = 0.42$
$I(U_3[1]) = 0.55$
$I(U_3[2]) = 0.03$
$I(U_3[3]) = 0$

In de volgende figuur wordt de situatie geponeerd waar een grotere spreiding is in de I(α)'s. De figuur representeert een legale afleiding, d.w.z. een globaal energieminimum.

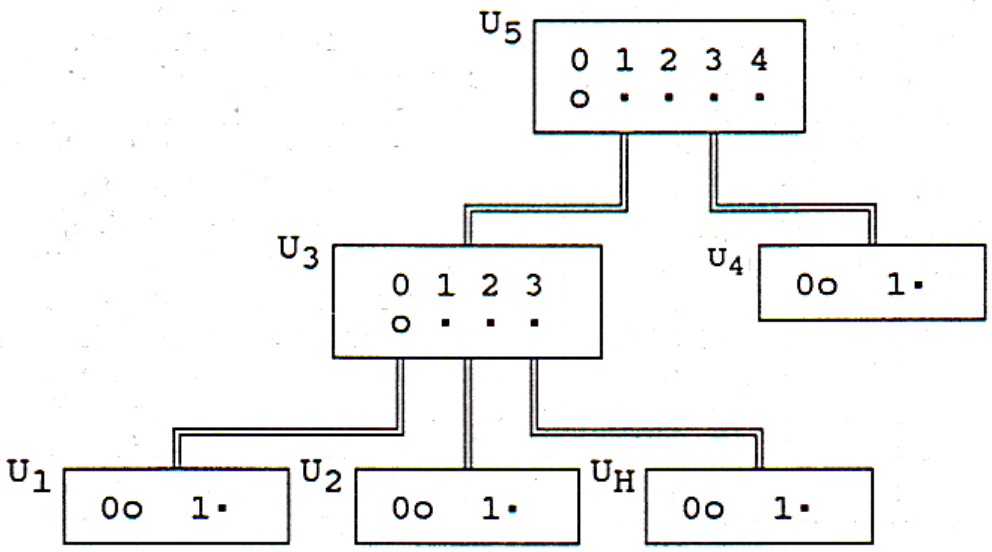


Fig. 5.3.c Een netwerkinstelling corresponderend met een legale afleiding.

T.o.v. figuur 5.3.a zijn de waarden van de units U_3, U_4 en U_5 veranderd. Als we nu opnieuw de I(α)'s van U_3 berekenen krijgen we het volgende resultaat:

$I(U_3[0]) = 0.92$
$I(U_3[1]) = 0.05$
$I(U_3[2]) = 0.03$
$I(U_3[3]) = 0$

Dit resultaat is op dezelfde manier ontstaan als in tabel 5.3.a. De $i(U_5,\alpha)$, dus de $r_{1,\alpha}$, is nu anders omdat de parentunit van U_3, dus unit U_5, een andere instelling heeft gekregen. $r_{2,\alpha}$ is hetzelfde gebleven omdat de instellingen van de childunits van U_3 niet zijn veranderd. Het laatste stukje van die berekening is in tabel 5.3.b weergegeven.

α	0	1	2	3
$r_{2,\alpha}$	.83	.1	.07	0
$i(U_{5,\alpha})$	1	0	0	0
$r_{1,\alpha}$	1	0	0	0
$I(U_3[\alpha])$	.92	.05	.03	0

Tabel 5.3.b De berekening van $I(U_3[\alpha])$ voor $\alpha \in \{0,1,2,3\}$

Duidelijk is te zien dat de spreiding nu veel hoger is geworden doordat U_1, U_2, U_H en U_5 allen hetzelfde advies aan unit U_3 geven.

Smolensky noemt de situatie van figuur 5.3.c een harmonieuze en de situatie van figuur 5.3.a een minder harmonieuze. Energie is de reciproke van harmonie. De bijbehorende 'Harmonie Theorie', besproken in subparagraaf 2.4.4 is niet in al zijn facetten toepasbaar op SCORE. Dit heeft te maken met het feit dat Smolensky in principe slechts twee niveaus (features en atoms) toestaat en dit aantal in SCORE in principe onbeperkt is. Een ander verschil is het feit dat in het model van Smolensky een verbinding van node a naar node b, dezelfde is als die van b naar a. Het kenmerkende van SCORE daarentegen is juist het feit dat ze verschillend zijn. Een derde verschil is het denken in termen van units in SCORE. Binnen de Harmonie Theorie zijn de nodes niet logisch gegroepeerd: hun handelen is volledig autonoom. De wiskundige grondslagen van de Harmonie Theorie zijn dan ook niet gebruikt als wiskundige basis voor SCORE. De Harmonie Theorie heeft echter wél als inspiratiebron gefungeerd m.b.t. tot de redeneerprincipes die in de Harmonie Theorie worden gebruikt. Door de verschillen tussen de Harmonie Theorie en SCORE, wordt in het vervolg dus niet meer de term Harmonie maar enkel nog de term energie gebruikt.

In het hiernavolgende zal het begrip energie voor een unit worden gedefinieerd. Er zal naar een methode worden toegewerkt, waarmee kan worden aangetoond dat toepassing van deze methode er voor zorgt dat in iedere unit de energie zal worden geminimaliseerd. Het begrip energie moet dan natuurlijk zo worden gedefinieerd dat minimalisatie van die energie ook precies oplevert wat we willen. Concreet betekent dit dat unit U_3 uit figuur 5.3.c volgens die definitie een lagere energie oplevert dan het netwerk van figuur 5.3.a.

Wat maakt dat U_3 uit figuur 5.3.a een betere instelling heeft dan U_3 uit figuur 5.3.c? Dit is eigenlijk heel eenvoudig. We willen natuurlijk dat voor de geactiveerde node α, de bijbehorende $I(\alpha)$ zo hoog mogelijk is en daarmee de $I(\tau)$'s ($\tau <> \alpha$) van de overige nodes zo

klein mogelijk. De energie van een unit, aangeduid door E_U, kan daarmee worden gedefinieerd als de reciproke van I(α), waarbij α de geactiveerde node van unit U voorstelt. Zie formule 5.3.a.ii.

$$E_U = 1 / (\Sigma_\alpha o(U[\alpha]) * I(\alpha))$$

(5.3.a.ii)

Doordat o(U[α]) slechts 1 is voor één node α blijft uiteindelijk voor E_U alleen de reciproke van I(α) over. In figuur 5.3.a is E_{U3} gelijk aan 1/0.55=l.82. In figuur 5.3.c is E_{U3} gelijk aan 1/0.92=1.09. In het laatste geval is de energie dus inderdaad lager dan in het eerste. Nu hebben we een wiskundig minimaliseringscriterium dat we op een intrinsiek parallelle wijze willen oplossen. Een 'intrinsiek parallelle wijze' verdient enige toelichting. Het systeem bestaat uit afzonderlijke units die slechts informatie krijgen uit naburige units. Er is niet iets dat als het ware boven het systeem hangt dat het energieniveau in de gaten kan houden. Terwijl de energieniveaus van alle units afzonderlijk vollédig van elkaar afhankelijk zijn, moet iedere unit voor zich streven naar een zo laag mogelijke energie.

In principe is, als we N units hebben, een probleem op te vatten als N+1 dimensionale energieruimte. Op elke dimensie worden de mogelijke waarden van de unit vastgelegd. Hierdoor hebben we N dimensies. Op de (N+1)-ste dimensie wordt de totale netwerk energie E uitgezet, welke een functie is van de afzonderlijke unitenergieën E_U. Hierdoor ontstaat er al bij een klein netwerk een onvoorstelbare ruimte die via op die 'intrinsiek parallelle wijze', doorzocht moet worden op het diepste energiedal. In dat dal representeren de bijbehorende unitwaarden de oplossing van het probleem.

Nu is het in het geval van SCORE niet echt duidelijk hoe deze totale energie moet worden gedefinieerd. Doordat de relatie tussen de afzonderlijk unitenergieën en de totale energie onduidelijk is, kunnen we niet op voorhand bewijzen met afzonderlijke unitenergieën, dus met welke specifieke oplossing, het diepste energiedal correspondeert. In ieder geval is duidelijk dat als we iedere unit in een situatie kunnen brengen waarbij de afzonderlijk unitenergieën zeer laag zijn, we een goede oplossing te pakken hebben. We mogen aannemen, en simulatie met SCORE-S heeft het geloof in deze aanname versterkt, dat het diepste energiedal niet zo snel zal liggen op een punt dat in tegenspraak is met de waarneming. Dit probleem zou kunnen gaan optreden bij grote netwerken met weinig waarnemingen.

Doordat de relatie tussen totale- en afzonderlijk unitenergie niet helemaal duidelijk is, is ook onduidelijk of aan alle units in het netwerk bijvoorbeeld evenveel belang wordt gehecht. Het zou bijvoorbeeld goed mogelijk kunnen zijn dat een waarneming op een unit op een hoog niveau meer invloed op de netwerkinstelling levert dan een gelijktijdige waarneming op een lager niveau. In het geval van een inconsistente waarneming op deze twee units zal in dat

geval de voorkeur worden gegeven aan een aanpassing aan de waarneming op de unit op het hoge niveau. Indien het wel een consistente waarneming betreft en meerdere oplossingen mogelijk zijn die allen even bekend zijn (d.w.z. even vaak in de vorm van een trainingsvoorbeeld aan het netwerk zijn aangeboden), is het ook niet op voorhand duidelijk welke oplossing correspondeert met het diepste dal. Dit betekent dat wanneer trainingsvoorbeelden even vaak aan het netwerk zijn aangeboden, toch voorkeur kan bestaan voor één van de twee, puur op basis van de structuur van het netwerk. Dit is echter geen praktisch probleem omdat het netwerk vrij getraind kan worden.

Als we vinden dat het ene trainingsvoorbeeld wat vaker teruggevonden moet worden, bieden we dat trainingsvoorbeeld simpelweg één of meer keer extra aan. Op dit punt verlaten we echter wel de wiskundige onderbouwing, en gaan we een meer intuïtieve benadering tegemoet. Dit is kenmerkend voor Connectionist Systems. We accepteren het bestaan van een energielandschap en gaan op zoek naar een methode die het diepste dal voor ons kan vinden.

Het blijkt dus zo te zijn dat als we de units rechtstreeks energiedaling laten nastreven, de kans klein is dat het globale energieminimum wordt bereikt. In de praktijk belanden we al snel in een lokaal energieminimum, zoals in figuur 5.3.a. Om aan dat probleem te ontsnappen hebben we nog een gedeeltelijk onbesproken hulpmiddel voor handen: de unitbesturing. In de volgende subparagraaf wordt het niet-lineaire algoritme van de unitbesturing gedefinieerd, Annealing genoemd, om er zo voor te zorgen dat toch dat globale energieminimum corresponderend met een legale, d.w.z. (voor zover mogelijk) consistente en volledige, afleiding kan worden gevonden.

5.3.2.2 Simulated Annealing

In de vorige subparagraaf is beschreven dat ons redeneermechanisme zo moet worden ingericht dat, op grond van lokale beslissingen, toch een, globaal gezien, zo optimaal mogelijke oplossing gevonden kan worden. Concreet houdt dit in dat we een unitbesturingsalgoritme moeten bedenken waarmee de netwerkenergie, waar in de voorgaande subparagraaf over is gesproken, kunnen minimaliseren. Het probleem van het bereiken van een lokaal i.p.v. een globaal energieminimum is grafisch in figuur 5.3.d weergegeven.

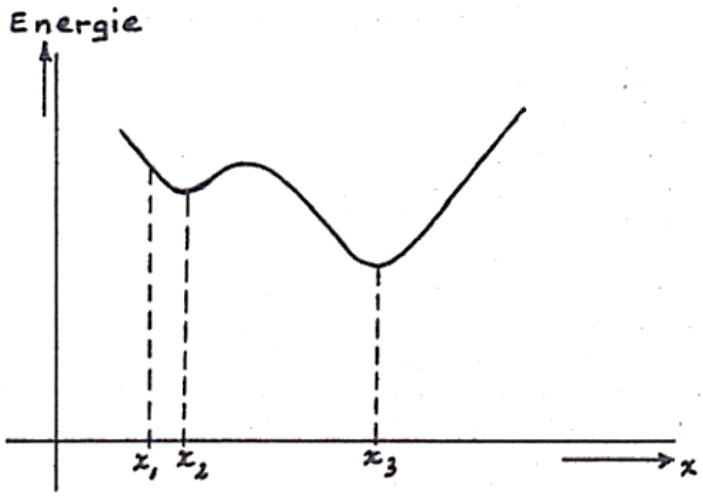


Fig. 5.3.d. Het energielandschap van de variabele x.

Indien x bij aanvang de waarde x_1 heeft zal, als de unitbesturing rechtlijnig zijn energieminimaliserings-strategie uitvoert, de waarde x_2 bereiken. Omdat zowel onmiddellijk links als rechts van x_2 de energie hoger is dan de energie in x_2 zelf zal dit ook meteen het eindpunt zijn van het zoekproces. Metaforisch gesproken is dit voor te stellen als een balletje dat we op x_1 neerleggen en onmiddellijk naar x_2 toe rolt. Toch moet niet x_2 maar x_3 worden gevonden. Er kan een nog concreter voorbeeld worden gegeven over het probleem van lokale en globale energieminima.

Gegeven is een raster van nodes, die allemaal de waarde +1 en -1 kunnen aannemen, zie figuur 5.3.e. Iedere node, aangegeven door een '*' , is slechts met z'n nabuur verbonden. Slechts één patroon van -1-en en +l-en is de beste oplossing en representeert het globale energieminimum. Welke dat is ligt impliciet opgeslagen in de verbindingen tussen die nodes.

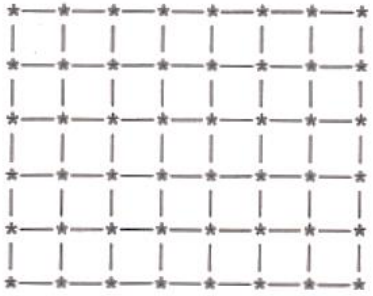


Fig. 5.3.e Noderaster waarbij iedere node met zijn nabuur is verbonden

Initieel heeft iedere node een random waarde. Afhankelijk van de sterktes van de verbinding tussen de nodes en de waarde van hun buren zal iedere node een nieuwe beslissing maken omtrent zijn nieuwe waarde. Er ontstaat een iteratieproces dat uiteindelijk zal eindigen als de nodes niet meer van waarde veranderen. We hopen dan een globaal energieminimum gevonden te hebben. Dat we echter ook gemakkelijk in een lokaal energieminimum kunnen uitkomen is als volgt in te zien. Kriskras door het rooster worden beslissingen genomen. De volgende slag gebeurt dit opnieuw, gebruik makend van de waarden van hun buren. Omdat deze beslissingen slechts op grond van lokaal aanwezige informatie wordt genomen is het duidelijk dat die eerste beslissingen hoogstwaarschijnlijk niet overeenstemmen met de globale oplossing. De hoop is dat deze foute beslissingen uiteindelijk gecorrigeerd zullen worden doordat de informatie van elders uit het rooster zal doordringen. Dit hoeft niet het geval te zijn indien er clusters van nodes ontstaan die elkaars beslissingen in stand houden. Binnen zo'n cluster is de oplossing optimaal, maar over het hele rooster bezien niet. Die clusters kunnen zo sterk zijn dat die clusters onderling geen invloed meer op elkaar kunnen uitoefenen. We zitten nu in een lokaal energieminimum.

Bovenstaand probleem lijkt veel op het natuurkundige probleem van het behandelen van scheuren in kristalroosters. Een methode hiervoor is: 'Annealing'. Annealing, letterlijk vertaald als 'temperen', is een proces waarbij het materiaal verhit en vervolgens zeer langzaam afgekoeld wordt. Het idee erachter is het feit dat wanneer het materiaal verhit wordt, de krachten tussen de atomen zwakker worden en ze zich dus relatief vrij kunnen heroriënteren. Ze bevinden zich dan in een toestand van hoge energie. Het materiaal wordt vervolgens langzaam afgekoeld en wel zó langzaam dat het de tijd heeft de effecten van clustertjes atomen door het hele materiaal te verspreiden. Uiteindelijk, bij T=0, is het materiaal bevroren en bevindt het zich in een globaal energieminimum: de scheuren zijn verwijderd.

Ditzelfde principe is ook gebruikt binnen de Harmony Theorie (beschreven in subparagraaf 2.4.4 en uitgebreider behandeld in bijlage I) en is, in aangepaste vorm, ook verwerkt binnen SCORE. De computertoepassing van Annealing wordt 'Simulated Annealing' genoemd. Simulated Annealing toegepast op figuur 5.3.d, houdt in dat we de besturing zó moeten programmeren dat het balletje dat in x_1 wordt neergelegd niet rechtstreeks naar x_2 rolt, maar dat we het ook tegen de helling in kunnen laten rollen. Bij hoge T mag de vorm van het energielandschap geen invloed hebben op de beweging van het balletje. Als we T laten dalen moeten we er voor zorgen dat het landschap meer grip krijgt op de beweging ervan. Dat dit een methode is om het minimum te vinden kan als volgt worden ingezien.

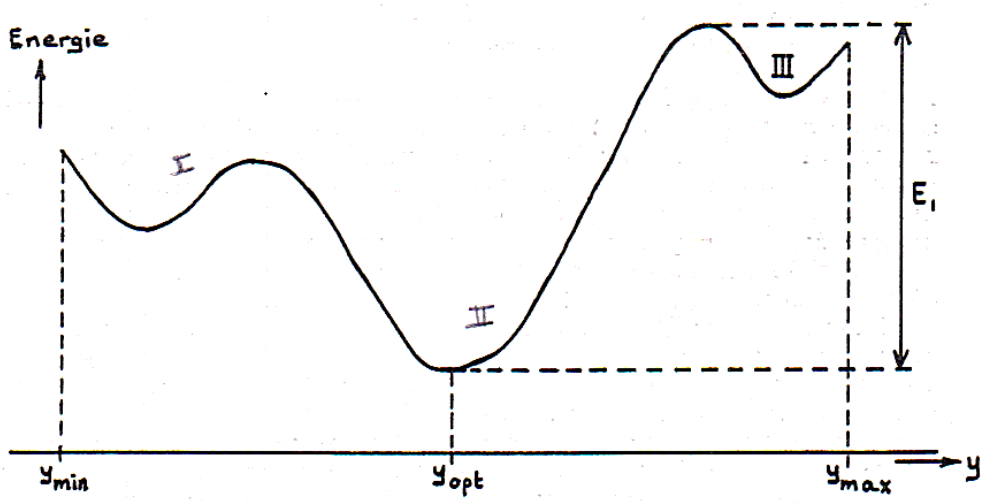


Fig. 5.3.f Het energielandschap van de variabele y.

Beschouw figuur 5.3.f. Als T hoog is, is de beweging van y willekeurig door het energielandschap. Anders gezegd, een hoge T veroorzaakt een trillende y. We beginnen met een T die groot genoeg is om E_1 gemakkelijk te overbruggen. De truc is dat het systeem voldoende de tijd moet krijgen om y in alle drie de dalen (I, II en III) te laten komen! We verlagen T nu langzaam. Op een bepaald ogenblik zal T te laag zijn om de rechter helling van dal II naar dal III over te trillen. Slechts dal I en II kunnen dan nog worden bereikt. T moet weer voldoende langzaam dalen opdat de kans groot is dat beide dalen I en II ook werkelijk aangedaan worden. Bij een nog wat lagere T zal de linkerzijde van helling II ook niet beklommen kunnen worden. Tot de verlaging op T=0 zal y nog wat na blijven sputteren in dal II om uiteindelijk te bevriezen in y_{opt}. In theorie is het zo dat als T oneindig langzaam daalt, het optimum met kans één wordt gevonden. In de praktijk hoeft dat niet zó langzaam te geschieden om toch met vrij grote waarschijnlijkheid in het, of een aanvaardbaar, minimum, te geraken.

Dit principe is als volgt binnen SCORE geïmplementeerd. Beschouw figuur 5.3.g voor een totaaloverzicht van de unitbesturing.

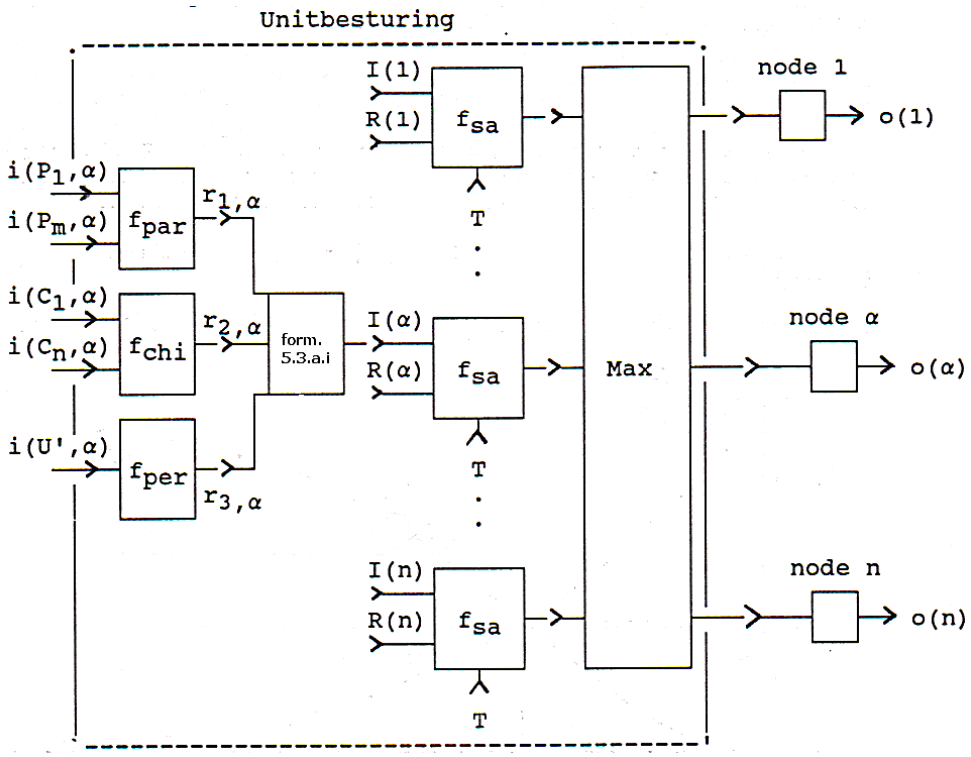


Fig. 5.3.g. De unitarchitectuur

Het grootste deel van de unitarchitectuur is in voorgaande reeds besproken. De i(V,α)'s afkomstig van parentunits, childunits en perceptorunit worden gescheiden behandeld, resulterend in $r_{1,\alpha}$, $r2_{,\alpha}$, $r_{3,\alpha}$ voor iedere node α. De functies die dit bewerkstelligen zijn behandeld in subparagraaf 5.2.1 en hebben toen respectievelijk de namen f_{par}, f_{chi}, en f_{per} gekregen. Vervolgens worden $r_{1,\alpha}$, $r2_{,\alpha}$, $r_{3,\alpha}$ gewogen volgens formule 5.3.a.i. om te resulteren in I(α). Deze omzetting van i(V,α) naar I(α) is in figuur 5.3.g, i.v.m. het behoud van het overzicht, alleen weergegeven voor node α. Dit proces vindt natuurlijk voor alle nodes 1≤α≤n plaats.

Indien deze signalen I(α) rechtstreeks aangeboden zouden worden aan het blok met de functie Max zouden we de situatie hebben waarbij we rechtstreeks naar het dichtstbijzijnde lokale minimum vallen. M bepaalt namelijk het maximum van zijn inputwaarden en activeert dié node α die de grootste input leverde en deactiveert de overige. Nu zitten er echter functies f_{sa} tussen, die allen ook nog worden gevoed met een R(α) en een T. f_{sa} staat voor de functie die de Simulated Annealing moet bewerkstelligen. R(α) is een random kans voor node α. Daarom geldt ook:

$$\Sigma_\alpha R(\alpha) = 1$$

De temperatuur T is een maat voor in hoeverre óf I(α) óf R(α), als resultaat worden doorgestuurd naar het blok Max. De functie die dit afhandelt is weergegeven in formule 5.3.b.

$$f_{sa}(R(\alpha),I(\alpha),T) = R(\alpha) * EXP(-1/T) + I(\alpha) * EXP(-T)$$

(5.3. b)

Ter verduidelijking van deze formule is figuur 5. 3. h opgenomen.

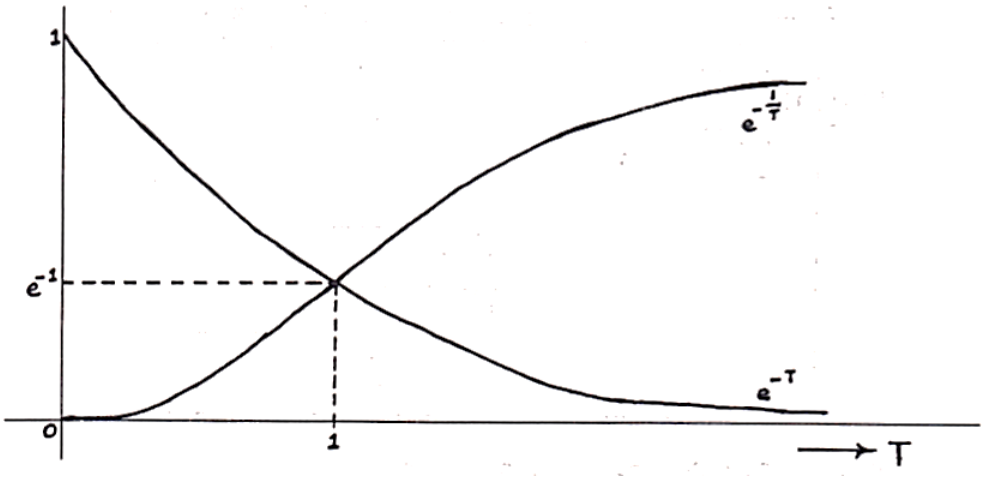


Fig. 5.3.h De invloeden van R(α) en I(α) op de functie f_{sa} als functie van T.

De temperatuur laten we lopen vanaf een waarde waarbij EXP(-T) verwaarloosbaar is t.o.v. EXP(-1/T), tot T=0. Bij aanvang moet iedere node α van de unit met ongeveer gelijke kans de hoogste f_{sa} kunnen krijgen, dus geactiveerd kunnen worden. T=2 blijkt in de praktijk van SCORE-S voldoende (EXP(-1/2) =0.6; EXP(-2)=0.13) om, mits we voor het afkoelingsproces voldoende stappen nemen, de belangrijkste nodes van een unit de kans te geven geactiveerd te worden. Op wat 'voldoende stappen' is, wordt later terug gekomen. Bij T=2 vindt de activering van de nodes uit de unit dus redelijk random plaats.

Langzaam laten we de temperatuur afnemen volgens een negatief exponentiële functie van het aantal stappen (n). Zie hiervoor figuur 5.3.i.

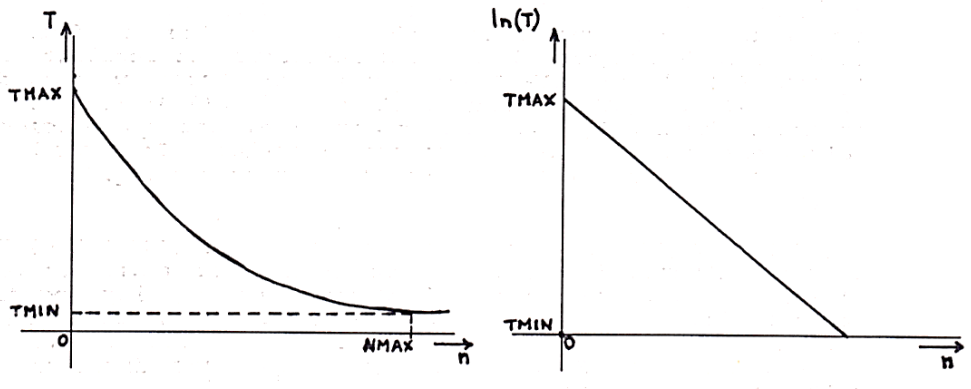


Fig. 5.3.i Negatief exponentiële temperatuurregeling.

In totaal NMax stappen laten we de temperatuur aflopen van TMax naderend naar TMIN=1/TMAX. Binnen SCORE-S is hiervoor de formule 5.3.c geïmplementeerd. Het bevat behalve de variabelen TMax, het startpunt van de afkoeling, en n, het stapnummer, ook de variabele s. D.m.v. deze variabele kan worden gespecificeerd hoeveel stappen de temperatuur constant moet blijven om vervolgens weer een nieuwe waarde aan te nemen. Verder zien we de variabele NMax, waarmee het totaal aantal stappen kan worden gedefinieerd. De temperatuur loopt van TMax in NMax stappen tot de waarde 1/TMax. Als we dus beginnen af te koelen bij T=2, zal het afkoelingsproces eindigen bij T=1/2.

$$T=EXP\{-(2*s*\ln(TMax) * Trunc(n/s))/NMax + Ln(Tmax)\}$$

(5. 3. c.)

Tijdens het afkoelingsproces neemt de invloed van I(α) langzaam maar zeker toe, tot R(α) een minimale bijdrage geeft aan de oplossing van $f_{sa}(R(\alpha),I(\alpha),T)$. Op dat moment reageert het redeneersysteem alsof de functie f_{sa} afwezig is: het zakt rechtstreeks naar het dichtstbijzijnde energiedal, die dan een globaal energiedal moet voorstellen: alle units zijn het maximaal eens.

Hoe in SCORE precies op deze manier lokale energieminima verlaten worden om steeds een lager energiedal op te zoeken kan het beste worden toegelicht aan de hand van het voorbeeld uit subparagraaf 5.2.2 waarin we een netwerk met 5 units trainden d.m.v. vijf trainingsvoorbeelden. In deze paragraaf is geschetst hoe we in een lokaal minimum kunnen blijven steken. In figuur 5.3.a gold:

$I(U_3[0]) = 0.42$
$I(U_3[1]) = 0.55$
$I(U_3[2]) = 0.03$
$I(U_3[3]) = 0$

Een aantal vragen kunnen nu worden gesteld. Wanneer, d.w.z. voor welke T, is het tijdens het afkoelingsproces nog mogelijk dat de nodes $U_3[0]$, $U_3[2]$ of $U_3[3]$ nog geactiveerd kunnen worden, d.w.z. dat $f_{sa}(R(0), I(0), T)$, $f_{sa}(R(2), I(2), T)$ of $f_{sa}(R(3), I(2), T)$ groter is dan $f_{sa}(R(1), I(1), T)$? Voor T=0 weten we dat f_{sa} precies I(α) volgt, maar voor T>0 krijgt het toeval invloed. Er wordt begonnen met de vraag wanner $U_3[3]$ het nog kan winnen van $U_3[1]$.

Wanneer, d.w.z. voor welke ondergrens van T, kán $f_{sa}(R(\tau), I(\tau), T)$ toch nog groter worden dan $f_{sa}(R(\alpha), I(\alpha), T)$ ook al is I(α) groter dan I(τ), waarbij in dit geval $\alpha=U_3[1]$ en $\tau=U_3[3]$. M.a.w. bij welke T kan nog welke helling opgeklommen worden? Als we het hebben over kúnnen, dan nemen we die situatie waarin de randomgenerator maximaal in het voordeel is van τ, dus R(τ)=l en R(α)=0. Het wordt nu een kwestie van invullen van de volgende vergelijking:

$f_{sa}(R(\tau), I(\tau), T) > f_{sa}(R(\alpha), I(\alpha), T) =>$
$R(\tau) * EXP(-1/T) + I(\tau) * EXP(-T) > R(\alpha) * EXP(-1/T) + I(\alpha) * EXP(-T)$
Dus: $1*EXP(-1/T) + 0*EXP(-T) > 0*EXP(-1/T) + 0.55*EXP(-T)$
Hieruit volgt:
$EXP(-1/T) > 0.55 * EXP(-T) =>$
$1/0.55 \times EXP(-T+1/T) =>$
$-LN(O.55) > -T + 1/T =>$
$T^2 - LN(0.55)*T - 1 > 0 =>$
abc-formule:
$T = [LN(0.55) +/- \sqrt{}(LN^2(0.55) +4)] / 2$

Omdat T> 0 valt één oplossing van deze vierkantsvergelijking weg en houden we over: T>0.75. Kortom alleen voor T>0.75 is het in principe nog mogelijk dat de randomgenerator het verschil tussen I(α) en I(τ) kan overbruggen. Voor T<=0.77 zal de oplossing $U_3[3]$ niet meer worden geprobeerd en houdt het systeem vast aan de activering van $U_3[2]$, $U_3[1]$ of $U_3[0]$. Tot wanneer kan deze laatste het nog winnen van $U_3[1]$. Bovenstaand verhaal kan ook worden opgesteld voor $\tau=U_3[0]$:

$f_{sa}(R(\tau), I(\tau), T) > f_{sa}(R(\alpha), I(\alpha), T) =>$
$R(\tau) * EXP(-1/T) + I(\tau) * EXP(-T) > R(\alpha) * EXP(-1/T) + I(\alpha) * EXP(-T)$
Dus: $1*EXP(-1/T) + 0.42*EXP(-T) > 0*EXP(-1/T) + 0.55*EXP(-T)$
Hieruit volgt:
$EXP(-1/T) > 0.13 * EXP(-T) =>$
$1/0.13 \times EXP(-T+1/T) =>$
$-LN(O.13) > -T + 1/T =>$
$T^2 - LN(0.13)*T - 1 > 0$

Oplossen van deze vergelijking resulteert in T>0.41. Dus voor T<=0.41 kan node τ het ook niet meer winnen van node α en zal het systeem vasthouden aan de activering van α. Op dat moment probeert het systeem dus geen alternatieve mogelijkheden meer uit en valt het rechtstreeks naar het dichtstbijzijnde energieminimum.

Stel dat we nu op een moment in het afkoelingsproces zitten dat T nog groot genoeg is om er voor te zorgen dat node $U_3[0]$ de grootste wordt (b.v. bij T=0.5) . Dit kan tot gevolg hebben dat $U_5[1]$ zijn macht verliest om (via dezelfde principes als we net besproken hebben bij U_3) $U_5[0]$ te activeren, die op zijn beurt $U_3[1]$ bevestigt. We zitten dan in de situatie van figuur 5.3.c. We hebben gezien dat dan t.a.v. U_3 geldt:

$I(U_3[0]) = 0.92$
$I(U_3[1]) = 0.05$
$I(U_3[2]) = 0.03$
$I(U_3[3]) = 0$

Welke T moet nú minimaal aanwezig zijn om $U_3[1]$ nog tot activeren aan te zetten? We geven d.m.v. $R(U_3[1])=1$ en $R(U_3[0])=0$, $U_3[1]$ daar de zo groot mogelijke gelegenheid toe. De afleiding wordt hier niet uitgewerkt maar alleen het resultaat wordt geponeerd. De methode is dezelfde als hiervoor beschreven. Resultaat: T>O. 93.

Kortom, en dit is waar het om het draait, tijdens het afkoelingsproces zal tot T=0.41 de randomgenerator er voor kunnen zorgen dat het systeem uit zijn lokaal evenwicht gebracht wordt. Als eenmaal een betere situatie is bereikt (d.w.z. lagere energie; zoals in subparagraaf 5.3.2.1 is gedemonstreerd is de energie E_3=1.82 in de eerste situatie en E_3=1.09 in de tweede situatie) dan moet T>0.93 zijn om het systeem naar het slechtere energieniveau terug te brengen.

In het hiernavolgende wordt het SCORE-annealingsproces puntsgewijs aan een algemene beschouwing onderworpen.

Beschouwing 1

Stel dat, in het algemeen gesproken, van een unit node α de grootste input I heeft. Wanneer kan een node r toch nog winnen?

Er zal nu een uitdrukking worden opgesteld in termen van I(α) en I(τ). We nemen weer R(α)=0 en R(τ)=1.

R(τ) *EXP (-1/T) +I(τ) *EXP(-T) > R(α) * EXP(-1/T) +I(α) * EXP(-T)
Dus: 1*EXP (-1/T) + I(τ) *EXP (-T) > I(α) *EXP (-T)
Hieruit volgt:
EXP (-1/T) > (I(α) - I(τ)) * EXP(-T) =>

1/(I(α) - I(τ)) > EXP (1/T - T) =>
Stel nu: g(T) = EXP (1/T - T)
We houden dan over:
1/(I(α) - I(τ)) > g(T)

g (T) zier er als volgt uit:

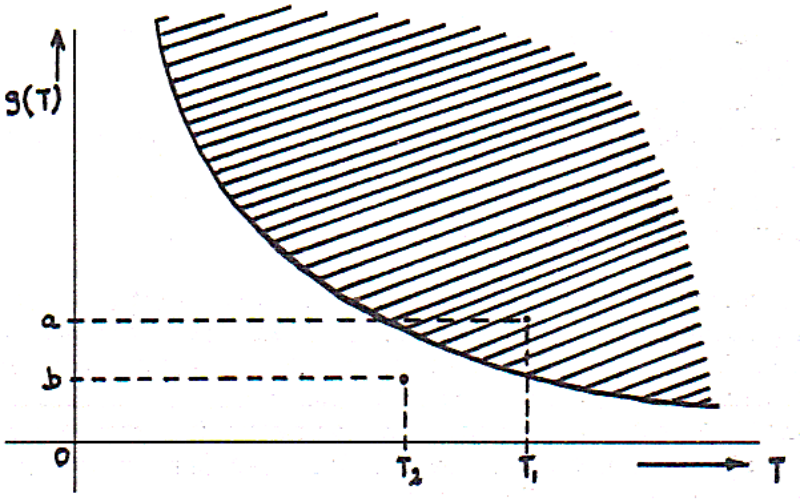


Fig. 5. 3.j De functie EXP(1/T - T)

g(T) vertelt ons dus wanneer de randomgenerator nog in staat is een andere node (T) met een lagere input dan de grootste (α) te activeren. Als 1/(I(α)-I(τ)) op een bepaald moment, d.w.z. bij een bepaalde temperatuur T, binnen het gearceerde gebied valt, dan kan de randomgenerator er in principe nog voor zorgen dat node t wordt geactiveerd. Voor grote waarden van T kunnen alle inputs I door de randomgenerator worden overruled. 1/(I(α)-I(τ)) ligt dan immers altijd in het gearceerde gebied.

Als 1/(I(α)-I(τ)) bijvoorbeeld op T1 de waarde a heeft dan kan node r toch nog winnen van a . Door het activeren van een node die op het eerste gezicht een slechter resultaat geeft, kan het toch zo zijn dat even later van de andere units een bevestiging wordt gekregen, waardoor 1/(I(α)-I(τ)) bijvoorbeeld in punt b terecht komt. b is lager dan a, dus is een betere oplossing. Eenmaal in b aangeland kan de randomgenerator er niet meer voor zorgen dat 'slechtere nodes' worden geprobeerd omdat dat punt nu buiten het gearceerde gebied valt.

Doordat g (T) langzaam stijgt bij dalende T en bij T=0 zelfs oneindig wordt, worden uiteindelijk alle keuzes bevroren. Node T stelt alle nodes van de unit behalve de node met de grootste input voor (a) . Tijdens het afkoelingsproces kunnen als eerste de nodes die een groot inputverschil hebben met de grootste niet meer worden geprobeerd, vervolgens die met een

iets kleiner verschil etc. Het is een voortdurende competitie 'om erbij te blijven'. Uiteindelijk blijft er een node over die een hoog inputverschil (I(α)-I(τ)) heeft met alle anderen. Dit proces streeft dus naar het verkrijgen van dié unitinstellingen met een hoog inputverschil (I(α)-I(τ)), dus een hoge consensus omtrent de activering van node α.

Beschouwing 2

Wat ook interessant is om te weten is het antwoord op de vraag vanaf welke T het maximale verschil tussen de inputwaarden van twee nodes te groot wordt om door de randomgenerator nog te kunnen worden overbrugd.

Het maximale verschil kan zijn: I(α) =1 en I(τ) =0 voor τ<>α.

Wanneer geldt dan toch nog:
$f_{sa}(R(\tau), I(\tau), T) > f_{sa}(R(\alpha), I(\alpha), T) =>$
R(τ) *EXP (-1/T) +I(τ) *EXP(-T) > R(α) * EXP(-1/T) +I(α) * EXP(-T)
Hieruit volgt (R(α)=0 en R(τ)=1):
EXP(-1/T) >EXP(-T) =>
T>1

Dus voor T>1 wordt er nog geen enkele node vergeten. Voor T<=1 gaat dat proces beginnen. Voor dat stuk moet dan ook rustig de tijd worden genomen. Dit is dan een argument voor een negatief exponentiële temperatuurregeling: eerst kunnen de nodes lekker spartelen, dit hoeft niet te lang te gebeuren, om vervolgens subtiel de invloed van inputwaarden I er bij te betrekken.

Beschouwing 3

Tot welke temperatuur T zijn we verplicht af te koelen. Moet dit echt T=0 zijn of mogen we eerder stoppen omdat bijvoorbeeld de stabiliteit misschien al eerder wordt bereikt?

Stel I(α)-I(τ)= Ɛ, waarmee een zeer klein verschil wordt aangeduid in het voordeel van α (€>=0), en R(α)=0, R(τ)=l. Wat nu belangrijk is te weten bij welke T de randomgenerator nog de kracht heeft dit verschil ter grootte van Ɛ te overbruggen. De moeilijkste situatie, namelijk voor Ɛ nadert naar nul, wordt bekeken.

$f_{sa}(R(\tau), I(\tau), T) > f_{sa}(R(\alpha), I(\alpha), T) =>$
R(τ) *EXP (-1/T) +I(τ) *EXP(-T) > R(α) * EXP(-1/T) +I(α) * EXP(-T)
Als we dit weer uitschrijven kom je op een gegeven moment tot de vergelijking:
$T > (LN(Ɛ) + \sqrt{(LN^2(Ɛ) + 4)}) / 2$

In de limiet voor Ɛ nadert naar nul, valt de 4 weg tegen $LN^2(€)$ en houden we over: T>0.

Voor praktijk situaties is dit een veel te zware eis. Het is niet nodig om van het systeem te verwachten dat het de subtiliteiten in een volmaakt homogene kansverdelingen er uit filtert. In het volgende punt wordt beargumenteerd waarom het beginpunt van de temperatuurregeling TMax=2 een goede keuze is. Eerder in deze paragraaf is de formule geponeerd van de negatief exponentiële temperatuurregeling die begon bij TMax en door regelde tot TMin=1/TMax. Dan zou TMin dus moeten worden: 1/2. De waarden Ɛ die dan niet meer gedetecteerd worden bij TMin=1/2 worden dan:

$$1/2 = (LN(Ɛ) + \sqrt{(LN^2(Ɛ) + 4)}) / 2$$

Als we dit oplossen volgt hieruit: Ɛ=0.225. Dus inputverschillen van kleiner dan 0.225 tussen de node met de grootste input en een andere node geven in principe een instabiele situatie. Dit blijkt in de praktijk klein genoeg, omdat op een unit waar consensus kan worden bereikt de verschillen veel hoger zullen liggen (zie ook in het voorbeeld van figuur 5.3.c), en is dus TMin=1/2 geoorloofd.

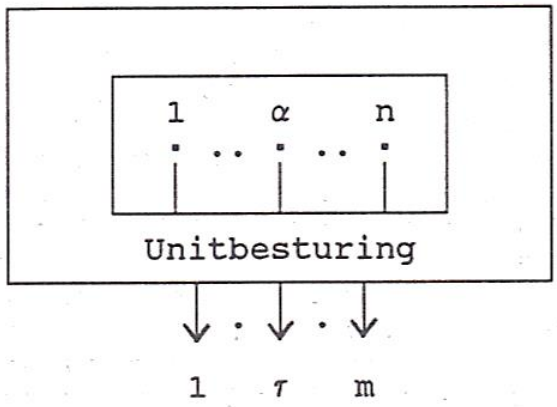
Beschouwing 4

Het beginpunt TMax=2 van de temperatuurregeling verdient een toelichting.

Het liefst nemen we TMax zo laag mogelijk om zoveel mogelijk tijd van de afkoeling te investeren in zinnige bezigheden. Het heeft bijvoorbeeld geen zin om een hele tijd random bezig te zijn, om het laatste moment nog even snel te bevriezen. Als T=2, dan is g(T) =EXP(1/T - T) gelijk aan 0.22. Dus alle inputverschillen groter dan 1/0.22 = 4.5 kunnen door de randomgenerator teniet worden gedaan. Aangezien de nodeinputverschillen nooit groter dan één kunnen zijn, is dat ruim voldoende. Nu is g(T) opgezet vanuit de extreme situatie dat R(α)=0 en R(τ)=1, waarbij α die node is die de grootste input I krijgt. Dit is niet echt reëel. Toch blijkt zelfs dat wanneer R(τ) - R(α) = 0. 22, bij T=2 precies nog verschillen van één overbrugd kunnen worden. Aangezien het afkoelingsproces langzaam moet worden geregeld, en in de praktijk zoals in het volgend hoofdstuk zal blijken soms wel zo'n 600 stappen nodig zijn, krijgen alle nodes dus ruimschoots de mogelijkheid geactiveerd te worden.

5.3.3 Concludering

Als het redenatieproces is beëindigd, hebben alle interne nodes een waarde. Deze waarden zijn nul of één; passief of actief. Voor de gebruiker hebben de interne nodes geen betekenis. Het gaat hem/haar om de externe nodes of instanties. Daarom moeten de antwoorden in termen van interne nodes worden getransformeerd naar antwoorden in termen van externe nodes. Zie hiervoor figuur 5.3.k.



Figuur 5.3.k. Afbeelding van de intern nodes 1<=α<=n op de externe nodes 1<=τ<=m, geregeld door de unitbesturing.

Hier ziet U dat de unitbesturing de activaties van de n interne nodes transformeert naar activaties van de m (m<=n) externe nodes. Zoals in subparagraaf 5.2.2 is beschreven, konden voor één instantie of externe node, meerdere interne nodes worden geïntroduceerd. Zie hiervoor figuur 5.3.l. Daarin worden p interne nodes afgebeeld op 1 externe.

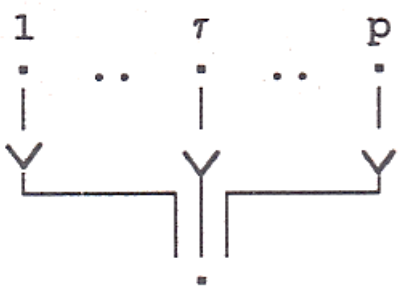


Fig. 5.3.l. De afbeelding van p interne nodes op één externe.

Als één van de p interne nodes actief is, moet naar de buitenwereld die ene externe node worden geactiveerd. Als geen van de p interne nodes actief zijn, moet de externe node ook passief blijven. Deze regeling is dus in handen van de unitbesturing.

In subparagraaf 5.3.1 is al vermeld dat het inferentieproces uit twee fasen bestaat. De eerste fase is de fase waarbij uitgaande van de waarneming, d.m.v. Simulated Annealing, naar een globaal optimum wordt gestreefd. Dit principe is behandeld in subparagraaf 5.3.2. Deze fase is afgelopen als de temperatuur tot z'n minimum is gedaald of wanneer er een aantal inferentiestappen geen verandering meer optreedt. Dit aantal kan binnen SCORE-S worden gespecificeerd d.m.v. de variabele MAXSTABLE. Ongeacht de manier waarop de eerste fase is

beëindigd wordt altijd overgegaan naar fase twee. Deze truc tot geforceerde beëindiging van het inferentieproces is niet een intrinsiek parallelle. Er hangt nu als het ware iets boven het netwerk dat controleert of er units zijn die van waarde veranderen. Lokaal, dus op unitniveau, is dat niet zichtbaar. Voor een simulatie is dit principe handig, maar in een werkelijk parallelle omgeving ingezet is het niet bruikbaar.

In fase twee wordt verder gegaan met het resultaat dat in fase één is bereikt. Het inferentiemechanisme blijft hetzelfde als in fase één, maar nu zonder waarneming en zonder Simulated Annealing, d.w.z. T blijft staan op zijn minimum waarde TMin=1/TMax (maar dit is zo laag dat de temperatuur als nul kan worden beschouwd) . Het effect hiervan is dat het netwerk vervalt naar het dichtstbij liggende energiedal. Fase twee wordt als beëindigd beschouwd als er geen verandering meer in waarden optreedt. De reden van deze fase ligt in het oplossen van inconsistente waarnemingen. Inconsistente waarnemingen kunnen het netwerk in principe in een illegale oplossing vastzetten. De eis was dat het netwerk altijd een legale oplossing zou vinden en desnoods een waarneming zou moeten kunnen corrigeren. Welnu, deze correctie kan plaats vinden in de tweede fase. Dié waarnemingen worden aangepast die niet overeenkomstig de eindinstelling van het netwerk, na de eerste fase, zijn. Dit gaat als volgt in zijn werk.

De eindinstelling na de eerste fase was een situatie met hoge consensus. Daarbij werd meer geluisterd naar bekende waarnemingen dan minder bekende. Als er nu een inconsistentie in de waarnemingen aanwezig is, dan zal het netwerk in de eerste fase convergeren naar een aanpassing aan de bekende waarneming, om vervolgens in fase twee de onbekende waarneming te corrigeren naar de waarde die past bij de bekende waarneming. Hiermee is de inconsistentie hersteld.

5.4 kennisacquisitie

SCORE kent twee manieren van leren of kennisacquisitie, die eerst genoemd en vervolgens besproken zullen worden.

- Initialisatie-Training
- Consultatie-Training

Initialisatie-Training

Deze eerste is het type training dat in subparagraaf 5.2.2 is besproken. In de vorm van trainingsvoorbeelden worden de verbindingen van het netwerk, welke wordt gedefinieerd door een afhankelijkheidsgraaf, geïnitialiseerd. Omdat de verbindingen conditionele kansen voorstellen, kunnen de sterktes van de verbindingen, uitgaande van een netwerk dat eventueel is uitgebreid met extra interne nodes en hidden units, met eenvoudige kanstheoretische principes worden bepaald. Het netwerk is nu klaar om gebruikt te worden

als redeneersysteem. Als blijkt dat de redeneerresultaten niet overeenkomen met wat men ervan verwacht, kan men trainingsvoorbeelden toevoegen en het netwerk opnieuw initialiseren. Dit proces kan net zo lang worden herhaald, tot het systeem naar tevredenheid functioneert.

Dit proces verschilt niet veel van een traditionele kennisacquisitie aanpak. Aldaar kan men ook voortdurend de knowledgebase aanvullen of wijzigen. Een belangrijk verschil is wel dat binnen SCORE de afhankelijkheidsgraaf en bijbehorende trainingsvoorbeelden eerst worden omgezet in een semantisch consistent netwerk, waar dan pas mee geredeneerd kan gaan worden. Het toevoegen van trainingsvoorbeelden kan geen inconsistenties in het netwerk tot gevolg hebben, doordat het opnieuw wordt geïnitialiseerd. Beschouw figuur 5.4.a voor een overzicht van deze methode van kennisacquisitie.

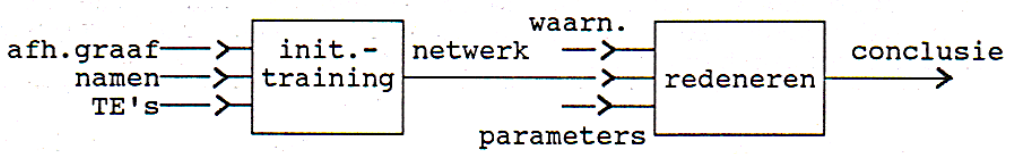


Fig. 5.4.a Handmatige kennisacquisitie.

De input voor het blok 'initialisatie-training' bestaat uit de afhankelijkheidsgraaf, de namen van de eigenschappen en de trainingsvoorbeelden (Training Examples). Het resultaat is een consistent netwerk. Samen met de waarneming en instellingen voor de benodigde systeemparameters kan er worden geredeneerd, resulterend in een conclusie. Kennisacquisitie bestaat uit het handmatig uitbreiden van de TE's.

Deze manier van kennis acquisitie kan ook worden geautomatiseerd. Hiermee komen we op de tweede manier van kennisacquisitie.

Consultatie-Training

Consultatie-Training is het automatisch trainen van het netwerk, tijdens de consultatie sessie. Er worden twee mogelijke uitvoeringen van Consultatie-Training besproken, waarvan de eerste (o.a. vanwege zijn eenvoud) in SCORE-S is geïmplementeerd. Het resultaat van beide uitvoeringen is exact hetzelfde, maar de eerste heeft een hogere tijds- en de tweede een hogere ruimtecomplexiteit. Beschouw figuur 5. 4. b.

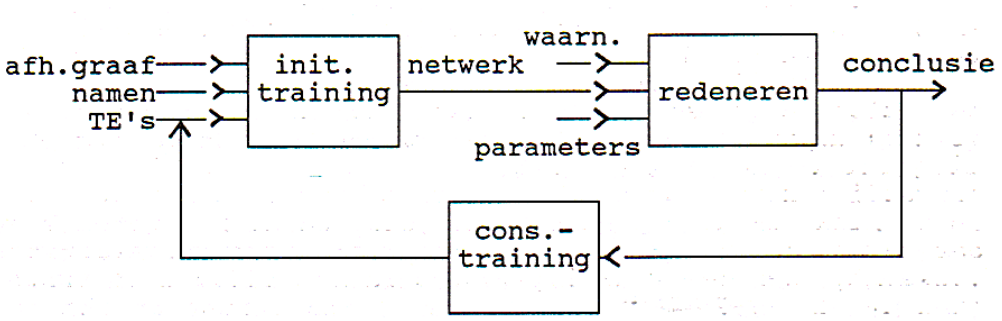


Fig. 5. 4.b Automatische kennisacquisitie.

De Consultatie-Training bestaat uit het automatisch uitbreiden van de TE-set gevolgd door de initialisatie van het netwerk. Na elke Consultatie-Training wordt er een nieuw netwerk gegenereerd.

Er is een blok 'Consultatie-Training' toegevoegd, t.o.v. figuur 5.4.a. Deze zorgt er voor dat, wat de gebruiker eerst nog handmatig moest doen, nu automatisch kan geschieden. De Consultatie-Training zorgt er voor dat de TE's worden uitgebreid met het resultaat van een redenatie. Als bijvoorbeeld veel dezelfde waarnemingen, resulterend in een bepaalde conclusie, worden gedaan, kan het netwerk via deze optie daarop worden getraind. In SCORE-S kan de gebruiker na ieder redenatieresultaat besluiten of het resultaat toegevoegd moet worden aan de TE's om het netwerk vervolgens automatisch opnieuw te initialiseren.

Het nadeel van deze manier van trainen is het feit dat het relatief tijdsintensief is, omdat het netwerk steeds helemaal opnieuw moet worden gedefinieerd. Dit is ook een sequentieel proces, doordat Consultatie-Training en Initialisatie-Training sequentiële processen zijn. Een belangrijk voordeel van deze methode is het feit dat we voortdurend de redenen van de kennis van het netwerk voorhanden hebben. De kennis in het netwerk is immers een 1: 1 afbeelding van de trainingsvoorbeelden die expliciet aanwezig zijn. Als niet helemaal duidelijk is waarom het netwerk op een bepaalde manier redeneert, kunnen we de trainingsvoorbeelden onderzoeken en eventueel besluiten daar wijzigingen in aan te brengen. Trainingsvoorbeelden spreken meer tot de verbeelding dan verbindingen dat doen.

Een tweede methode van Consultatie-Training komt nu aan bod. Beschouw hiervoor figuur 5.4.c.

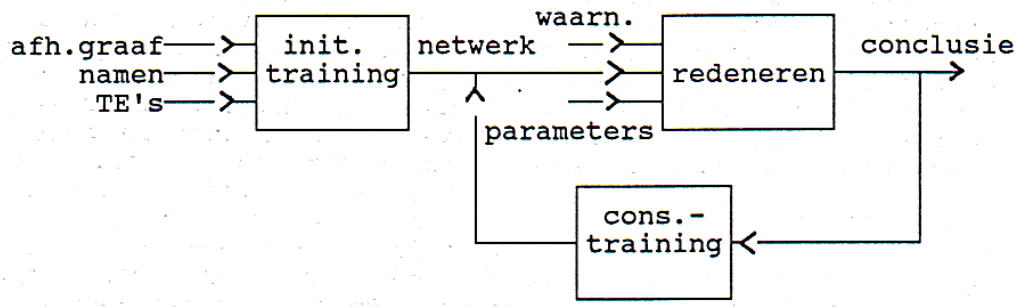


Fig. 5.4.c Automatische kennisacquisitie.

De Consultatie-Training bestaat uit het automatisch aanpassen van de gewichten van de verbindingen van het netwerk.

Zoals in deze figuur is te zien, wordt het netwerk éénmalig geïnitialiseerd. Dat was een sequentieel proces. Als het netwerk eenmaal is opgebouwd, is het redeneren, en bij deze methode nu ook het leren, intrinsiek parallel. Het leren berust op het aanpassen van de sterktes van de verbindingen. Dit proces van aanpassen kan lokaal, d.w.z. door de netwerkprocessoren, geschieden.

Het wordt hiermee de taak van de processoren de semantiek van de verbindingen, dus het feit dat de verbindingen conditionele kansen voorstellen, te handhaven. Beschouw figuur 5.4.d:

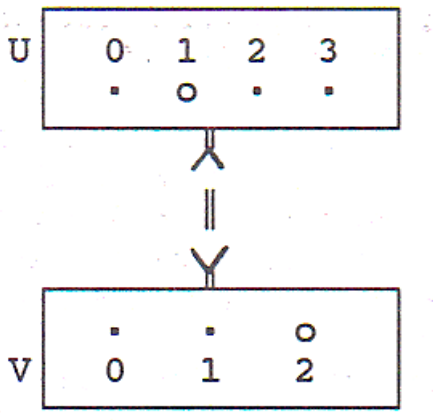


Fig. 5.4.d Twee units U en V, met actieve nodes U[1] en V[2]

Stel dat dit het resultaat is van twee units U en V, na een redenatieproces. Reeds eerder is beschreven dat de verbindingen vanaf parentunits nooit worden gewijzigd omdat deze verbindingen tot taak hebben de semantiek aan het netwerk op te leggen. De waarde van die verbindingen blijven één. Het gaat nu dus om de verbindingen van V naar U. De verbindingen van V[2] naar alle nodes van unit U moeten worden aangepast omdat een conditionele kans alleen betrekking heeft op het wáár zijn van de premisse. De enige premisse die nu waar is, is V[2]. De verbindingen afkomstig van de overige nodes kunnen constant blijven omdat die nodes onwaar zijn.

Omdat de sterktes van de verbindingen echte kansen betreffen, kunnen we niet de verbindingen met één of andere constante factor verhogen of verlagen, zoals dat wel in Connectionist Systems gebeurt. Bij iedere node moeten hiertoe evenveel registers als het verbindingen naar parentunits heeft, worden opgenomen. In deze registers moet worden bijgehouden hoe vaak de node, waar de bijbehorende verbinding naar toe loopt, actief is geweest. Deze registers worden aangeduid met r(U[α],V[δ]), waarbij V dus een parentunit is van U. Verder is er nog één register nodig waarin wordt bijgehouden hoe vaak de node zélf actief is geweest. Dit register wordt aangeduid met de naam r(U[α]), als het node U[α] betreft. De sterkte van een verbinding W(U[α],V[δ]) wordt dan r(U[α],V[δ])/r(U[α]), overeenkomstig de kanstheorie.

Stel nu dat er 10 redenaties hebben plaatsgevonden, waarbij U[0] 2 keer, U[1] 4 keer en U[2] ook 4 keer actief is geweest. Dan geldt r(U[0]) =2, r(U[1]) =4 en r(U[2])=4. Van de vier keer dat U[2] actief was, kan het zo zijn dat V[0] 0 keer, V[1] 1 keer, V[2] 1 keer en V[3] 2 keer actief was. Hierdoor geldt: r(U[2],V[0])=0, r(U[2],V[1])=1, r(U[2], V[2])=1 en r(U[2],V[3]) =2, waaruit volgt: W(U[2],V[0])=0, W(U[2],V[1])=1/4, W(U[2], V[2])=1/4 en W(U[2],V[3]) =2/4.

Stel verder dat een 11-de redenatieproces het resultaat oplevert zoals aangegeven in figuur 5.4. De procedure voor het aanpassen van de verbindingen wordt dan: incrementeer r(U[2]), dus r(U[2]) wordt 5, en r(U[2],V[1]) , dus r(U[2],V[1]) wordt 2, en bereken opnieuw de waarden van alle verbindingen van U[2] naar V. Hieruit volgt: W(U[2], V[0]) =0, W(U[2],V[1])=2/5, W(U[2], V[2]) =1/5 en W(U[2],V[3]) =2/5. De som van de verbindingen blijft keurig één.

Het nadeel van deze methode is dat het vrij veel geheugenruimte in beslag neemt. In een simulatieomgeving kan dit worden geïmplementeerd door voor elke verbinding niet de sterkte van een verbinding in één real-variabele (in PASCAL is dit een variabele met een waarde uit de reële getallen) op te slaan, maar in twee integer-variabelen (in PASCAL is dit een variabele uit de natuurlijke getallen) . Als een verbinding de waarde twee en vijf draagt (de real-variabele zou de waarde 0. 4 hebben gehad) , weten we dat de premisse vijf maal is waargenomen en de conclusie twee maal. Als de premisse actief wordt verhogen we de vijf naar zes en als de conclusie ook nog actief is, kan de twee worden verhoogd naar drie.

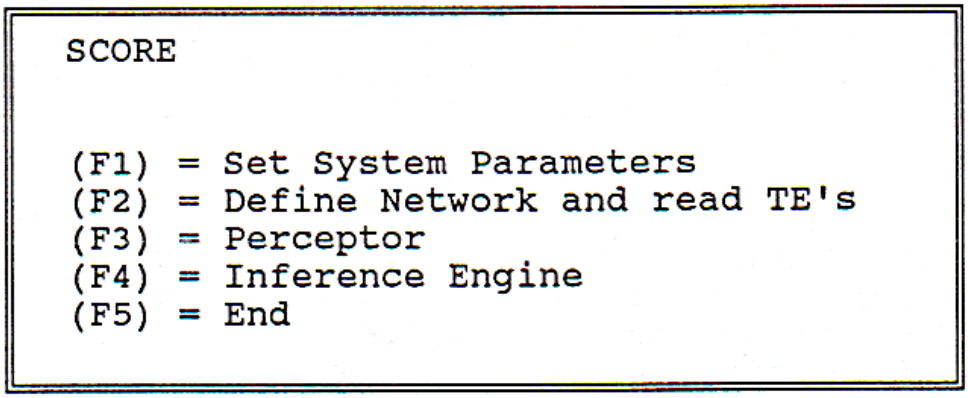
6 Simulatie

6.1 Inleiding

Ten behoeve van de beoordeling van SCORE is een simulatieprogramma geschreven met de naam SCORE-S, hetgeen staat voor SCORE-Simulatie. Met dit programma kunnen niet al te grote problemen zeer eenvoudig en overzichtelijk worden getest.

Het programma is geschreven in de programmeertaal TURBO PASCAL. De reden voor deze keuze en bijvoorbeeld niet het qua opzet vergelijkbare TURBO C, was het feit dat voor TURBO PASCAL veel handige routines voorhanden waren die het programmeren deed versnellen. Het nadeel van het gebruik van deze taal, namelijk de lagere executiesnelheid van het programma, is voor lief genomen omdat het snelheidsaspect nog niet zo'n rol speelt in een testomgeving. In de praktijk ingezet, zou dit wel belangrijk worden.

Nadat de executable file (SCORE.EXE) is geïnstalleerd op een gewone Personal Computer, kan het worden opgestart d.m.v. het intypen van de naam 'SCORE'. Op het beeldscherm verschijnt:



D.m.v. de cursortoetsen of rechtstreeks d.m.v. een functietoets kan een keuze worden gemaakt. D.m.v. keuze 1 kunnen een aantal systeeminstellingen worden geregeld. Keuze 2 zorgt er voor dat een netwerkdefinitie met bijbehorende Training Examples uit een file wordt ingelezen. D.m.v. optie 3 kan een waarneming op het netwerk worden gepleegd om via keuze 4 vervolgens het redenatieproces op te starten. Eén voor één zullen deze keuzes worden besproken.

Set System Parameters

Als voor deze optie is gekozen verschijnt op het scherm:

```
Set System Parameters

Unitdispatcher  :   1          (0=FIXED,1=RANDOM)
PerceptionMode
     No Clamping    :   0      (0=FALSE,1=TRUE)
     Beta           :   1.00
MaxStable       :   10
# Monitor Cycles:   0          (0=MONITOR OFF)
Logfile         :   0          (0=OFF,1=ON)
```

Dit zijn de standaardinstellingen waarvan gebleken is dat die voor de meeste problemen acceptabel zijn. D.m.v. de cursortoetsen kan er in willekeurige volgorde langs deze velden worden gewandeld.

De keuze van de eerste optie 'Unitdispatcher ' is bepalend voor de volgorde waarin de units in deze sequentiële simulatie worden afgehandeld. FIXED houdt in dat de units in vaste volgorde worden doorlopen: cyclisch vanaf de unit met de laagste index tot en met de unit met de hoogste index. Op de indexen wordt later terug gekomen. RANDOM zorgt er voor dat de units in willekeurige volgorde worden afgehandeld. Bij iedere unit wordt een 'tag' bijgehouden waardoor eerst alle units afgehandeld moeten zijn voordat een unit opnieuw kan worden geselecteerd.

De PerceptionMode bevat twee variabelen die betrekking hebben op de perception, ofwel de waarneming. Ten eerste de variabele 'No Clamping'. Als deze uit staat (nul is), wordt de waarneming geclampt. In subparagraaf 5.3.1 is beschreven dat clamping wordt bewerkstelligd door een hoge β en alleen plaatsvindt bij de units met een zekere waarneming. Bij een onzekere waarneming is β gewoon gelijk aan één, waardoor omgeving (d.w.z. parent en childs samen) en waarneming even zwaar worden geteld. Bij deze instelling kan de variabele Beta, de tweede perceptionmode variabele, niet worden gewijzigd. Als de No Clamping variabele wordt aangezet (één wordt), worden zékere waarnemingen niet meer hard op de unit overgenomen. De variabele Beta kan nu wél worden gewijzigd. De gebruiker kan nu specificeren wat de invloed van de waarneming t.o.v. de omgeving moet zijn. In formule 5.3.a is precies aangegeven hoe, d.m.v. Beta, die invloed wordt geregeld.

De volgende variabele die moet worden gespecificeerd is de variabele MaxStable. In paragraaf 5.3.3 is beschreven dat d.m.v. deze variabele kan worden ingesteld na hoeveel stabiele inferentiestappen, de conclusie door het systeem moet worden geaccepteerd. Als

MaxStable een hoge waarde heeft (bijvoorbeeld 999), dan zal het inferentieproces pas eindigen als de minimale temperatuur is bereikt. Vaak is de stabiele situatie al veel eerder bereikt, wat gedetecteerd kan worden m.b.v. de variabele MaxStable.

De volgende twee variabelen, '# Monitor Cycles' en 'Logfile', zijn aanwezig voor respectievelijk het testen van de performance van het systeem en het testen van de programmatuur. D.m.v. de eerste kan worden gespecificeerd hoe vaak hetzelfde redenatieproces, d.w.z. de zelfde waarneming op het zelfde netwerk, dient plaats te vinden. De resultaten worden dan weggeschreven naar de file 'monitor.fil'. Dit i.v.m. de performance statistieken zoals die in paragraaf 6. 2 zijn opgenomen. Als de variabele 'logfile' aan staat wordt er debug informatie uit het programma op het scherm afgedrukt. Als het uit staat, dan niet.

Define network and read TE's

Deze tweede optie uit het hoofdmenu van SCORE verzorgt het inlezen van het netwerk met bijbehorende trainingsvoorbeelden. Het creëert units, nodes, verbindingen tussen de nodes en bepaalt de sterktes van de verbindingen. Na het aanroepen van deze optie verschijnt in beeld:

Filename: Sarco.def

Nu is het mogelijk deze filename te wijzigen. Het voorbeeld dat in de volgende paragraaf wordt besproken staat opgeslagen in de file Sarco.def. Dit voorbeeld is overgenomen van Gallant, zie paragraaf 2.5, die dit voorbeeld gebruikte om zijn expertsysteem te testen. Beschouw nogmaals de figuren 2.5.a en 2.5.b. Hierin ziet U de symptomen, ziektes en behandelingsmethoden en het overzicht van de structuur. In figuur 2.5.c staan de trainingsvoorbeelden afgedrukt. In bijlage III is het uiteindelijke netwerk van Gallant opgenomen.

Nadat de filename is ingevoerd antwoord het systeem, als een soort controle, met het aantal units waaruit het netwerk bestaat.

De netwerkdefinitie Sarco.Def ziet er als volgt uit:

```
Units
{
   1: Swollenfeet;
   2: RedEars;
   3: HairLoss;
   4: Dizziness;
   5: SensAretha;
   6: Plac.Allergy;
   7: Supercilliosis;
   8: Namastosis;
   9: Placibin;
   10:Biramibio;
   11:Posiboost;
}
Dependencies
{
   0 0 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0 0
   1 1 1 0 0 0 0 0 0 0 0
   0 0 1 1 1 0 0 0 0 0 0
   0 0 0 0 0 1 1 1 0 0 0
   0 0 0 0 0 0 1 1 0 0 0
   0 0 0 0 0 0 0 0 1 1 0
}
Examples
{
   1 1 1 0 0 0 1 0 1 0 1
   0 0 0 1 1 0 0 1 1 1 0
   0 0 1 1 0 1 1 1 0 0 0
   1 1 0 0 1 0 0 0 0 0 0
   1 0 0 1 1 1 1 1 0 1 1
   1 0 0 1 1 0 1 1 1 1 0
   1 1 1 0 0 1 1 0 0 0 0
   0 1 1 0 1 1 0 1 0 0 0
}
```

Als eerste moeten de units met bijbehorende namen worden opgegeven. Net als bij het systeem van Gallant, moeten de units in dié volgorde worden gespecificeerd, waarbij unit U_a vóór unit U_b in de lijst komt als U_b afhankelijk is van U_a (Zoals U_1 voor U_7 komt omdat U_7 afhankelijk is van U_1). De nummering van de units dient te geschieden vanaf het getal één, steeds met één oplopend. Iedere regel wordt afgesloten met een punt-comma en het geheel van unitbeschrijvingen wordt omsloten door een accolade-openen en een accolade-sluiten. De indeling, dus het aantal returns, spaties, etc., is niet van belang.

Het tweede deel van de netwerkdefinitie bevat een beschrijving van de unitafhankelijkheden. Omdat we in dit geval 11 units hebben, ontstaat er een tabel van 11x11 elementen. Net als bij Gallant staat element ij voor de informatie of unit i afhankelijk is van unit j. Een 1 staat voor afhankelijk, een 0 voor onafhankelijk. Zo is unit 7 dus blijkbaar afhankelijk van unit 2 en is unit 1 nergens van afhankelijk. T.a.v. Gallant is in de afhankelijkheidsstructuur één element veranderd van een 1 in een 0: Unit 10 is niet meer rechtstreeks afhankelijk van unit 3. Semantisch gezien is die constructie namelijk vreemd: $U_8=U_3+U_4+U_5$ en ook $U_{10}=U_7+U_3+U_8$. U_3 is nu zowel child van U_8 als van U_{10}, terwijl U_8 ook child is van U_{10}. Dit is opgelost door de relatie van U_3 met U_{10} te verwijderen omdat U_{10}, via U_7 en U_8 toch al indirect afhankelijk is van U_3. De afhankelijkheden zien er nu uit als in figuur 6.1.a.

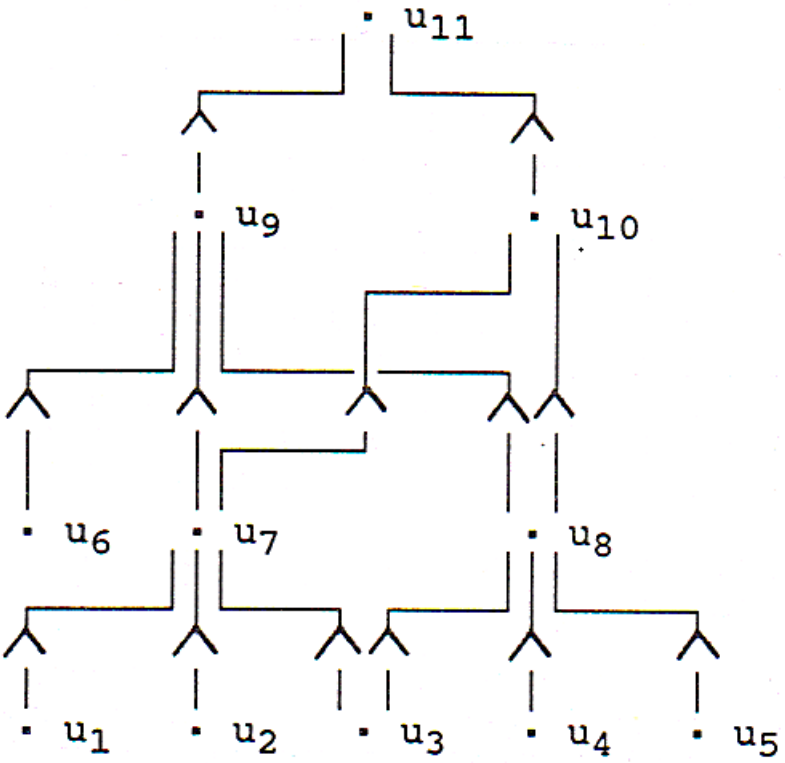


Fig. 6.1.a Een grafische weergave van de afhankelijkheden in Sarco.def

Figuur 6.1.a stelt dus het semantisch model voor. Er zijn nu nog geen instanties gedefinieerd, maar t.a.v. welke instantiecombinaties de omkeerbaarheidseis moet gelden is nu d.m.v. de afhankelijkheidsstructuur wel al vastgelegd. Zo moet bijvoorbeeld iedere instantie uit U_8 worden opgespannen door een unieke combinatie van instanties uit U_3, U_4 en U_5. En omgekeerd mogen instantiecombinaties uit U_3 , U_4 en U_5 slechts worden afgebeeld op precies één instantie uit U_8. Als de trainingsvoorbeelden niet aansluiten op deze eisen moeten er

extra nodes, respectievelijk hidden units worden geïntroduceerd. Hoe beter de structuur, hoe minder extra nodes en hidden units, hoe beter de performance.

Het laatste onderdeel is nu nog het inlezen van de trainingsvoorbeelden. In subparagraaf 5.2.2 is al genoemd dat in SCORE-S omkeerbaarheidsfouten, m.b.t. het introduceren van extra nodes, automatisch worden hersteld maar dat hidden units (nog) niet automatisch worden gecreëerd. Dat dit nog niet in SCORE-S aanwezig is heeft te maken met het feit dat het voorbeeld Sarco.def omkeerbaarheidsfouten in die richting ook niet kent. Omkeerbaarheidsfouten in de andere richting worden dus wel automatisch hersteld en dat is maar goed ook, want in de trainingsvoorbeelden zitten die kant op ook heel wat fouten. Zo zou bijvoorbeeld niet alleen ($U_1[0]$, $U_2[0]$, $U_3[0]$) maar ook ($U_1[1]$, $U_2[1]$, $U_3[1]$) moeten resulteren. Hier moeten dus twee nodes voor worden geïntroduceerd. De oude en de gewijzigde trainingsvoorbeelden staan opgenomen in figuur 6.1.b.

	TE's OUD (Extern)											TE's NIEUW (Intern)										
Unit#	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11
TE#1	1	1	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0
TE#2	0	0	0	1	1	0	0	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1
TE#3	0	0	1	1	0	1	1	1	0	0	0	1	1	0	1	0	1	2	2	2	2	2
TE#4	1	1	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	3	3	3	3	3
TE#5	1	0	0	1	1	1	1	1	0	1	1	0	1	1	1	1	1	4	1	4	4	4
TE#6	1	0	0	1	1	0	1	1	1	1	0	0	1	1	1	1	0	4	1	5	4	5
TE#7	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	6	0	6
TE#8	0	1	1	0	1	1	0	1	0	0	0	1	0	0	0	1	1	5	4	7	5	7

Fig. 6.1.b. De oude trainingsvoorbeelden (TE's OUD) tegenover de getransformeerde trainingsvoorbeelden (TE 's NIEUW).

De trainingsvoorbeelden sluiten nu aan bij de netwerkstructuur. In figuur 6.1.c is een schets opgenomen van het netwerk met al zijn interne nodes. De nodes aangegeven door een 'o' zijn de externe nullen en de nodes aangegeven door een '•' zijn de externe enen.

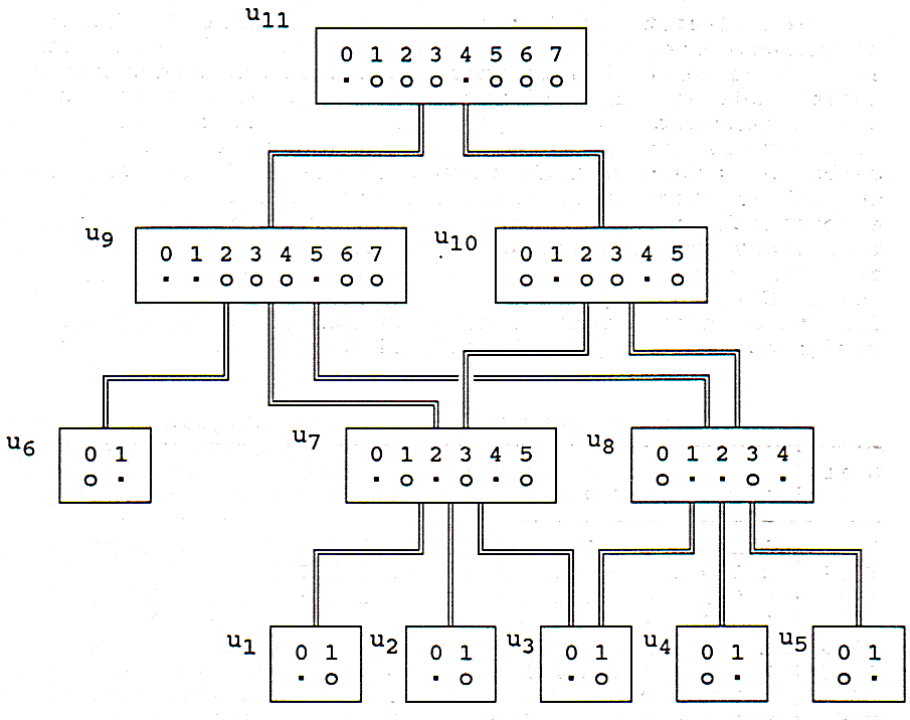


Fig. 6.1.c Een compleet overzicht van Sarco.Def met afhankelijkheden en interne nodes.

Het netwerk is nu gedefinieerd en de 'initialisatie-training' (zie paragraaf 5.4) is afgerond. Alle verbindingen tussen de nodes hebben nu hun waarde overeenkomstig de trainingsvoorbeelden. Zo is bijvoorbeeld $P(U_7\text{-i}[0] \mid U_1\text{-i}[0]) = 2/5$, waardoor $W(U_1\text{-i}[0], U_7\text{-i}[0]) = 2/5$. Dit kan met eenvoudig kanstheoretische principes uit figuur 6.1.b (het deel TE's-NIEUW, dus in termen van de interne nodes) worden gehaald: van de vijf keer dat U_1-i[0] voorkwam, kwam U_7-i[0] twee keer voor.

Perceptor

Voordat er echt met het netwerk geredeneerd kan worden moet er natuurlijk eerst iets worden waargenomen. Dat kan via deze derde keuze. Op het scherm verschijnt:

```
UnitCode        UnitDescription        NodesExtern
   -1
```

De cursor staat bij de -1. -1 is het stopcriterium. Als nu meteen een return wordt gegeven, gaan we terug naar het hoofdmenu en is er niets waargenomen. Als we de -1 overschrijven met bijvoorbeeld een 7, dan verschijnt er:

```
UnitCode      UnitDescription      NodesExtern
    7         Supercilliosis         1       0
                                     0
```

Unit 7 heeft dus de naam 'Supercilliosis' en twee externe nodes 1 en 0. (Voor de gebruiker zijn alleen de externe nodes zichtbaar!). De cursor staat bij de 0 onder node 1. Nu kan een kanswaarde worden opgegeven. Na return springt de cursor onder node O :

```
UnitCode      UnitDescription      NodesExtern
    7         Supercilliosis         1       0
                                     0.750   0
```

Nu mag node 0 slechts de waarde 0.25 krijgen. De som van de kansen mag immers niet ongelijk aan één zijn. Als dit wel zo is wordt er een foutmelding gegeven en wordt de waarneming hersteld. Het resultaat ziet er in het legale geval als volgt uit:

```
UnitCode      UnitDescription      NodesExtern
    7         Supercilliosis         1       0
                                     0.750   0.250
    -1
```

De cursor staat nu weer bij de -1. Als return wordt gegeven, dan is de perceptionsessie afgesloten. Als een nieuwe unitcode wordt opgegeven, wordt een volgende waarneming gepleegd. Dit kan worden herhaald tot eventueel op alle units een waarneming is gepleegd. Meestal zal de waarneming zich uitstrekken over een klein aantal units.

Inference Engine

De inferentie kan nu worden opgestart, maar niet voordat een aantal voor de inferentie noodzakelijke parameters zijn ingesteld. Deze parameters worden met hun defaultwaarden getoond. De gebruiker kan deze eventueel wijzigen.

```
Initial Temperature (TMax*10):  20
Number of cycles per T (s)   :   1
Total number of cycles (NMax): 600
Display results, # Steps     :   1
```

Als de waarden zijn ingesteld wordt het inferentieproces automatisch opgestart. Het scherm wat nu verschijnt kan er als volgt uitzien:

```
Initial Temperature (TMax*10):  20  19.908
Number of cycles per T (s)   :   1  0
Total number of cycles (NMax): 600  2
Display results, # Steps     :   1  0

        Reasoning...
  Press <RETURN> for interrupt
```

	α-e	α-i	r1,τ	r2,τ	r3,τ
Swollenfeet	1	0	1.00(1)		
RedEars	0	1	1.00(1)		
HairLoss	1	0	0.50(1)		
Dizziness	0	0	1.00(1)		
SensAretha	0	0	1.00(2)		
Plac.Allergy	1	1	1.00(2)		
Supercilliosis	0	3	0.50(4)	0.61(3)	0.25(0,2,4)
Namastosis	1	4	0.50(4)	0.39(1)	
Placibin	1	0	1.00(8)	0.42(8)	
Biramibio	0	5	1.00(6)	0.50(4)	
Posiboost	0	2		0.50(1)	

De onderstreepte delen zijn in de simulatie geïnverteerd. De eerste vier waarden die in het kopje van het scherm zijn onderstreept geven de huidige stand van zaken aan. Tijdens het inferentieproces, begint de temperatuur in dit geval bij 20 (TMax is dan 2) , en zal volgens formule 5.3.b afnemen. In die formule komen ook de variabelen s en TMax voor. Deze staan voor respectievelijk het aantal stappen dat de temperatuur constant gehouden moet worden en het totaal aantal stappen. D.m.v. het laatste veld kan worden gespecificeerd hoe vaak de resultaten op het scherm dienen te worden afgedrukt. Als daar bijvoorbeeld 10 wordt opgegeven, wordt eens in de tien stappen het scherm opnieuw beschreven. Dit i.v.m. snelheidswinst. De bij behorende onderstreepte waarde loopt dan steeds van 0 tot en met 3.

Supercilliosis is ook onderstreept hetgeen inhoudt dat op die unit een waarneming is gepleegd. Als de geactiveerde externe node tijdens de waarnemingsfase een kans groter dan nul heeft toegekend gekregen, wordt deze ook onderstreept. Zo kan worden gecontroleerd of de keuze van activering overeenkomstig de waarneming is: als een node tijdens de waarneming kans nul heeft toegekend gekregen, zouden we het antwoord als fout kunnen beschouwen als de naam van de unit wel is onderstreept en de keuze α-e niet (dit is de extern geactiveerde node waarop in de volgende alinea wordt teruggekomen). Dan is er een oplossing gevonden die niet overeenkomstig de waarneming is. Dit geldt alleen als het een consistente waarneming betreft. Een inconsistente waarneming kan nooit tot een resultaat leiden waarbij alle waarnemingsadviezen ook werkelijk door de unit zijn overgenomen.

De eerste kolom achter de unitnamen is α-e, hetgeen staat voor de externe node α. Deze kolom is waar de gebruiker in is geïnteresseerd. De stabiele situatie representeert daar de oplossing van het probleem. In de kolom α-i (α-intern) is zichtbaar wat intern nu precies is geactiveerd. Bij één externe node kunnen immers meerdere interne nodes horen. Als er nog geen stabiele situatie is bereikt (de activering van de interne nodes verandert nog steeds), betekent de activering van bijvoorbeeld U_1-i[0] (onder α-i staat bij Swollen Feet een nul) dat op dit moment U_1-i[0] wordt geloofd. Dit heeft gevolgen voor U_7 die ook weer wat gaat geloven, etc. Het feit dat op dit moment geldt: (U_1[0] , U_2[1], U_3[0]) maakt dat op unit U_7, node 3 wordt geadviseerd, uitmondend in een $r_{2,U7[3]}$ van 0.61. Deze combinatie kwam niet voor, waardoor de adviezen van U_1, U_2 en U_3 worden gemiddeld (de algoritmes hoe de adviezen worden gebundeld in $r_{1,\tau}$, $r_{2,\tau}$ en $r_{3,\tau}$ zijn beschreven in subparagraaf 5.2.1). Activering van node U_7-i[3] maakt op zijn beurt dat U_7 de activering van (U_1[0] , U_2[0], U_3[1]) adviseert. Etc.

Het geheel van activeringskeuzes staat onder invloed van de temperatuur. In formule 5.3.b is precies aangegeven hoe dit gebeurt. Daarbij is I(τ) gelijk aan de middeling (voor zover aanwezig) van $r_{1,\tau}$, $r_{2,\tau}$ en $r_{3,\tau}$, die allen op het scherm te zien zijn. Achter de bijbehorende kanswaarden staat tussen haakjes aangegeven wat τ op dat moment is. Dié τ met bijbehorende waarde wordt afgedrukt die het hoogste advies aan de unit geeft. Als we naar unit 7 kijken krijgt node U_7-i(4) het hoogste advies van de parents; U_7-i[3] het hoogste advies van de childs en U_7-i[0] het hoogste advies van de perceptor (ook nodes U_7-i[2] en U_7-i[4] zijn de hoogste, deze worden dan ook afgedrukt). Er is dus duidelijk geen consensus. Dit heeft natuurlijk te maken met de hoge temperatuur (19.908) waardoor de nodes min of meer random worden geactiveerd.

Uiteindelijk moet er een situatie ontstaan waarbij de adviezen van parents, child en perception met elkaar in overeenstemming zijn. Concreet houdt dit in dat per unit de waarden tussen haakjes, dus de nodes τ , in overeenstemming moeten zijn met de node α-i die uiteindelijk van die unit is geactiveerd. Het resultaat kan er bijvoorbeeld als volgt uitzien:

```
Initial Temperature (TMax*10):  20  9.954
Number of cycles per T (s)   :   1  0
Total number of cycles (NMax): 600  600
Display results, # Steps     :   1  0

        Reasoning...
  Press <RETURN> for interrupt

                α-e α-i    r1,τ      r2,τ      r3,τ
Swollenfeet      0   1     1.00(1)
RedEars          0   1     1.00(1)
HairLoss         1   0     1.00(0)
Dizziness        1   1     1.00(1)
SensAretha       0   0     1.00(0)
Plac.Allergy     1   1     1.00(1)
Supercilliosis   1   2     1.00(2)  0.61(2)  0.25(0,2,4)
Namastosis       1   2     1.00(2)  0.58(2)
Placibin         0   2     1.00(2)  1.00(2)
Biramibio        0   2     1.00(2)  1.00(2)
Posiboost        0   2              1.00(2)

          Without perception, Press <RETURN>
```

Nu is er volledige consensus over de geactiveerde nodes. Er kan nu gesteld worden dat α=τ. Ook bij unit 7 (Supercilliosis) is er consensus over de keuze van α, dus de uiteindelijk geactiveerde, en de grootste waarneming τ, maar ook de andere hoogste τ's worden afgedrukt. De adviezen $r_{3,0}$, $r_{3,2}$ en $r_{3,4}$ zijn allen gelijk aan 0.25 (namelijk de externe waarneming van 0.75, verdeeld over drie interne nodes). De 2 zit erbij dus is de consensus compleet.

Als na dit resultaat een return wordt gegeven, wordt de inferentie zonder waarneming bij TMin (=1/TMax) uitgevoerd (zie de paragrafen 5.3.1 en 5.3.3). In dit geval bevindt het systeem zich in een prima evenwicht. Als de waarneming van unit 7 wordt losgekoppeld, wordt de keuze van activering van U_7-i[2] in stand gehouden door de parents en childs van U_7. Die adviseren immers ook U_7-i[2] . Als er discrepantie zou zijn in die adviezen krijgt het systeem in de volgende inferentiefase de mogelijkheid de waarneming te corrigeren. In dit geval is het na één slag klaar omdat het meteen ontdekt dat er niets veranderd.

Tot slot van de inferentie komt het systeem met de vraag:

```
Store Result?(y/n) n
```

De cursor staat op de laatste n. Als nu een return wordt gegeven is het proces beëindigd en keren we terug naar het hoofdmenu om vandaar de systeemparameters te wijzigen, een nieuw netwerk te laden, een nieuwe waarneming te plegen, de inferentie opnieuw op te starten of gewoon terug te gaan naar het operating system. Indien 'y' wordt ingetoetst wordt het resultaat bijgeschreven in de lijst van trainingsvoorbeelden. In dit geval zouden de trainingsvoorbeelden in de file Sarco.Def worden uitgebreid met de regel:

```
0 0 1 1 0 1 1 1 0 0 0
```

waarop het netwerk automatisch opnieuw wordt getraind. Deze manier van 'consultatie training' is besproken in paragraaf 5.4. De verbindingen hebben nu waarden gekregen die zijn aangepast aan de herkenning van dit resultaat.

6.2 Performance

In deze paragraaf wordt , a.h.v. testen met het simulatiepakket SCORE-S dat in de inleiding van dit hoofdstuk is besproken, de performance van SCORE getest. Doordat het probleem op lossend vermogen van SCORE d.m.v. zijn temperatuurregeling stochastisch is (het energielandschap wordt immers, afhankelijk van de temperatuur, in het begin meer en later in mindere mate kriskras doorzocht) kan op het ene moment antwoord A en op het andere moment antwoord B op een zelfde vraag worden gegeven. SCORE kan daardoor zelfs terugkomen met een volledig verkeerd antwoord. Wat nu interessant wordt, zijn de statistieken van de oplossingen. Als een paar keer een fout antwoord op een groot aantal testen wordt gegeven hoeft dat niet zo erg te zijn. Dit kan bijvoorbeeld liggen aan het feit dat de temperatuur in te weinig stappen van het maximum naar het minimum moet dalen of dat het probleem überhaupt te ingewikkeld is voor SCORE-S. In het hiernavolgende zullen dus statistische resultaten van verschillende problemen worden gedemonstreerd. Aan de basis hiervoor ligt het netwerk over de sarcofaag-ziektes zoals die in de vorige paragraaf is gedefinieerd en getraind.

De trainingsset zoals die in figuur 6.1.b is gedefinieerd, wordt in het vervolg de 'Standaard TE-Set ' genoemd. Welke TE-Set wordt gebruikt wordt bij iedere statistiek aangegeven. Soms zijn bepaalde trainingsvoorbeelden vaker toegevoegd. Bijvoorbeeld 'Standaard TE-Set + 2*TE#2' , betekent dat de standaard Trainingsset is uitgebreid met twee TE#2's. De percentages in de statistieken zijn steeds gebaseerd op ongeveer 50 testen (de variabele #Monitor Cycles stond dus steeds ingesteld op 50).

Uit experimenten is gebleken dat een begintemperatuur TMax, zoals in subparagraaf 5.3.2.1 al is bewezen, gelijk aan twee een prima begin is. De activeringen verlopen dan niet te lang volledig random. Al vrij snel is het systeem dan bezig met het gericht zoeken naar de beste oplossing.

Een andere instelling die uiteindelijk toch het beste blijkt te werken is de clampinstelling. In subparagraaf 5.3.1 is dit aspect uitgelegd. Het komt er kortweg op neer dat bij clamping de zekere waarnemingen onaantastbaar worden overgenomen in de unit, terwijl onzekere waarnemingen via de perceptorunit de unit zelf adviseren.

Om het systeem te versnellen hadden we de variabele Maxstable voor handen. Deze is in de experimenten ingesteld op 10. Als het netwerk dus 10 volledige inferentiecycles (met één inferentiecycle wordt bedoeld dat alle units één keer hun waarde hebben berekend) niet van instelling is veranderd, stopt inferentiefase 1 en wordt het resultaat geaccepteerd.

Een instelling die waarschijnlijk het meest overeenkomt met de parallelle werkelijkheid is een random afhandeling van de units. De variabele Unitdispatcher is daarom ingesteld op RANDOM.

Verder moet er worden uitgezocht hoeveel afkoelingsstappen (NMax) we minimaal nodig hebben om het moeilijkste probleem aan te kunnen. Dit aantal wordt dan gebruikt voor alle hierna te bespreken problemen. Uit simulaties is gebleken dat weinig (maar wel meer dan één), onzekere waarnemingen, verspreid over ver van elkaar afgelegen units, de moeilijkste zijn voor SCORE. Een voorbeeld hiervan is de waarneming (U_1[1], U_8[0]). Omdat het aantal interne nodes bij U_8[0] twee is, krijgen beide nodes van de perceptor kans 0.5 toegediend. In feite hebben we nu dus te maken met een onzekere waarneming. Verder liggen U_1 en U_8 op afstand drie van elkaar. De resultaten, die in 6 tabellen zijn afgedrukt, waarbij het aantal afkoelingsstappen is gevarieerd van 100 tot 600, zijn in de tabellen 6.2.a t/m 6.2.f weergegeven.

Iedere tabel begint met een kopje met instellingen. Dit zijn welke TE-set het betreft, de waarde van NMax en de perception. De resultaten zijn vervolgens ingedeeld in drie kolommen. In de eerste kolom staat het nummer van de trainingexample (TE#), de tweede kolom bevat het percentage van optreden en de derde kolom geeft het argument waarom betreffend trainingexample al dan niet een correcte oplossing van het probleem is. Het kopje van de derde kolom geeft aan van welke units de geactiveerde nodes worden afgedrukt. Onder in de tabel is het uiteindelijke resultaat af te lezen. De asterisk (*) in de TE-kolom staat voor een illegale oplossing, d.w.z. een oplossing die niet in het lijstje trainingsvoorbeelden voorkomt. Hiermee kan worden gecontroleerd in hoeverre inferentiefase 2 de convergentie naar een legale oplossing naar tevredenheid uitvoert.

TE's = Standaard TE-set
NMax = 100
Perception = (U_1[1],U_8[0])

TE#	Percent.	(U_1,U_8)
1	10	(1,0) => Correct
2	6	(0,1) => Wrong
3	12	(0,1) => Wrong
4	38	(1,0) => Correct
5	6	(1,1) => Wrong
6	6	(1,1) => Wrong
7	22	(1,0) => Correct
8	0	(0,1) => Wrong
*	0	Illegal answer=> Wrong
RESULT	100	70% Correct ; 30% Wrong

Tabel 6. 2. a

TE's = Standaard TE-set
NMax = 200
Perception = (U_1[1],U_8[0])

TE#	Percent.	(U_1,U_8)
1	16	(1,0) => Correct
2	2	(0,1) => Wrong
3	4	(0,1) => Wrong
4	40	(1,0) => Correct
5	2	(1,1) => Wrong
6	12	(1,1) => Wrong
7	22	(1,0) => Correct
8	2	(0,1) => Wrong
*	0	Illegal answer=> Wrong
RESULT	100	78% Correct ; 22% Wrong

Tabel 6. 2. b

TE's = Standaard TE-set
NMax = 300
Perception = (U_1[1],U_8[0])

TE#	Percent.	(U_1,U_8)
1	32	(1,0) => Correct
2	0	(0,1) => Wrong
3	4	(0,1) => Wrong
4	30	(1,0) => Correct
5	4	(1,1) => Wrong
6	4	(1,1) => Wrong
7	24	(1,0) => Correct
8	2	(0,1) => Wrong
*	0	Illegal answer=> Wrong
RESULT	100	86% Correct ; 14% Wrong

Tabel 6.2.c

TE's = Standaard TE-set
NMax = 400
Perception = (U_1[1],U_8[0])

TE#	Percent.	(U_1,U_8)
1	20	(1,0) => Correct
2	0	(0,1) => Wrong
3	6	(0,1) => Wrong
4	30	(1,0) => Correct
5	18	(1,1) => Wrong
6	10	(1,1) => Wrong
7	14	(1,0) => Correct
8	2	(0,1) => Wrong
*	0	Illegal answer=> Wrong
RESULT	100	64% Correct ; 36% Wrong

Tabel 6.2.d

TE's	= Standaard TE-set	
NMax	= 500	
Perception	= (U_1[1],U_8[0])	
TE#	Percent.	(U_1,U_8)
1	24	(1,0) => Correct
2	0	(0,1) => Wrong
3	4	(0,1) => Wrong
4	30	(1,0) => Correct
5	2	(1,1) => Wrong
6	2	(1,1) => Wrong
7	36	(1,0) => Correct
8	2	(0,1) => Wrong
*	0	Illegal answer=> Wrong
RESULT	100	90% Correct ; 10% Wrong

Tabel 6.2.e

TE's	= Standaard TE-set	
NMax	= 600	
Perception	= (U_1[1],U_8[0])	
TE#	Percent.	(U_1,U_8)
1	14	(1,0) => Correct
2	0	(0,1) => Wrong
3	0	(0,1) => Wrong
4	56	(1,0) => Correct
5	2	(1,1) => Wrong
6	6	(1,1) => Wrong
7	22	(1,0) => Correct
8	0	(0,1) => Wrong
*	0	Illegal answer=> Wrong
RESULT	100	92% Correct ; 8% Wrong

Tabel 6.2.f

Een aantal zaken kunnen uit de tabellen worden opgemerkt. Ten eerste stijgt de performance bij toenemend aantal stappen, alleen NMax=400 vormt daarop een uitzondering. Bij NMax=100 is het foutpercentage nog 30%, terwijl dat bij NMax=600 is teruggedrongen tot

8%. Wat we ook kunnen zien is dat de TE#'s 3, 4 en 8 in zowel U_1 als U_8 van de waarneming afwijken. Bij NMax=100 worden deze TE's nog in 18% van de gevallen geconcludeerd, terwijl ze bij NMax=600 nooit meer worden geconcludeerd. Bij hoge NMax wordt dus niet alleen het foutpercentage kleiner, maar ook worden steeds minder conclusies getrokken die ver van de waarneming afliggen. Bij NMax=600 treden ook TE#4 en TE#5 nog een klein aantal keer op (8%). Deze TE's verschillen in alleen U_8 van de waarneming.

Verder heeft het systeem blijkbaar ook voorkeur voor bepaalde trainingsvoorbeelden. Zowel TE#1, TE#4 als TE#7 zijn correct, maar gemiddeld komt TE#4 er toch als favoriet uit. Om de voorkeuren van het netwerk te onderzoeken zijn de statistieken bepaald van inferenties zonder waarnemingen. Beschouw hiervoor tabel 6.2.g.

TE's = Standaard TE-set
NMax = 600
Perception = ()

TE#	Percent.
1	10
2	20
3	14
4	14
5	12
6	4
7	4
8	22
*	0
RESULT	100

Tabel 6.2.g

Er bestaan inderdaad voorkeuren voor bepaalde patronen. Hier krijgt TE#4 bijvoorbeeld ook de voorkeur boven TE#7 en TE#1. De grootste 'netwerkvoorkeur' heeft TE#8 gekregen. Het is wel aardig te zien dat TE#8 in de situatie van tabel 6.2.f helemaal verdwenen is. De voorkeuren zijn dus niet zo hard.

Een vergelijkbaar resultaat is te zien als we een uitgebreide waarneming doen. Zie tabel 6.2.h.

TE's = Standaard TE-set
NMax = 100
Perception = (U_1[0],U_2[0],U_3[0],U_4[1],U_5[1],U_6[0])

TE#	Percent.	(U_1,U_2,U_3,U_4,U_5,U_6)
1	0	(1,1,1,0,0,0) => Wrong
2	78	(0,0,0,1,1,0) => Correct
3	8	(0,0,1,1,0,1) => Wrong
4	2	(1,1,0,0,1,0) => Wrong
5	2	(1,0,0,1,1,1) => Wrong
6	4	(1,0,0,1,1,0) => Wrong
7	0	(1,1,1,0,0,1) => Wrong
8	6	(0,1,1,0,1,1) => Wrong
*	0	Illegal answer=> Wrong
RESULT	100	78% Correct ; 22% Wrong

Tabel 6. 2. h

Aan één kant is dit probleem makkelijker omdat de waarneming nu vollediger is dan die van de tabellen 6.2.a t/m 6.2.f, aan de andere kant wordt het moeilijker omdat er nu maar één correct antwoord bestaat, namelijk TE#2. De zelfde waarneming bij 600 stappen levert het resultaat als aangegeven in tabel 6.2.i.

TE's = Standaard TE-set
NMax = 600
Perception = (U_1[0],U_2[0],U_3[0],U_4[1],U_5[1],U_6[0])

TE#	Percent.	(U_1,U_2,U_3,U_4,U_5,U_6)
1	0	(1,1,1,0,0,0) => Wrong
2	96	(0,0,0,1,1,0) => Correct
3	0	(0,0,1,1,0,1) => Wrong
4	0	(1,1,0,0,1,0) => Wrong
5	4	(1,0,0,1,1,1) => Wrong
6	0	(1,0,0,1,1,0) => Wrong
7	0	(1,1,1,0,0,1) => Wrong
8	0	(0,1,1,0,1,1) => Wrong
*	0	Illegal answer=> Wrong
RESULT	100	96% Correct ; 4% Wrong

Tabel 6.2.i

Nu ziet het resultaat er zeer acceptabel uit. 600 stappen blijkt dus goede resultaten op te leveren. In de testen die volgen, wordt met dit aantal gewerkt hoewel sommige problemen al een prima resultaat zouden opleveren bij 100 stappen. Zie hiervoor tabel 6.2.j. Voor de duidelijkheid dient opgemerkt te worden dat stabiliteit bij een consistente waarneming al plaatsvindt rond T=1 ofwel de helft van het totaal aantal opgegeven stappen (zie voor de relatie tussen het aantal stappen en de temperatuur figuur 5.3.i). In de praktijk wordt bij Nmax=600 na ruim 300 stappen het stopcriterium bereikt.

TE's = Standaard TE-set
NMax = 600
Perception = (U_{11}[1])

TE#	Percent.	(U_{11})
1	64	(1) => Correct
2	0	(0) => Wrong
3	0	(0) => Wrong
4	0	(0) => Wrong
5	32	(1) => Correct
6	2	(0) => Wrong
7	2	(0) => Wrong
8	0	(0) => Wrong
*	0	Illegal answer=> Wrong
RESULT	100	96% Correct ; 4% Wrong

Tabel 6.2.j

Deze waarneming U_{11}[1] levert bij NMax=600 zelfs een score op van 100%. Een waarneming op U_{11} zou je nauwelijks nog een onvolledige waarneming kunnen noemen omdat voor elke TE een node is gereserveerd. Ondubbelzinnig worden nodes uit U_9 en U_{10}, bij gegeven nodeactivering in U_{11}, geactiveerd (de verbindingen van parents richting childs zijn immers altijd één). De geactiveerde nodes in U_9 en U_{10} activeren op hun beurt weer ondubbelzinnig nodes uit de units U_6, U_7 en U_8 die op hun beurt de units U_1 t/m U_5 éénduidig activeren.

De volgende vraag die gesteld kan worden is of een enkele waarneming op laag niveau ook zo goed doorwerkt naar boven toe. Stel bijvoorbeeld dat U_1[1] wordt waargenomen. Het verschil met de waarneming U_{11} [1] is dat U_{11}[1] wordt verdeeld over U_{11}-i[0] en U_{11}-i[4], terwijl een waarneming U_1[1] ten volle U_1-i[0] adviseert. Wat dat betreft zou de waarneming U_1[1] minstens even goed moeten gaan als U_{11}[1] . Het resultaat is weergegeven in tabel 6.2.k.

TE's = Standaard TE-set
NMax = 600
Perception = (U_1[1])

TE#	Percent.	(U_1)
1	6	(1) => Correct
2	0	(0) => Wrong
3	4	(0) => Wrong
4	32	(1) => Correct
5	26	(1) => Correct
6	14	(1) => Correct
7	16	(1) => Correct
8	2	(0) => Wrong
*	0	Illegal answer=> Wrong
RESULT	100	94% Correct ; 6% Wrong

Tabel 6.2.k

Tot zover de onvolledige, zekere (op de extern nodes) waarnemingen. Het hiernavolgende betreft onzekere waarnemingen, te beginnen met een onzekere waarneming over U_7. Allereerst wordt in tabel 6.2.l eerst aangegeven wat de resultaten zijn van een zekere waarneming over U_7[1], om daarna het resultaat van onzekere waarnemingen te kunnen gaan onderzoeken.

TE's = Standaard TE-set
NMax = 600
Perception = (U_7[1])

TE#	Percent.	(U_7)
1	13	(1) => Correct
2	6	(0) => Wrong
3	32	(1) => Correct
4	6	(0) => Wrong
5	9	(1) => Correct
6	19	(1) => Correct
7	9	(1) => Correct
8	6	(0) => Wrong
*	0	Illegal answer=> Wrong
RESULT	100	82% Correct ; 18% Wrong

Tabel 6.2.l

Nu hebben we vergelijkingsmateriaal voor de volgende testen, te beginnen bij de waarneming: $U_7[0]=0.25$, $U_7[1]=0.75$.

TE's = Standaard TE-set
NMax = 600
Perception = ($U_7[0]=0.25, U_7[1]=0.75$)

TE#	Percent.	(U_7)
1	6	(1) => Largest
2	6	(0) => Smallest
3	27	(1) => Largest
4	10	(0) => Smallest
5	23	(1) => Largest
6	6	(1) => Largest
7	10	(1) => Largest
8	12	(0) => Smallest
*	0	Illegal answer=> Wrong
RESULT	100	72% Largest;28% Smallest;0% Wrong

Tabel 6.2.m

Beschouw tabel 6.2.m. In deze tabel wordt per TE aangegeven of de waarde van U_7 overeenkomt met ofwel de grootste (largest) van de twee waarnemingen, ofwel met de kleinste (smallest). T.a.v. tabel 6.2.l is nu inderdaad het aantal oplossingen die $U_7[1]$ bevatten, omlaag gegaan en het aantal oplossingen die $U_7[0]$ bevatten, omhoog. Het is niet duidelijk hoe de relatie nu precies ligt tussen onzekere waarnemingen en de frequentie van optreden. Het enige dat we kunnen zeggen is dat $U_7[1]$ duidelijk vaker voorkomt dan $U_7[0]$. We mogen bijvoorbeeld niet de conclusie trekken dat $U_7[1]$ drie keer zo vaak voor komt als $U_7[0]$ omdat de waarneming $U_7[1]$ ook drie keer zo groot is als $U_7[0]$. Dat de largest wel ongeveer drie keer zo vaak wordt geconcludeerd, berust waarschijnlijk op toeval omdat in tabel 6.2.l ook al 18% op $U_7[0]$ viel, terwijl de waarneming op $U_7[0]$ nul was.

In tabel 6.2.n is nog wat meer met de kans van de waarneming gevarieerd.

TE's = Standaard TE-set NMax = 600 Perception = (U_7[0]=0.45,U_7[1]=0.55)		
TE#	Percent.	(U_7)
1	20	(1) => Largest
2	6	(0) => Smallest
3	4	(1) => Largest
4	10	(0) => Smallest
5	12	(1) => Largest
6	8	(1) => Largest
7	10	(1) => Largest
8	30	(0) => Smallest
*	0	Illegal answer=> Wrong
RESULT	100	54% Largest;46% Smallest;0% Wrong

Tabel 6.2.n

Hierin is te zien dat het percentage van conclusies dat U_7[1] (largest genoemd) bevat is gedaald van 72%, in tabel 6.2.m, naar 54% in deze tabel. In de traditionele AI geeft een onzekere waarneming een onzekere conclusie in termen van kansen of in termen van een mate van geloof o.i.d. De conclusie is dan een getal, bijvoorbeeld P(U_7[1])=0.54. In SCORE is de conclusie geen getal, maar een legale set van nodeactiveringen. Het in 54% van de gevallen (allen legale gevallen) optreden van U_7[1] impliceert een kans van 0.54 en niet andersom. De wereld dient zich ook niet aan ons aan in termen van kansen. In de wetenschap is het ook zo dat waarnemingen resulteren in statistieken waardoor wetmatigheden kunnen worden geconstateerd. De wereld dient zich voortdurend met kans één aan ons aan, alleen kan die verschijning iedere keer anders zijn. Dit kan vervolgens worden uitgedrukt in statistieken. Hoe onzeker de waarneming in SCORE dus ook is, het systeem zal altijd streven naar een afleiding die past in zijn wereldbeeld en aansluit bij zijn ervaring. Het zoeken naar zo'n afleiding is een indeterministisch proces (voor zover een randomgenerator voor echt indeterminisme kan zorgen). Daarom zal de ene keer afleiding A en de andere keer afleiding B worden gekozen bij dezelfde waarneming. Zo werkt dat bij de mens ook: een mens trekt ook geen conclusies in termen van kansen, maar kiest een zo goed mogelijke oplossing.

In de volgende test verhogen we de moeilijkheidsfactor van het probleem iets. De waarneming wordt weer P(U_7[1])=0.75 en P(U_7[0]) =0.25, maar nu breiden we de trainingsset uit met een trainingsvoorbeeld TE#2. Deze komt in totaal dan dus twee keer voor. Dit kan worden gedaan door, als de conclusie van een redenatieproces TE#2 is, bij de vraag ' Store

Result? (y/n) ' een 'y' in te toetsen. Het kennisacquisitie principe, zoals besproken in paragraaf 5.4 , wordt dan opgestart en de verbindingen krijgen nieuwe waarden.

TE's = Standaard TE-set + 1*TE#2
NMax = 600
Perception = ($U_7[0]$=0.25,$U_7[1]$=0.75)

TE#	Percent.	(U_7)
1	14	(1) => Largest
2	10	(0) => Smallest
3	28	(1) => Largest
4	12	(0) => Smallest
5	4	(1) => Largest
6	10	(1) => Largest
7	10	(1) => Largest
8	12	(0) => Smallest
*	0	Illegal answer=> Wrong
RESULT	100	66% Largest;34% Smallest;0% Wrong

Tabel 6.2.o

T.o.v. tabel 6.2.m is het vóórkomen van TE#2 met 4% toegenomen. Ook al is de kans P($U_7[0]$) slechts 0.25, als we voor het netwerk $U_7[0]$ bekender maken, zal het vaker optreden. Doordat $U_7[1]$ minder bekend is geworden is het totale percentage 'largest' ook gedaald. In tabel 6.2.p is de Standaard TE-Set met drie TE#2's uitgebreid.

TE's = Standaard TE-set + 3*TE#2
NMax = 600
Perception = ($U_7[0]$=0.25,$U_7[1]$=0.75)

TE#	Percent.	(U_7)
1	9	(1) => Largest
2	33	(0) => Smallest
3	20	(1) => Largest
4	9	(0) => Smallest
5	3	(1) => Largest
6	4	(1) => Largest
7	16	(1) => Largest
8	6	(0) => Smallest
*	0	Illegal answer=> Wrong
RESULT	100	52% Largest;48% Smallest;0% Wrong

Tabel 6.2.p

In deze tabel is te zien hoe zelfs een duidelijke voorkeur ontstaat voor TE#2, terwijl maar met kans 0.25 $U_7[0]$ is waargenomen. De voorkeur blijft nog net bestaan voor de TE's die voldoen aan de grootste waarneming (largest), maar die is absoluut minimaal. Voor de duidelijkheid nog één geval waarin TE#2 acht maal extra is toegevoegd.

TE's = Standaard TE-set + 8*TE#2
NMax = 600
Perception = ($U_7[0]$=0.25,$U_7[1]$=0.75)

TE#	Percent.	(U_7)
1	10	(1) => Largest
2	42	(0) => Smallest
3	19	(1) => Largest
4	4	(0) => Smallest
5	2	(1) => Largest
6	2	(1) => Largest
7	15	(1) => Largest
8	6	(0) => Smallest
*	0	Illegal answer=> Wrong
RESULT	100	48% Largest;52% Smallest;0% Wrong

Tabel 6.2.q

Nu ontstaat er zelfs de voorkeur voor een completering aan de waarneming met de kleinste kans (smallest) want van de 16 trainingsvoorbeelden zijn er 9 TE#2. Deze zit nu zo sterk in het wereldbeeld van het netwerk dat, ook al is de waarnemingskans klein, toch in een groot deel van de gevallen TE#2 wordt geconcludeerd.

We hebben nu de performance van onvolledige en onzekere waarnemingen bekeken. Wat nog rest is een onderzoek naar inconsistente waarnemingen. Nu gaat inferentiefase twee een rol spelen. Na inferentiefase 1 zal er, doordat een legale completering van een inconsistente waarneming onmogelijk is, een illegaal resultaat ontstaan. In inferentiefase 2, alwaar de waarneming wordt verwijderd en de temperatuur op TMin blijft hangen, worden nu waarnemingen aangepast. In de volgende statistieken worden de aangepaste waarnemingen gedemonstreerd.

Een inconsistente waarneming is bijvoorbeeld ($U_1[1]$, $U_2[0]$, $U_5[0]$): deze combinatie komt niet voor in het lijstje trainingsvoorbeelden. In tabel 5.2.r zijn de resultaten van de waarneming weergegeven. In de derde kolom is weer te zien hoe, per TE, de bijbehorende waarden van de units U_1, U_2 en U_5 zijn. Daarachter is aangegeven welke unit(s) een verkeerde waarde hebben.

Als na deze waarneming bijvoorbeeld TE#8 wordt geconcludeerd, dan zijn dus alle drie de waarnemingen aangepast.

TE's = Standaard TE-set
NMax = 600
Perception = (U_1[1],U_2[0],U_5[0])

TE#	Percent.	(U_1,U_2,U_5)
1	14	(1,1,0) => Difference: U_2
2	0	(0,0,1) => Difference: U_1,U_5
3	24	(0,0,0) => Difference: U_1
4	4	(1,1,1) => Difference: U_2,U_5
5	28	(1,0,1) => Difference: U_5
6	22	(1,0,1) => Difference: U_5
7	6	(1,1,0) => Difference: U_2
8	2	(0,1,1) => Difference: U_1,U_2,U_5
*	0	Illegal answer=> Wrong
RESULT	100	94% Diff.1 ; 4% Diff.2 ; 2% Diff.3

Tabel 6.2.r

Op de laatste regel van de tabel is aangegeven in hoeveel procent van de gevallen één (Diff. 1), twee (Diff. 2) , respectievelijk drie (Diff. 3) van de drie waarnemingen zijn aangepast. Het systeem heeft er dus duidelijk voor gezorgd dat zo veel mogelijk slechts één waarneming werd aangepast. In de meeste gevallen (50%) is U_5 aangepast. Dit heeft waarschijnlijk te maken met het feit dat U_1 en U_2 samen sterker staan dan U_5, omdat de eerste twee samen een grote invloed hebben op unit U_7, terwijl unit U_5 minder macht heeft over U_8.

TE#1 werd in 14% van de gevallen gekozen. Wat in het hiernavolgende wordt bekeken is wat de invloed op dit resultaat is als we TE#1 extra gaan leren. In tabel 6.2.s wordt TE#1 éénmaal extra geleerd, in tabel 6.2.t driemaal en in tabel 6.2.u, achtmaal.

TE's = Standaard TE-set + 1*TE#1
NMax = 600
Perception = (U_1[1],U_2[0],U_5[0])

TE#	Percent.	(U_1,U_2,U_5)
1	34	(1,1,0) => Difference: U_2
2	0	(0,0,1) => Difference: U_1,U_5
3	22	(0,0,0) => Difference: U_1
4	6	(1,1,1) => Difference: U_2,U_5
5	18	(1,0,1) => Difference: U_5
6	10	(1,0,1) => Difference: U_5
7	6	(1,1,0) => Difference: U_2
8	4	(0,1,1) => Difference: U_1,U_2,U_5
*	0	Illegal answer=> Wrong
RESULT	100	90% Diff.1 ; 6% Diff.2 ; 4% Diff.3

Tabel 6.2.s

Het resultaat van TE#1 is flink omhoog gegaan en over de gehele linie is er wat ingeleverd. Het systeem convergeert dus inderdaad naar dié conclusie die voor hem het bekendst is. Het heeft duidelijk voorkeur voor bekende patronen. Dit is ook precies wat we willen. In de volgende twee tabellen wordt gedemonstreerd dat de voorkeur voor TE#1 nog verder toeneemt als er nog vaker TE#1 's worden geleerd.

TE's = Standaard TE-set + 3*TE#1
NMax = 600
Perception = (U_1[1],U_2[0],U_5[0])

TE#	Percent.	(U_1,U_2,U_5)
1	54	(1,1,0) => Difference: U_2
2	0	(0,0,1) => Difference: U_1,U_5
3	18	(0,0,0) => Difference: U_1
4	2	(1,1,1) => Difference: U_2,U_5
5	18	(1,0,1) => Difference: U_5
6	8	(1,0,1) => Difference: U_5
7	0	(1,1,0) => Difference: U_2
8	0	(0,1,1) => Difference: U_1,U_2,U_5
*	0	Illegal answer=> Wrong
RESULT	100	98% Diff.1 ; 2% Diff.2 ; 0% Diff.3

Tabel 6.2.t

TE's = Standaard TE-set + 8*TE#1
NMax = 600
Perception = (U_1[1],U_2[0],U_5[0])

TE#	Percent.	(U_1,U_2,U_5)
1	70	(1,1,0) => Difference: U_2
2	2	(0,0,1) => Difference: U_1,U_5
3	10	(0,0,0) => Difference: U_1
4	2	(1,1,1) => Difference: U_2,U_5
5	10	(1,0,1) => Difference: U_5
6	2	(1,0,1) => Difference: U_5
7	0	(1,1,0) => Difference: U_2
8	4	(0,1,1) => Difference: U_1,U_2,U_5
*	0	Illegal answer=> Wrong
RESULT	100	92% Diff.1 ; 4% Diff.2 ; 4% Diff.3

Tabel 6.2.u

Het percentage van optreden van TE#1 neemt dus steeds meer toe en de rest zwakt langzaam af. Doordat TE#1 in het wereldbeeld van dit netwerk steeds bekender wordt zullen andere patronen langzaam minder bekend worden.

In de volgende test wordt aangetoond dat de structuureisen aan het netwerk zeer flexibel zijn. De structuur zoals die in de voorgaande voorbeelden is gedefinieerd, is domweg van Gallant overgenomen, maar had ook heel anders kunnen zijn. Doordat SCORE-S zelf de omkeerbaarheidsfouten herstelt, zal er altijd een consistent netwerk ontstaan. Daarvoor hoeven we dus niet te laten een ander netwerk te proberen. De beste structuur is ongetwijfeld dié structuur waarbij alles met alles is verbonden ('fully interconnected network'). Dit legt echter zeer zware eisen op aan de hardware. Het doel van een netwerkstructuur is het aantal relaties, dus het aantal verbindingen, zoveel mogelijk te beperken. Indien de netwerkstructuur random wordt gegenereerd, is de kans groot dat er vaak omkeerbaarheidseisen worden geschonden waardoor extra nodes en hidden units moeten worden geïntroduceerd die alsnog de hardware extra belasten. Toch hoeft een random structuur niet ten koste te gaan van de performance. In de volgende test is een netwerk vrij willekeurig gedefinieerd waarin dit wordt aangetoond. Deze is afgedrukt in figuur 6.2.a.

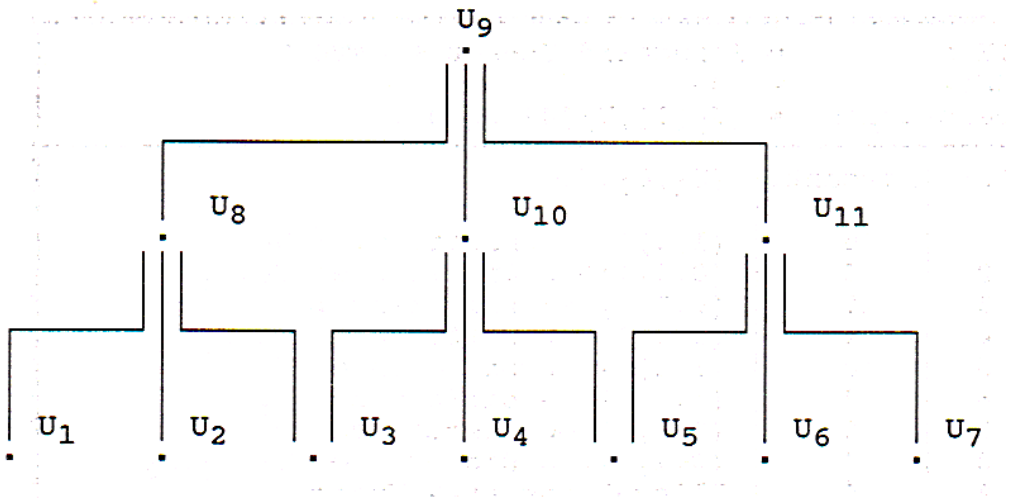


Fig. 6.2.a. Een 'willekeurige' netwerkstructuur

Het netwerk is in die zin willekeurig opgesteld, dat er wel voor gezorgd is dat er geen hidden units nodig waren. De unitnummers zoals die in de figuur zijn vermeld komen weer overeen met de unitnummers uit figuur 2.5.a. Een eis die gesteld was aan de netwerkdefinitie was het feit dat units niet afhankelijk mochten zijn van units met een hogere index. In figuur 6.2.a is dat wel zo omdat U_9 afhankelijk is van U_{10} en U_{11}. Daardoor heeft bij de test U_9 (Posiboost), de naam U_{11} gekregen en U_{11} (Placibin), de naam U_9. Dit is een puur technisch probleempje dat niets met de resultaten, die hierna worden afgedrukt, te maken heeft.

De unit met behandelingsmethoden (U_{11}), oorspronkelijk de unit op het hoogste niveau, is nu op het tweede niveau opgenomen en een ziekteunit (U_9) zit nu helemaal bovenaan in het netwerk. De assertie U_9=U_8, U_{10}, U_{11} heeft hierdoor niets meer met semantiek te maken, waardoor ieder trainingsvoorbeeld een grote kans heeft de introductie van nieuwe extra units en hidden units tot gevolg te hebben. In plaats van vier niveaus zijn er nu ook maar drie niveaus.

De test is uitgevoerd op het allereerste probleem dat in deze paragraaf is besproken, namelijk de waarneming (U_1[1], U_8[0]), NMax=100. Beschouw hiervoor nogmaals tabel 6.2.a. In tabel 6.2.v zijn de resultaten van de zelfde test op het netwerk van figuur 6.2.a opgenomen.

TE's = Standaard TE-set
NMax = 100
Perception = (U_1[1],U_8[0])

TE#	Percent.	(U_1,U_8)
1	26	(1,0) => Correct
2	0	(0,1) => Wrong
3	0	(0,1) => Wrong
4	44	(1,0) => Correct
5	0	(1,1) => Wrong
6	0	(1,1) => Wrong
7	22	(1,0) => Correct
8	2	(0,1) => Wrong
*	6	Illegal answer=> Wrong
RESULT	100	92% Correct ; 8% Wrong

Tabel 6.2.v

Tabel 6.2.v. demonstreert een verrassend resultaat. Ten eerste moet worden opgemerkt dat het percentage correcte antwoorden prima is. Ook al wordt de structuur door elkaar gehusseld, het geheel blijft dus prima werken. SCORE is duidelijk niet gevoelig voor structuurfouten.

Ten tweede valt op dat we voor het eerst 'illegal answers' hebben. In de voorgaande voorbeelden waren we die nog niet tegengekomen. Blijkbaar is inferentiefase 2 niet afdoende voor het verkrijgen van een legale oplossing. Waarschijnlijk heeft dit te maken met de ligging van de units waarop werd waargenomen: U_1 en U_8 zijn buren van elkaar. Als de waarneming in inferentiefase 2 namelijk wordt losgekoppeld, kunnen door de sequentiële unitafhandeling de foute instellingen op U_1 en U_8 elkaar in principe in stand houden. Een oplossing zou kunnen zijn dat inferentiefase 2 niet meteen eindigt als de unitinstellingen twee cycli constant blijven, maar dat de units méér tijd krijgen om van waarde te veranderen. Dit aspect verdient nader onderzoek.

7 Conclusies en Aanbevelingen voor verder Werk

In hoofdstuk twee is het Connectionistisch AI-Paradigma tegenover het traditioneel AI-Paradigma gezet. Een combinatie van beide paradigma 's heeft geleid tot SCORE. Door de connectionistische AI-principes is er een systeem ontstaan dat in staat is tot redeneren met onzekerheid, onvolledigheid en inconsistenties in de waarnemingen. Door de verwerking van traditionele AI-principes vindt dit redeneren plaats op basis van een zeker, volledig en consistent netwerk. Het connectionistisch redeneren is doorgevoerd tot op het niveau van de units. Daar stuiten we op de unitbesturing alwaar het connectionistisch redeneren plaats maakt voor traditioneel redeneren.

Door de traditionele benadering van de units is er ruimte ontstaan voor inferentieheuristieken. Hierdoor ontstaat toch nog de mogelijkheid om d.m.v. regels de inferentieperformance te verbeteren, hoewel we toch moeten blijven streven naar een volledig connectionistisch systeem. Want juist die unitbesturing is door zijn sequentiële karakter toch de zwakste schakel: als de unitbesturing faalt, faalt ook de bijbehorende eigenschap. Doordat de inferentie op het niveau van het complete netwerk wel connectionistisch is, zal bij een falende unit alleen betreffende eigenschap er de dupe van zijn en niet de afleiding als geheel. Dit is een belangrijk voordeel t.o.v. het traditionele AI-Paradigma.

Het nadeel van het expliciet aanwezig zijn van eigenschappen, namelijk de foutgevoeligheid op eigenschap niveau, heeft aan de andere kant ook duidelijke voordelen, zeker in het geval van simulaties waar dergelijke nadelen nog geen rol spelen. Doordat de sterktes van de verbindingen in het netwerk volledig zijn gebaseerd op kanstheorie is de kennis, in termen van relaties tussen instanties van eigenschappen, expliciet voorhanden. Daardoor kunnen de redenen van de conclusies na een inferentieproces altijd aan een onderzoek worden onderworpen. Het systeem zal geen onduidelijk eigen leven gaan leiden. Dit is een belangrijk voordeel t.o.v. het Connectionist AI-Paradigma. Aldaar zit de onduidelijk vooral in de hidden-layers. Hoewel SCORE ook gebruik maakt van extra nodes en hidden units, hebben deze toch betekenis, waardoor ook de relaties met hen betekenis hebben.

Een ander voordeel van het feit dat eigenschappen expliciet aanwezig zijn is het feit dat daarmee subconclusies ook expliciet aanwezig zijn. Als we denken aan het Sarcofaag-probleem uit het vorige hoofdstuk, is het een voordeel dat niet alleen symptomen (input) en behandelingsmethoden (output) voor handen zijn, maar ook de ziektes. Doordat alle units technisch hetzelfde zijn, is er dus ook geen afgebakende input-output layer en kunnen er zelfs waarnemingen worden gepleegd op subconclusieniveau: het redeneren is een combinatie van bottom-up en top-down waardoor waarnemen en concludering op ieder niveau kan plaatsvinden.

Zoals in de simulaties te zien was, is er een duidelijk verband tussen de bekendheid van een patroon en de frequentie waarmee het na een waarneming werd geconcludeerd. Zo was er ook een duidelijk verband tussen de kans waarmee iets wordt waargenomen en de frequentie van optreden van bijbehorende patronen. Hoe dat verband nu precies ligt is niet precies duidelijk. Dit is een kenmerk van Connectionist Systems: het verband ontstaat en wordt niet geprogrammeerd. Dit is op zich heel prettig. Als we vinden dat een bepaald patroon te weinig wordt geconcludeerd, trainen we het netwerk eenvoudigweg nogmaals met betreffend patroon. Om met Connectionist Systems in het algemeen en met SCORE in het bijzonder verder te kunnen is het waarschijnlijk toch belangrijk te weten hoe dit verband nu precies ligt. Een onduidelijke stap is immers een laatste stap.

Uit de simulaties is ook gebleken dat er voorkeur bestaat voor bepaalde units (een waarneming op U_{11} gaf een beter resultaat dan op U_7). Hiervoor zijn in het simulatie-hoofdstuk al redenen geopperd. Toch kan dat aspect verder worden uitgezocht. Het zou misschien beter zijn als de units wat meer worden uitgebalanceerd. Hiermee in verband staat het volgende. In subparagraaf 5.3.2.1 is beschreven dat de relatie tussen de totale netwerk-energie (E) en de afzonderlijke unit-energie (E_U) onduidelijk is. Om dit te kunnen opstellen is het ook van belang te weten hoe de units zich t.o.v. elkaar verhouden. Als het verband tussen E en E_U is gelegd kan onmiddellijk worden bewezen wat, in termen van kansen, het optimaliseringscriterium van het systeem is. We weten namelijk dat d.m.v. Simulated Annealing één of andere E wordt geminimaliseerd. Als E beschreven kan worden in termen van E_U, weten we ook wat er in termen van $I(\alpha)$, het kansadvies van alle units aan node α, wordt geminimaliseerd (zie hiervoor formule 5.3.a).

Een voordeel van SCORE is het feit dat de netwerkstructuur eigenlijk niet zo verschrikkelijk belangrijk is. Omdat automatisch omkeerbaarheidsfouten worden hersteld, voldoen in principe alle structuren met alle mogelijke trainingsvoorbeelden. Een volgende stap voor SCORE zou een automatische netwerkstructuur-generator kunnen zijn. Er zou dan onderzoek gedaan moeten worden naar welke structuren gemiddeld een goede performance leveren en om niet te veel unitverbindingen vragen (bijvoorbeeld gebalanceerde boomstructuren?).

Er is met het in dit rapport beschreven onderzoek duidelijk aangetoond dat Connectionist Systems toekomst kunnen hebben m. b. t. expert systemen. Vervolgonderzoek is zonder meer interessant. Er bieden zich op dit moment echter nog meer AI-inzichten aan. Connectionist Systems of Neurale Netwerken zijn zeker niet de enige weg naar werkelijke Artificiële Intelligentie. Een andere belangrijke AI-ontwikkeling is die van het 'chaos'-onderzoek. Kortgezegd komt het erop neer dat systemen kunnen ontstaan vanuit een chaos van losse autonome componenten die op een gegeven moment met elkaar gaan samenwerken. Dat zou een mooie ingang zijn voor het kennisstructureringsprobleem. Ook binnen de taalfilosofie wordt veel onderzoek naar intelligentie gedaan: taal is immers altijd nog een belangrijke maatstaf voor intelligentie. Ook resultaten van onderzoek naar genetisch materiaal, het

substraat voor intelligentie, worden al geschikt gemaakt voor computers. Allen hebben hun eigen sterke en zwakke kanten. Het toekomstig AI-onderzoek zal zich waarschijnlijk kenmerken door een gebundelde kracht van vele, en niet alleen technische, onderzoeksdisciplines.

Bijlage I De Harmony Theorie

De Harmony Theorie is begin jaren '80 ontwikkeld door P. Smolensky, lid van de 'PDP Research Group'. Deze theorie wordt uitvoerig behandeld in [Smolensky, 1986]. In deze bijlage behandel ik de belangrijkste facetten ervan.

De Harmony Theorie is een wiskundige theorie. Het heeft een aantal parallellen met theorieën uit de fysica, in het bijzonder de thermodynamica. Veel van de binnen dat vakgebied opgedane kennis blijkt bij zonder bruikbaar voor theorieën over Connectionist-Systems.

De Harmony Theorie is ontwikkeld voor het oplossen van 'Completion Tasks'. Dit houdt in dat slechts een deel van een patroon, woord, zin, o.i.d., wordt waargenomen en dat het Connectionist-System, op grond van zijn kennis, het niet bekende of niet zichtbare deel van de waarneming completeert. Een Harmony netwerk bestaat uit twee lagen nodes. Zie figuur I. a.

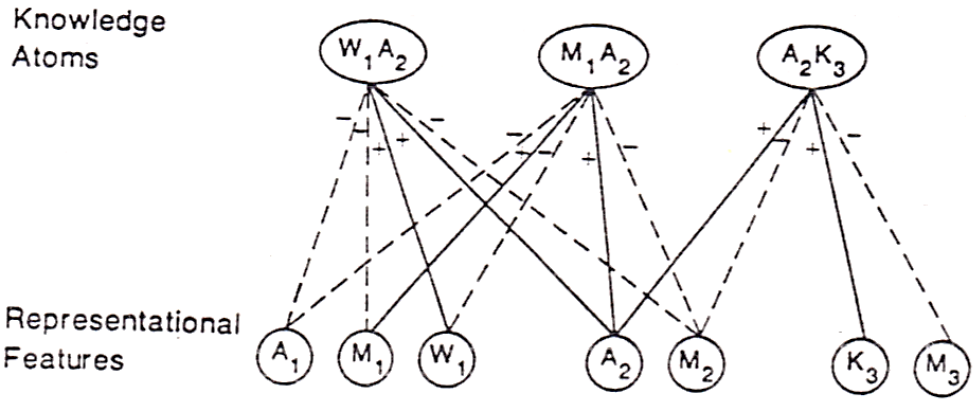


Fig. I.a De grafische representatie van een specifiek Harmonymodel. In dit model staan de 'representational features' voor letters op bepaalde posities, terwijl de 'knowledge atoms' de bekende lettercombinaties representeren.

De onderste laag bestaat uit 'representational features', waarvan de activationvalues +1 en -1 kunnen zijn. De bovenste laag bestaat uit 'knowledge atoms' met mogelijke activationvalues 0 en 1. Bij de representational features betekent -1 dat de betreffende eigenschap afwezig is en + 1 dat het aanwezig is. Bij de knowledge atoms betekent een 0, inactief en een 1, actief.

Hoe lost een Harmony netwerk het completion task probleem nu op? Welnu, de nodes op het laagste niveau, de representational features, staan voor de eigenschappen van de

waarneming in de buitenwereld. De knowledge atoms staan voor coderingen van de waarnemingen. Beschouw nogmaals figuur I.a. Dit systeem kent slechts drie-letterige woorden. Het heeft als eerste letters ooit A, M en W, A en M als tweede letter en K en M als derde letter, waargenomen. De knowledge atoms geven een drietal coderingen aan van de binnen het systeem bekende woorden. In dit geval: WA-, MA en -AK. In principe zijn nu dus alleen de woorden WAK en MAK bekend. De codering vindt plaats door het leggen van verbindingen van betreffende knowledge atoms naar de van belang zijnde representational features. De vector k_α staat voor deze codering. Zie hiervoor nogmaals figuur I.a en aanschouw bijvoorbeeld k_{w1A2}.

Stel nu dat slechts een W op de eerste positie van het 3-letterige woord wordt waargenomen. Feature node W1 krijgt de waarde 1, A1 en M1 de waarde 0. De overige featurenodes worden onbepaald. De completion task is de taak aan het netwerk opgelegd, ook de tweede en derde letter te bepalen. Een waarneming resulteert in een gedeeltelijke invulling van de waarden van de representational features. Nu moet dié aanvulling worden gevonden die, gegeven de kennis in het netwerk, de grootste kans heeft. Als de eerste letter een W is en het netwerk kent het woord 'WAK', welke in 40% van de gevallen werd waargenomen, en het woord 'WAM' , welke in 60% van de gevallen werd waargenomen, dan heeft 'WAM' dus de grootste kans. Nu blijkt er een relatie te bestaan tussen de 'Harmony ' van de oplossing (de completion) en de kans op die oplossing. Die relatie ziet er als volgt uit:

Probability ≈ EXP(H/T)

Hierin is H de Harmony van de oplossing en T de temperatuur waar we het later over zullen hebben. Hoe hoger de Harmony van de oplossing, hoe hoger de kans op die oplossing. De invloed van H wordt afgezwakt door T voor T>1 en versterkt voor 0<T<1. Deze relatie is afgeleid van de Gibbs en Boltzmann wet uit de fysica. De Harmonyfunctie H luidt (met vectoren dikgedrukt):

$$H_\beta(r,a) = \sum_\alpha \sigma_\alpha * a_\alpha * h_\beta(r,k_\alpha) \quad \text{met}$$

$$h_\beta(r,k_\alpha) = (r \cdot k_\alpha)/|k_\alpha| - \beta \quad \text{met}$$

$$r \cdot k_\alpha = \sum_i r_i * (k_\alpha)_i \quad \text{en}$$

$$|k_\alpha| = \sum_i |(k_\alpha)_i|$$

Wat staat hier nu precies? Knowledge atoms worden aangegeven door de index α, representational features door de index i. a_α is de activationvalue (1 of 0) van atom a en r_i is de waarde (+1 of -1) van feature i. σ_α is de 'sterkte' van atom α, hetgeen een maat is voor de frequentie waarmee de bij atom α behorende code is waargenomen. σ_α moet dus d.m.v. één

of andere learning rule na elke waarneming worden aangepast. σ_α is symbolisch aangegeven in atom α, maar wordt geïmplementeerd in de verbinding naar atom α door σ_α gelijkelijk te verdelen over die verbindingen. $h_\beta(\mathbf{r}, \mathbf{k\alpha})$ is het aandeel in de totale Harmony veroorzaakt door atom α. $h_\beta(\mathbf{r}, \mathbf{a})$is de Harmony van het netwerk met gegeven actvationvalue patroon **a** over de knowledge atoms en oplossing **r** over de representational features. Deze Harmony H is dus de gewogen som van alle afzonderlijke Harmonies over alle actieve atomen α. $h_\beta(\mathbf{r}, \mathbf{k\alpha})$ is een maat in hoeverre de codering, waar atom α voor staat, overeenkomt (in Harmony is !) met de oplossing **r**. β ligt in het interval (-1, +1) . Als β=0, dan wordt de Harmony verhoogd door het activeren van atom α, indien 50% of meer van de elementen van k_α overeenstemmen met **r**. Als β=+1, dan wordt het altijd gunstig atom α te activeren. Als β=-1, dan heeft de activering van atom α altijd een negatief effect op de Harmony. Met β kan dus de gevoeligheid van atom α op de input **r** worden geregeld.

Nogmaals, we willen dié toestand **r** bereiken, die de grootste kans heeft en doen dit door naar die toestand met een zo groot mogelijke Harmony te zoeken. De nodes van het netwerk moeten dus tijdens de completion task voortdurend beslissen hoé die Harmony H te verhogen. Voor het knowledge atom is dat de beslissing +1 of 0 te worden, voor de representational feature -1 of + 1. Overigens vindt deze completion task plaats d.m.v. een combinatie van parallellisme en sequentie. Als eerste bepalen alle atoms parallel hun waarde, vervolgens alle features, daarna weer de atoms, etc. Als we de input definiëren als hierna is aangegeven krijgt ieder atom precies het verschil in Harmony binnen tussen de situatie waarin het actief zou zijn en de situatie waarin het inactief zou zijn. De input van een atom α wordt aangegeven door I_α. De input I_i voor de feature nodes is natuurlijk het verschil tussen de situaties waarin het de waarde +1 en -1 zou aannemen.

$$I_i = 2 * \sum_i W_{i\alpha} * a_\alpha \quad \text{en} \quad I_\alpha = \sum_i W_{i\alpha} * r_i - \sigma_\alpha * \beta$$
$$\text{met } W_{i\alpha} = (k_\alpha)_i * (\sigma_\alpha / |k_\alpha|)$$

$W_{i\alpha}$ is de sterkte van de verbinding tussen feature i en atom α. Dat d.m.v. deze inputdefinitie inderdaad het verschil in Harmony bij ieder neuron binnenkomt is eenvoudig te bewijzen, uitgaande van de Harmony functie. Dit zal hier niet verder worden uitgewerkt.

Tot zover de definitie van het gebruik van de Harmony functie. Ik wil even een uitstapje maken naar de fysica om daarna de Harmony functie te herinterpreteren. Beschouw de situatie van figuur I.b.

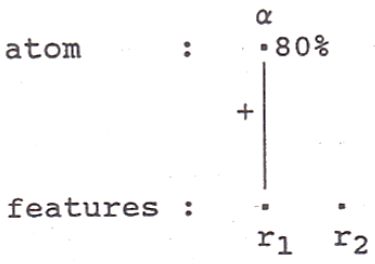


Fig. I.b Een zeer eenvoudig Harmony netwerk.

Er zijn twee features r_1 en r_2 en één atom α met σ_α=80%, hetgeen inhoudt dat in 80% van de waarnemingen **r**, r_1=+ was. Deze 80% slaat op een deel van **r**. Als nu r_1=+ wordt waargenomen, hoe moet de waarde van r_2 nu bepaald worden? Uit het schema is niets meer te halen dan $P(r_1=+, r_2=+) + P(r_1=+, r_2=-) = 0.8$ (P staat voor kans). Dit is één vergelijking met twee onbekenden en heeft dus oneindig veel oplossingen. C.E. Shannon (1963) geeft ons een methode om een voorkeursverdeling van al die mogelijkheden te maken. De voorkeur van de ene oplossing voor de andere zit hem daarbij in de hoeveelheid 'missing information'. Shannon stelt dat er niet meer informatie uit het systeem gehaald kan worden dan erin zit. Dit is volgens hem te realiseren door het streven naar een zo homogeen mogelijke kansverdeling: hoe homogener de verdeling, hoe onzekerder de kennis en hoe minder er ten onrechte informatie aan wordt toegevoegd. Shannon's formule voor de hoeveelheid 'missing information' van een kansverdeling P is :

$$-\sum_x P(x) * LN\,(P(x))$$

Twee oplossingen van bovenstaand voorbeeld zijn bijvoorbeeld:

$$P(r_1=+,r_2=+) = 0.7 \text{ en } P(r_1=+,r_2=-) = 0.1 \text{ of}$$
$$P(r_1=+,r_2=+) = 0.4 \text{ en } P(r_1=+,r_2=-) = 0.4$$

Shanon's formule hierop toegepast levert:

$$-[.7 * LN(.7) + .1 * LN(.1)] = .48 \quad \text{en}$$
$$-[.4 * LN(.4) + .4 * LN(.4)] = .73$$

De keuze $P(r_1=+, r_2=+) = 0.4$ en $P(r_1=+, r_2=-) = 0.4$ is de meest homogene en voegt inderdaad niets toe aan kennis over r_2.

Figuur I.b is nu te interpreteren als figuur I.c.

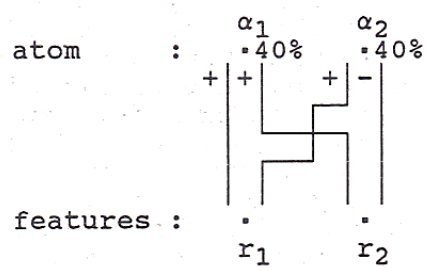


Fig. I.c Interpretatie van figuur I.a m.b.t. $P(r_1=+, r_2=+) + P(r_1=+, r_2=-) = 0.8$

Behalve P(r1=+) = 0.8, weten we natuurlijk ook dat P(r1=-)=0.2. Hetzelfde verhaal m.b.t. de 'most missing information' toegepast hierop levert nog twee extra atoms α_3 en α_4 op. Het resultaat is te zien in figuur I.d.

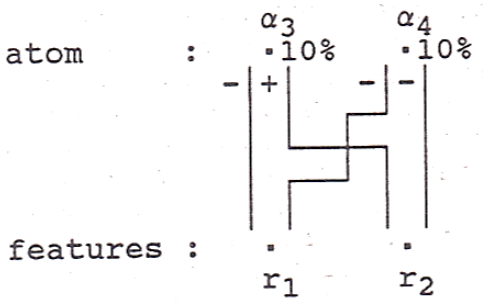


Fig. I.d Interpretatie van figuur I.a m.b.t. $P(r_1=-, r_2=+) + P(r_1=-, r_2=-) = 0.2$

Als nu de waarneming 'r_2=+' plaatsvindt, dan worden de atomen α_1 en α_3 actief. α_1 werd in 40% van de gevallen, en α_3 in 10% van de gevallen actief. De completion vindt dus plaats via α_1 en r_i wordt '+'. Gegeven een incomplete waarneming over **r** (in bovenstaand voorbeeld was dat r_2) . Er ontstaat dan dus, op basis van Shannon 's most missing information theorema , een kansverdeling over alle mogelijke combinaties van alle features. Deze verdeling wordt $\pi(\mathbf{r})$ genoemd. Er kan nu worden bewezen dat het zoeken naar de beste completion, rekening houdend met Shannon, kan geschieden door het zoeken naar dié completion met de hoogste Harmony H(**r**,a). Deze verdeling wordt dan niet meer $\pi(\mathbf{r})$ genoemd, maar P(**r**, a).

De begrippen 'missing information' en 'Harmony' hebben een analogie met de natuurkunde. Daar heten de begrippen respectievelijk 'entropie' en 'energie'. De tweede hoofdwet van de thermodynamica stelt dat als fysische systemen zich in de tijd ontwikkelen, het streeft naar maximalisering van chaos of entropie. Een hoge entropie of een grote chaos heeft inderdaad ook veel missing information: de definitie van een chaotische situatie is juist dat er niets valt te zeggen over die situatie, kortom het bevat een minimale hoeveelheid informatie. Het streven naar die hoge entropie, die grote hoeveelheid missing information, vindt plaats door

te zoeken naar het lage energieniveau ofwel de hoge Harmony. Het is daarom niet verwonderlijk dat het completion proces formele overeenkomsten zou moeten vertonen met natuurkundige theorieën. In het hiernavolgende wordt een uit de fysica overgenomen methode beschreven voor het ontdekken van dat laagste energieniveau of hoogste Harmony.

Als we besluiten puur op basis van voorgaande de completion task uit te voeren, blijkt de kans groot dat we niet in een globaal maar lokaal Harmony maximum terecht komen. Dit komt doordat direct naar het dichtstbij zijnde energiedal wordt gestapt. In tekening is dit te zien in figuur I.e.

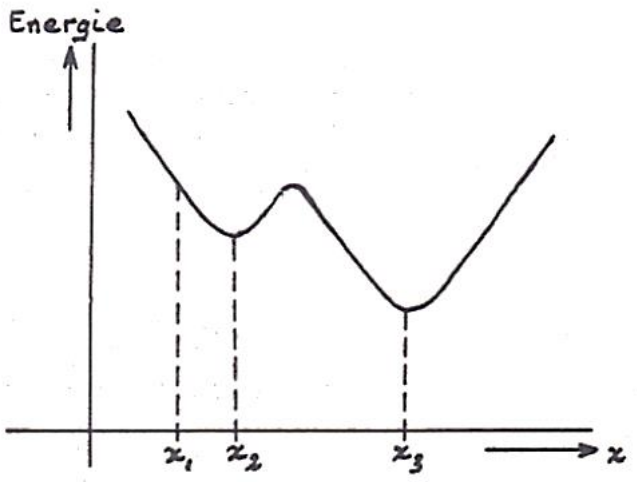


Fig. I.e Het energielandschap van de variabele x

Indien x bij aanvang de waarde x_1 heeft, zal na een aantal stappen de waarde x_2 worden bereikt. Dit zal ook het eindpunt zijn omdat meteen rechts van x_2 de energie weer zal toenemen. Binnen de Harmony theorie is de volgende oplossing gevonden voor het toch bereiken van het globale minimum x_3.

Niet altijd, maar slechts met een bepaalde kans wordt voor dié oplossing gekozen die een hogere Harmony representeert. Dit geschiedt d.m.v. de volgende formule:

P(Value=1) = 1 / (1 + EXP (-I/T))

P staat voor kans, I voor de input, zoals hiervoor gedefinieerd voor betreffende node, en T voor de temperatuur. De grafiek van deze ' update functie ' ziet er uit als in figuur I.f.

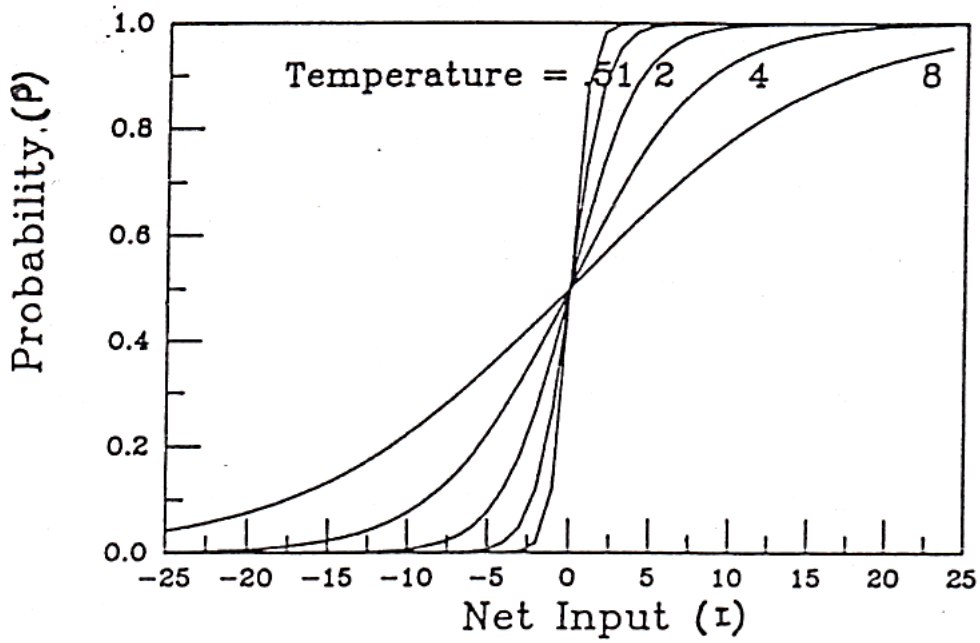


Fig. I.f Activeringskansen als functie van de netinput I en de temperatuur T.

Voor grote waarden van T loopt de grafiek nagenoeg horizontaal, waardoor de input I niet of nauwelijks van invloed is op de keuze de node al dan niet te activeren. Het activeringsproces vindt dan random plaats. Bij lagere T zal de invloed van I op de activeringsbeslissing toenemen. Als T=0, dan wordt de node met kans één geactiveerd als I>0 en gedeactiveerd als I<0. De onzekerheid is dan geëlimineerd.

Wat is het effect van dit onzekerheids-element in het zoeken naar een globaal energieminimum? Beschouw nogmaals figuur I.e. We beginnen weer in x_1. Doordat de update probabilistisch is, hoeft het niet noodzakelijkerwijs naar beneden af te dalen. Met een bepaalde kans kan het ook omhoog. Bij grote T is het stijgen zelfs even waarschijnlijk als het dalen. Startend in x_1 is het dus niet automatisch zo dat we in het lokale minimum x_2 terecht komen. Als we oneindig langzaam afkoelen wordt met kans één het globale optimum x_3 gevonden.

Bijlage II Het Pocket Algoritme

The Pocket Algorithm: A Procedure that Generates Weights for Discrete Networks

Although a comprehensive treatment of learning in connectionist models lies beyond the scope of this article, we present a quick overview of the learning algorithms used in our system.

It is important that the training examples specify the desired activations for intermediate and output cells in the network (easy learning). This allows us to decompose the problem and consider each cell independently in terms of training example inputs, desired cell activations, and weights to be generated. Therefore, we can drop the subscript i and refer only to u and w_u to state the learning algorithm for a single cell. This makes weight computations much simpler.

To compute the vector of weights w_u for intermediate or output cell u, we first set $w_j = 0$ for every connection from a cell that is not in u's dependency list. We ignore these weights for the remainder of the computation. For connections from the remaining cells (that are in u's dependency list), we use the following procedure to compute the rest of w_u: For cell u let $\{E^1\}$ be the set of training example activations, and $\{C^1\}$ the corresponding correct activations for u. For simplicity we assume Boolean activations so that each C^1 takes on values $\{+1, -1\}$ for $\{True, False\}$, respectively, while each component of E^1 can take on values of $\{+1, -1, 0\}$ for $\{True, False, Unknown\}$. $E_0^1 = +1$ so that w_0 will be the computed bias for the cell in question.

The basic learning algorithm for generating weights is a modification of perceptron learning [33] called the pocket algorithm [13]. It computes perceptron weight vectors, P, which occasionally replace pocket weight vectors, w_u, as follows:

(1) Set P to the 0 vector.

(2) Let P be the current perceptron weights. Randomly pick a training example E^1 (with corresponding classification C^1).

(3a) If P correctly classifies E^1, that is,

$$\{P \cdot E^1 > 0 \quad \text{and} \quad C^1 = +1\}$$

or

$$\{P \cdot E^1 < 0 \quad \text{and} \quad C^1 = -1\},$$

then,

(3aa) if the current run of correct classifications with P is longer than the run of correct classifications for the weight vector w_u in your pocket,

(3aaa) replace the pocket weights w_u by P, and remember the length of its correct run.[*]

(3b) Otherwise, form a new set of weights P' as follows:

$$P' = P + C^1E^1.$$

(4) Go to (2).

Table I illustrates the algorithm for several iteration steps.

A drawback to the pocket algorithm is that there is no known bound on the number of iterations required to achieve a fixed probability that the pocketed weights are the best possible. Nevertheless, if there are not too many training examples ($<10^5$), it is relatively easy to periodically check the pocketed weights against the set of all training examples in order to evaluate their performance.

An important advantage of the pocket algorithm over perceptron learning is that it works well with nonseparable or even contradictory training examples.

RATCHETS

Whenever there is a fixed set of training examples, we have found it very useful to modify (3aa) above to include a *ratchet*:

(3aa)′ If the current run of correct classifications with P is longer than the run of correct classifications for the weight vector w_u in your pocket *and P correctly classifies more training examples than w_u*, ...

Thus, we check a potential new w_u to see if it is really an improvement before making it the pocketed weights. This guarantees the new w_u correctly classifies a greater number of training examples than the previous w_u. Note that (3aa)′ is not possible when training examples are generated dynamically as described in the "Extensions" section since there are too many potential training examples to examine.

RULES

Another important modification allows rules to be specified in addition to the training examples. Here we define a *rule* as an example E^r with corresponding classification C^r that *must* be satisfied by the resulting weights w_u. Normal training examples, on the other hand, need not be satisfied by w_u if they are noisy or contradictory, or if no w_u exists that can simultaneously satisfy all training examples (i.e., nonseparable training examples). Thus, we now seek w_u that

(1) satisfies all rules, and
(2) satisfies as many training examples as possible without violating (1).

To meet these conditions, we first must modify the initial dependency network to form a final network where the rules are separable. Note that directly contradictory rules ($E^r = E^s$, but $C^r \neq C^s$) are not allowed. (Training examples, however, may be specified arbitrarily.)

We now modify (3b) as follows:

(3b)′ Otherwise, form a new set of weights P' as follows:

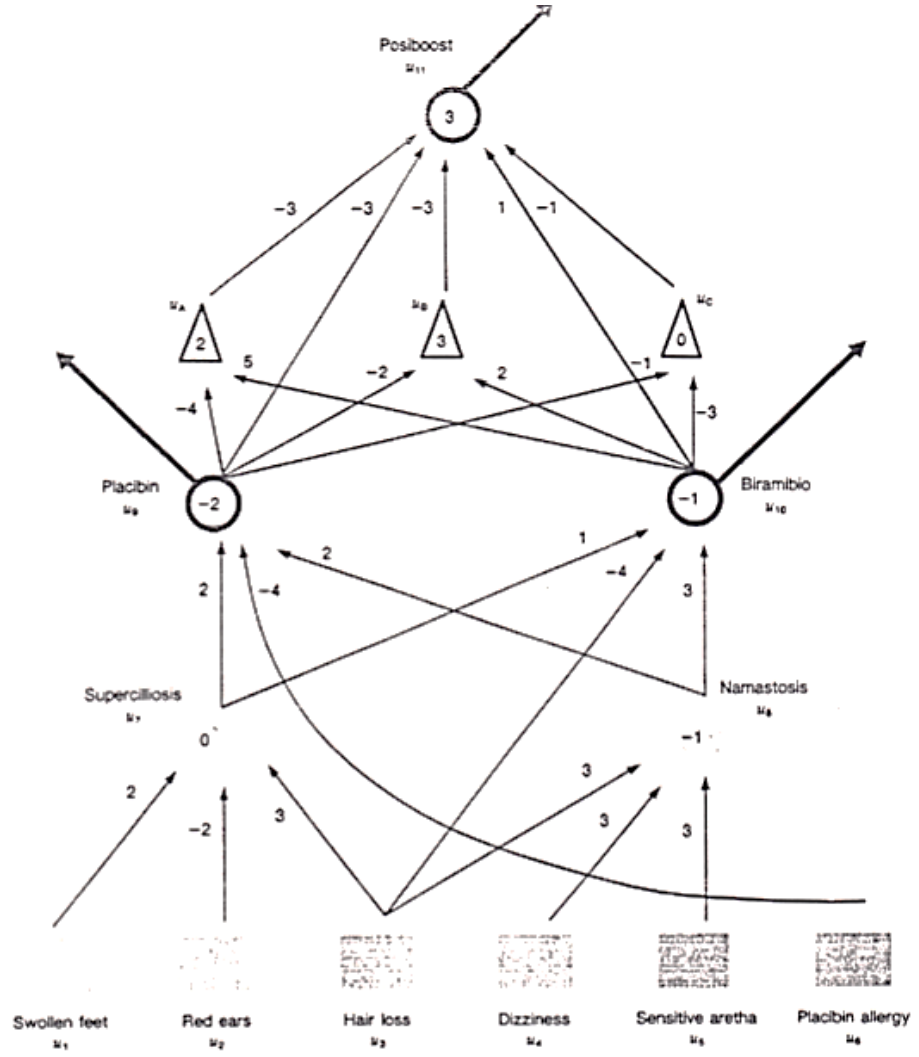
$$P' = P + C^rE^r, \tag{3}$$

and while P' violates any rule E^r (with classification C^r) repeat eq. (3) using E^r and C^r.

One final modification to the basic algorithm involves a *choice group* of cells. Here we stipulate that exactly one cell from a group of cells should be *True* for any input presented to the group; in other words, we make a single choice from the group for any input. (Nilsson [28] refers to such groups as *linear machines*.) See [14] for more details.

[*] It might be said that these weights fit handily in *pocket* or *perceptron*.

Bijlage III Lineair Discriminant Netwerk voor Sarcofaag-Ziektes



Biases are pictured within cells. The triangular cells were added to the original dependency network by the learning algorithm.

Referenties

[Bekke, 1988]
J. H. ter Bekke
Database Ontwerp
Stenfert Kroese

[Bokhoven, 1990]
W. M. G. van Bokhoven
Artificial Neural Networks - Concepts and Models
Symposium Neurale Netwerken TU-Delft Uitgave
Electrotechnische Vereeniging

[Feldman, 1988]
Computing with Structured Neural Networks
J. A. Feldman, M. A. Fanty, N. H. Goddard
University of Rochester
Computer, maart 1988

[Forsyth, 1986]
R. Forsyth, R. Rada
Machine learning; applications in expertsystems and information retrieval
Ellis Horwood Limited

[Gallant, 1988]
Connectionist Expert Systems
S. I. Gallant
Communications of the ACM, februari 1988

[Grimmett, 1982]
Probability and Random Processes
G. R. Grimmett & D. R. Stirzaker
Oxford Science Publications

[Hinton, 1986]
Learning and Relearning in Boltzmann Machines
G.E. Hinton, T. J. Sejnowski
Parallel Distributed Processing, chapter 7
MIT Press

[Hoekstra, 1990]
J. Hoekstra Simulatie van Neurale Netwerken op Conventionele Computers
Symposium Neurale Netwerken, TU-Delft
Uitgave Electrotechnische Vereeniging

[Hofstadter, 1979]
Hersenen en Denken
D. R. Hofstadter
Gödel, Escher, Bach: een eeuwige gouden band, hoofdstuk 11
Contact

[Hofstadter, 1988]
Ontwaken uit de booleaanse droom, of, Subcognitie als rekensom
D. R. Hofstadter
Metamagische thema's: op zoek naar het wezen van geest en patroon
Contact

[Hudson, 1990]
P. T. W. Hudson
Neural Networks
Symposium Neurale Netwerken, TU-Delft
Uitgave Electrotechnische Vereeniging

[Li-Min, 1989)
Building Expert Systems on Neural Architecture
Li-Min Fu
University of Wisconsin–Milwaukee, USA
First IEE International Conference on Artificial Neural Networks

[Lucas, 1988]
P. J. F. Lucas, L. C. van der Gaag
Principes van Expertsystemen
Academic Service

[Murre, 1990]
J.M.J. Murre
Lerende neurale netwerken: een beknopte inleiding
Symposium Neurale Netwerken, TU-Delft
Uitgave Electrotechnische Vereeniging

[Rich, 1983]
E. Rich Artificial Intelligence, Chapter one

[Ringland, 1988]
G.A. Ringland & D. A. Duce
Approaches to knowledge representation
Research studies, Press LTD Letchworth, Hertfordshire, Engeland

[Rumelhart&Hinton, 1986]
A General Framework for Parallel Distributed Processing
D.E. Rumelhart, G.E. Hinton, J.L. McClelland
Parallel Distributed Processing, chapter 2
MTT Press

[Rumelhart&Smolensky, 1986]
Schemata and Sequential Thought Processes in PDP Models
D.E. Rumelhart, P.Smolensky, J.L. McClelland, G.E. Hinton Parallel Distributed Processing, chapter 14
MTT Press

[Rumelhart&Zipser, 1986]
Feature Discovery by Competitive Learning
D.E. Rumelhart, D. Zisper
Parallel Distributed Processing, chapter 5
MIT Press

[Smolensky, 1986]
Information Processing in Dynamical Systems: Foundations of Harmony Theorie
P. Smolensky
Parallel Distributed Processing, chapter 6
MTT Press

[Smolensky, 1987]
Connectionist AI, Symbolic AI, and the Brain
P. Smolensky
Artificial Intelligence Review

[Tank, 1987]
Collective Computation in Neuronlike Circuits
D.W. Tank, J. Hopfield
Scientific American; Trends in Computing

[Thornton, 1989]
Learning Mechanisms which Construct Neighbourhood Representations
C. J. Thornton
University of Edinburgh
Connection Science, Vol. 1, No. 1

[Williams, 1988]
The Logic of Activation Functions
R.J. Williams
Parallel Distributed Processing, chapter 10
MTT Press

[Zeppenfeldt, 1988]
Generaliseren m.b.v. neurale netwerken
F. Zeppenveldt
Taakverslag voor deelgroep Kennisgestuurde systemen, TU Delft