KUNSTMATIGE INTELLIGENTIE EN DE GEHEIMZINNIGE IPS

door D. M. G. de Champeaux de Laboulaye
Universiteit van Amsterdam
Mirror Co

1. Kunstmatige Intelligentie en de overige wetenschappen

Al jaren lang wordt er gehamerd op het belang van informatica onderwijs en onderzoek aan de universiteiten. Computer Science is op vele universiteiten in de V.S. een al jaren bestaande studierichting en er zijn nogal wat landen in Europa met universitaire informatica instituten. In Nederland beperkt men zich tot het instellen van commissies en werkgroepen. Ondanks het gespierde taalgebruik in de door deze commissies geproduceerde rapporten [1, 2, 3, 5] ziet het er niet naar uit dat informatica op korte termijn een plaats zal krijgen aan de Nederlandse universiteiten.

De oorzaak is niet voor de hand liggend. Van belang is natuurlijk dat de universiteiten gevangen zitten in een personeelsstop. Het invoeren van een nieuwe discipline kan slechts geschieden door inkrimping van bestaande formaties. De huidige gedemocratiseerde en gedecentraliseerde universiteiten, die eerder in het nieuws komen door morrend personeel, als verworven rechten aangetast dreigen te worden, dan door fascinerende onderzoeksresultaten, missen ten enenmale de instrumenten om herorganisaties door te voeren.

Gegeven deze situatie is het niet verbazingwekkend dat Kunstmatige Intelligentie (K.I.) als *sub-discipline van Computer Science* in Nederland nauwelijks bedreven wordt.

K.I. heeft raakvlakken met o.a. psychologie en linguistiek en is een interdisciplinaire wetenschap par excellence. Dientengevolge zou men kunnen menen, dat K.I. in Nederland vanuit deze disciplines tot ontplooiing had kunnen komen. Een zodanige ontwikkeling werd belemmerd doordat K.I. van meet af aan door verschillende korte-termijn doelstellingen, door verschillende methodiek en wellicht ook door jeugdige overmoedigheid in de V.S. eerder een competentiestrijd leverde met bovengenoemde gevestigde wetenschappen dan een overbruggende en stimulerende taak vervulde. Eerst de laatste jaren vindt er binnen de psychologie en linguistiek theorievorming plaats die er op wijst, dat de door de K.I. aangeboorde problemen (h)erkend worden. Deze ontwikkeling heeft Nederland inmiddels bereikt.

In sectie 2, Drie motivaties binnen K.I., geven we een globaal overzicht van wat er zoal omgaat binnen K.I. In de daarop volgende sectie, Geconsolideerde resultaten, wordt een aantal prototypen van interessante toepassingen geschetst. In de laatste sectie vermelden we een in de toekomst aan te pakken klasse van problemen die voorshands de gewone intelligentie te boven gaan.

2. Drie motivaties binnen K.I.

Het karakteriseren van K.I. wordt gehinderd door het ontbreken van een behoorlijke definitie van intelligentie. Hoewel snelheid b.v. zeker een belangrijke component is - iemand heet vlug van begrip - zullen weinigen geneigd zijn een computer die drie miljoen vermenigvuldigingen per seconde aan kan, intelligent te noemen. Omdat snelheid en diversiteit van gedrag in de praktijk omgekeerd evenredig gekoppeld zijn, kan K.I. ruw gekarakteriseerd worden als het vergroten van de intelligente gedragscala ten koste van de snelheid. Hoe het gedragspakket samengesteld dient te worden en wat het manifesteerbare prestatieniveau behoort te zijn - de responstijd van interactieve programma's kan al bedenkelijk lang zijn - opdat er sprake is van machinale intelligentie, is een open problematiek. Als men al overeenstemming zou kunnen bereiken over zo'n pakket, dan is het plausibel dat, wanneer een programma het vereiste gedrag zou vertonen, inmiddels al is besloten dat echte intelligentie toch nog meer vereist, of zelfs wezenlijk iets anders is dan wat het programma demonstreert. Inmiddels is er vrijwel geen dimensie van intelligentie meer waarvoor niet een programma bestaat dat het betreffende gedrag in eerste aanzet simuleert. Daarmee wordt de suggestie gewekt dat het machine intelligentie-probleem in essentie is opgelost. Het ziet er echter naar uit dat de integratie van al deze programmatuur lastiger is dan de problemen die men tot op heden heeft opgelost.

Het is ook goed gebruik om te twijfelen aan de mogelijkheid van machinale intelligentie. Er wordt vaak naar voren gebracht dat computers in vergelijking met mensen niets „nieuws" zouden kunnen doen. Voor zover dit standpunt weerlegbaar is (d.w.z. niet het gevolg van een a priori filosofische visie waarin ondiscussieerbaar aan de mens unieke eigenschappen worden toegekend) moet er op gewezen worden, dat er voor allerlei vormen van leren programma's bestaan en dat er eveneens programma's zijn die zelf gecreëerde plannen tot procedures kunnen generaliseren. De tegenwerping dat het uiteindelijk de programmeur is die het allemaal heeft gedaan, is in zoverre unfair, dat ook menselijke intelligentie zich nooit in een vacuum ontplooit en na een zeer gericht en ondersteund ontwikkelingsproces voortdurend „gevoed" moet blijven. Wel moet erkend worden, dat permanente educatie door middel van assimilatie voor programma's momenteel praktisch onmogelijk is.

K.I. wordt met verschillende oogmerken bedreven. Deze doelstellingen sluiten elkaar in eerste instantie uit, zodat indien een project uiterst succesvol is vanuit één perspectief het tegelijk prutswerk lijkt vanuit een ander. Het uiteenzetten van de verschillende oogmerken is daarom nuttig voor het in kaart kunnen brengen van een op het eerste gezicht chaotisch aandoend geheel van activiteiten.

Een eerste doelstelling is om psychologisch relevante programma's te construeren. Daarbij wordt niet alleen gestreefd naar psychologische relevantie van het invoer-uitvoer gedrag, maar tot op zekere hoogte ook van de geïmplementeerde mechanismen. De eis van psychologische relevantie kan zelfs leiden tot het construeren van programma's die voor een specifieke taak dezelfde soort fouten begaan als mensen. Het in [4] gerapporteerde programma is b.v. in staat om, naar de auteur beweert, op psychologisch significante wijze twee getallen foutief af te trekken.

Bij het tweede oogmerk bekommert men zich allerminst om psychologische relevantie, maar streeft men naar het construeren van krachtige, efficiënte pro-

gramma's die een zo ruim mogelijke, maar doorgaans beperkte probleemklasse aan kunnen. De ontwerpers van dit type programma's laten zich overigens leiden door middels introspectie opgedane ideeën, maar omwille van het resultaat zullen ze niet schromen om waar nodig ad hoc technieken te gebruiken. Het DENDRAL-project [7], waar al sinds 1965 aan gewerkt wordt, is een voorbeeld bij uitstek. Dit programma is in staat diverse eigenschappen - waaronder massa-spectogrammen - van een onbekende organische stof te interpreteren en aldus de stof te identificeren. Het programma kan wedijveren met experts en indien de onbekende stof een mengsel is dan overtreft het zelfs experts. Andere voorbeelden uit deze categorie worden behandeld in de volgende sectie.

De derde doelstelling is theoretisch van aard. Gestreefd wordt naar het onderkennen van principes en methoden. Een manier is om de ruime scala van succesvolle programma's te analyseren en te trachten terugkerende componenten op te sporen. Deze activiteit kan uitmonden in het ontwerpen van „hogere", complexere programmeertalen welke frequent nodig gebleken operaties als primitieven bezitten. Aan de andere kant is er het streven om minimale programmeertalen te ontwikkelen, b.v. zgn. production systems, die aantrekkelijk zijn als doeltaal van een automatisch programmeringssysteem en/of als medium waarin leeractiviteiten kunnen plaatsvinden.

Een andere weg om een methode op te sporen is uitputtende analyse van een als lastig bekendstaand probleem. Rond 1975 was er b.v. een hausse in het bestuderen van een op het eerste gezicht lachwekkend eenvoudig drie-blokkenprobleem, zie fig. 1.

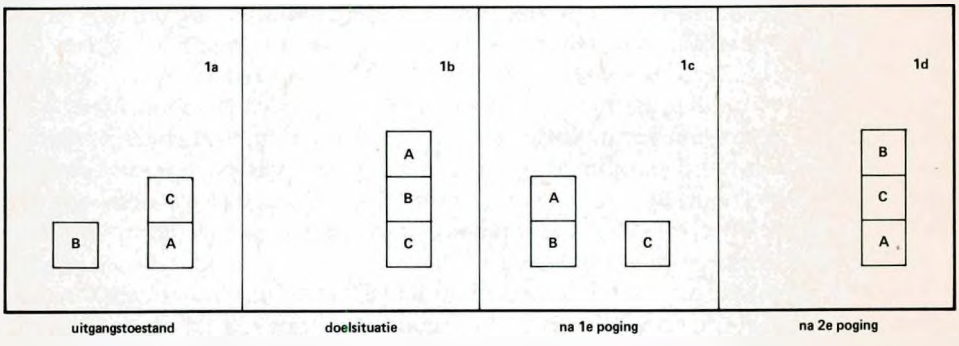

Fig. 1. In dit drie-blokken-probleem dient de configuratie uit 1a overgevoerd te worden in die van 1b. De doelsituatie kan beschreven worden als: (A op B) en (B op C). De crux van dit voorbeeld is dat een standaard strategie, nl. het successief aanpakken van deelproblemen, faalt. Het oplossen van (A op B) leidt tot 1c en van (B op C) tot 1d. In beide gevallen is het tweede deelprobleem, bij protectie van de eerste oplossing, niet meer oplosbaar. De vraag is nu hoe de verdeel-en-heers-strategie zo te verfijnen dat men geen gebruik hoeft te maken van een uitputtende zoektechniek, aangezien die in meer realistische situaties al snel tot onaanvaardbare zoektijden leidt. Een bevredigende oplossing laat nog op zich wachten.

Alle problemen die men in de loop van de jaren heeft onderzocht, blijken steeds de beheersing van in principe willekeurig complex gestructureerde en willekeurig grote datacollecties te vereisen. Om een indruk te geven van de orde van grootte waar het om gaat: een ordinaire deductiemachine genereert al snel een onhanteerbare database met een omvang van 10^7 à 10^8 bit. Het kunnen inperken van zo'n op hol slaand programma vereist het door middel van kennis *selectief* kunnen

toepassen van logische operaties. Deze kennis dient op het juiste moment ter beschikking te kunnen komen uit een zelf al immense database. Het effectief kunnen onttrekken van een willekeurig gegeven uit een kenniscollectie vereist echter dat deze collectie partieel z'n eigen meta-kennis (en meta- . . . metakennis) bevat, zonder welke een zoekmechanisme al snel in dezelfde problematiek verzeild raakt als deductiemachines.

Slechts door de complexiteit van data tevoren in te perken en ook de toegelaten operaties in te dammen, is het momenteel mogelijk om grote gegevensverzamelingen onder controle te houden, maar dan bevinden we ons eigenlijk al op het terrein van de datamanagement. Voor een datastructuur die een grotere complexiteit van data toelaat en welke te beschouwen is als een geliberaliseerde relational database zij verwezen naar de appendix.

3. Geconsolideerde resultaten

We kunnen hier door de beperkte toegemeten ruimte slechts enkele aspecten behandelen. We zullen bijvoorbeeld niet kunnen ingaan op de prestatieniveaus van schakende, wiskundige stellingen bewijzende, visuele beelden verwerkende, gesproken tekst verwerkende en natuurlijke taal producerende programma's.

Het oplossen van een probleem, indien dat geformuleerd is als het zoeken van een pad in een grote ruimte, terwijl kennis omtrent die ruimte ter beschikking staat en uitgedrukt kan worden in een heuristische afstandsfunctie, levert in feite geen moeilijkheden meer op. Er zijn zelfs voorwaarden bekend, die wanneer eraan voldaan is, garanderen, dat een beste oplossing in termen van een kortste pad gevonden zal worden. Zo'n zoektechniek zou bijvoorbeeld toegepast kunnen worden op een database die de gegevens uit het spoorwegboekje bevat, om vragen van het type „Hoe laat moet ik uit Groningen vertrekken als ik om 15h30m in Goes wil zijn?" te kunnen beantwoorden. De vertaling van een probleem - indien mogelijk - naar een formalisme waarop zo'n zoektechniek van toepassing is en het vinden van een maximaal geïnformeerde heuristische functie is echter geen eenvoudige zaak, hoewel er wel al enkele, voorlopige resultaten zijn.

De afgelopen jaren heeft de K.I. een ruim assortiment nieuwe computertalen geproduceerd. De meeste werden niet of slechts provisorisch geïmplementeerd. Wel geïmplementeerd is de gigantische taal INTERLISP, met meer dan 1000 ingebouwde, speciale functies, die de neerslag vormen van circa 15 jaar ervaring met LISP1.5. Afgezien van allerlei praktische faciliteiten die o.a. interactieve, incrementele programmering mogelijk maakt, heeft het een zgn. „spaghetti-stack" mechanisme (zie ook [6]) waarmee deze taal sterker is dan de alleen recursie toelatende talen zoals ALGOL60/68 en LISP1.5. Ruw gezegd komt dit spaghetti-mechanisme er op neer, dat, wanneer in een doolhof-type probleem twijfel rijst omtrent een al enige tijd ingeslagen weg, met behoud van de toestand een andere voortzetting bij een voormalig keuzepunt geprobeerd kan worden, zodat, indien dit alternatief niet tot voortgang leidt, alsnog de eerste poging vervolgd kan worden. En dat allemaal voor een willekeurig aantal keuzepunten met tijdelijk onderbroken voortzettingen.

Een andere belangwekkende sub-taal is de ATN van Woods [14]. Deze taal is speciaal ontwikkeld voor het kunnen beschrijven van een grammatica voor een (fragment van een) natuurlijke taal. Daarmee wordt het veel simpeler om een pro-

gramma van een (pseudo-) natuurlijke taal invoerverwerkingscomponent te voorzien. Binnen een maand konden we b.v. een gesimuleerde pocket-calculator de vraag „Hoeveel is de som van het kwadraat van de sinus van één en het kwadraat van de cosinus van 1?" laten verwerken en beantwoorden. In 1972 (zie [15]) werd met deze taal een geologische database-ondervrager gebouwd. De volgende dialoog werd o.a. gerapporteerd:
G(ebruiker): How many samples contain chromite?
P(rogramma): 3.
G: What are they?
P: S20, S45, S84.
G: Can you give me all chromite analysis for those samples?
P: S20: . . ., S45: . . . etc.

Speciale aandacht verdienen de anaphorische referenties (they, those) die dit programma, in beperkte mate, kan behandelen.

In een vragenbeantwoorder PLANES [13], die werkt op een veel grotere, en ditmaal relational database werd eveneens gebruik gemaakt van de ATN-taal. De vragenbeantwoorder heeft als extra mogelijkheden:

- het kunnen behandelen van ellipsis, b.v.
  How many flights did it make in feb 73?
  . . .
  During april?
  . . .;
- het kunnen behandelen van ongrammaticale maar toch betekenis hebbende zinnen;
- het kunnen opvangen van foutief getypte woorden;
- het kunnen beantwoorden van vragen *over* de database (d.w.z. niet alleen betreffende de inhoud);
- het kunnen verwerken van eenvoudige definities, waarmee de gebruiker zonder chirurgische ingreep het begripsvermogen kan uitbreiden (op bescheiden schaal weliswaar, maar toch).

Een geheel andere methode voor het opslaan en het terugzoeken van kennis wordt gerapporteerd in [7, 8, 9]. Het gaat daarbij om minder objectieve kennis van het type „in situatie zus of zo verdient het aanbeveling om met de mogelijkheden x, y, z rekening te houden". Het geheugen bestaat uit een collectie regels, hiërarchisch geordend, zodat een regel òf een eindconclusie bevat òf verwijst naar overige potentieel relevante regels.

Deze techniek is al gebruikt voor classificatie van onbekende molecuulstructuren in het eerder vermelde DENDRAL-project en voor consultering (diagnose en therapievoorstel) op het gebied van bacteriële infectiebestrijding in het MYCIN-project. De MYCIN-consultant wordt in [8, 9] vanwege de cryptische medische terminologie gepresenteerd als een beleggingsadviseur. Een MYCIN-regel, geherformuleerd, ziet er dan voor de gebruiker in het Engels vertaald uit als:

„If (1) the time-scale of the investment is long term,
(2) the desired return on the investment is greater that 10% and
(3) the area of the investment is not known
then AT & T is a likely (0.4) choice for the investment".

Omdat deze projecten met het oog op praktisch gebruik ontwikkeld zijn en worden, heeft men veel aandacht geschonken aan natuurlijke-taal-interfaces, aange-

zien interactief gebruik anders vrijwel onmogelijk is. Niet alleen de gebruiker kan op deze faciliteiten rekenen, maar ook de specialist die de kennis moet leveren voor de opbouw en/of uitbreiding van de regels. Programmacomponenten staan hiervoor ter beschikking, die o.a. gebruik makend van meta-regels vanuit natuurlijke taal bestuurd kunnen worden.

Een dialoog, vermeld in [8] leidt b.v. tot opname van de nieuwe regel:

„If (1) the income-tax bracket of the client is 50%,
(2) the client follows the market carefully and
(3) the amount of investment experience of the client is moderate
then there is evidence (0.8) that the area of the investment is high-technology".

Daarmee werd een onderkend foutief advies gecorrigeerd.

Een voor de gebruiker essentiële faciliteit is het kunnen beantwoorden van waarom-vragen, waarmee de gebruiker in staat is na te gaan hoe een aanbeveling wordt bereikt, zodat het vertrouwen op peil kan blijven en/of een vermeende onvolledigheid getraceerd kan worden.

4. K.I. in de toekomst

Kennisneming van de wijze waarop een „intelligent" programma functioneert werkt doorgaans ontluisterend. Dan blijkt namelijk dat enkele zeer voor de hand liggende huis- tuin- en keuken-methodes verantwoordelijk gesteld moeten worden voor het vertoonde gedrag. Tegelijkertijd is het ontmoedigend te moeten ervaren hoeveel programmeeractiviteit noodzakelijk is om de programma's te ontwikkelen. Implementatie van een ordinair goed idee vergt al snel een jaar werk. Het ziet er dan ook naar uit dat K.I. voorlopig de handen vol heeft aan het formaliseren en combineren van de overvloed aan intuities die overal voor het oprapen liggen. Pas daarna kan de vraag gesteld worden of intelligentie meer is dan een handige combinatie van de principes die men nu al heeft onderkend.

Of machinale intelligentie potentieel die van de mens kan overstijgen is een open kwestie.

Dit vraagstuk krijgt een nieuwe dimensie door de opkomst van de micro-processor. Het is momenteel wel mogelijk om in 1 m^3 zo'n 30000 micro-processors onder te brengen. Elke micro-processor is een computertje op zich met zo'n 24000 geheugenplaatsen. Een cruciaal vraagstuk is echter hoe chips met elkaar verbonden moeten worden. Deze problematiek is nauw verbonden met de vraag hoe een probleem op te splitsen, zodat meer dan enkele processoren effectief kunnen bijdragen. Terwijl men jaren gemeend heeft dat computers zo lastig hanteerbaar zijn omdat programma's in een sequentieel keurslijf gewrongen moeten worden, blijkt tegenwoordig dat niemand goed weet hoe een groot aantal processors simultaan nuttig bezig te houden. (De lezer wordt uitgenodigd een aanpak te verzinnen voor het sorteren van N getallen met een willekeurig aantal processors.) Intrigerende vragen zijn nu of we hier op een blinde vlek in ons denken stuiten en of er toch een goed schakelschema bestaat waarmee voorshands ongekende mogelijkheden toegankelijk worden. De ips?

Appendix

We schetsen hier een alternatief voor een relational database. Een relational database kan gezien worden als een drietal bestaande uit: xx een of andere gegevenscollectie die zich bevindt op een door een computer manipuleerbaar medium; xx welke een structuur van een (n-dimensionale) tabel met een bijbehorende beschrijvingstaal heeft; xx voorzien van interfaces die een verbinding leggen tussen het conceptuele niveau van beschrijvingen en het computer manipulatie mechanisme. Deze opzet heeft het belangrijke voordeel dat de gebruiker afgeschermd is van onsmakelijke implementatie details van de gegevenscollectie. Tevens is het mogelijk om een verschillend „zicht" te hebben op de gegevenscollectie: enerzijds door tegen een andere zijkant van de tabel aan te kijken, anderzijds door een deel van een tabelingang af te kunnen schermen.

Ernstige nadelen van deze opzet zijn: xx dat het later toevoegen van een extra dimensie of zelfs van een extra tabelingang een stevige ingreep vereist; xx dat partieel of, nog erger, incidenteel ingevulde tabelkolommen hoewel niet onmogelijk toch tegennatuurlijk blijven; xx dat het beschikbaar kunnen houden van inmiddels verouderde gegevens (voor b.v. „Wat was het personeelsverloop tussen 1970-1975 op afdeling X?") met wijduiteenlopende mutatiefrequentie van individuele elementen geen voor de hand liggende oplossing heeft.

Een (1-dimensionale) relationele database kan gezien worden als een weiland met graspollen (zie fig. 2 voor één graspol), waarbij elke graspol evenveel gelijkgenaamde sprieten heeft. De sprietnamen corresponderen met kolomnamen en aan de punten „hangen" de bijbehorende tabelwaarden. Elke graspol bezit een unieke naam, de zgn. sleutel.

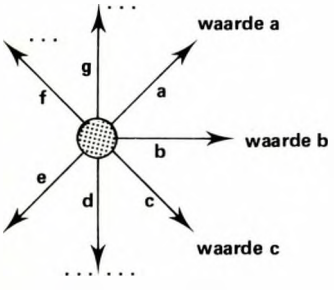

Fig. 2. Een „graspol" met „sprieten" uit een relational database

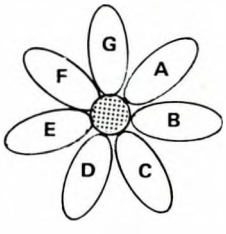

Fig. 3. Een „bloem" met „bladeren" uit een gegeneraliseerde database

Als eerste stap laten we toe dat graspollen verschillende sprietencollecties bezitten. Als tweede stap eisen we niet meer dat sprieten enkelvoudige namen hebben, maar laten we toe dat sprietnaam evenals sprietwaarde eventueel complexe entiteiten zijn.

Tenslotte integreren we sprietnaam en sprietwaarde tot een „feit", hier een blad genaamd (zie fig. 3), waarbij we er van uit gaan dat er een retrieve-mechanisme bestaat dat bereid is om met behulp van een partiële beschrijving alle bladeren aan te wijzen die aan de beschrijving voldoen. Aangezien we nog altijd toestaan dat elke bloem, voorheen graspol, zijn eigen collectie bladeren heeft, kan zo'n database een veldboeket heten.

Bovengenoemde nadelen van een relationele database worden door deze structuur ondervangen. Een blad kan bijvoorbeeld representeren: „het salaris van werknemer X was tussen t1 en t2 *f* Y".

Efficiënte batchverwerking vereist evenwel een zorgvuldiger implementatie door de meer complexe structuur.

Kanttekeningen

xx De graspollen datastructuur correspondeert in feite met de uit 1960 stammende zgn. „property lists" uit LISP1.5 [11].

xx Retrieving door partiële omschrijving is mogelijk door pattern-matching technieken, waarbij het echte werk gedaan wordt door unificatie-algorithmen [10, 12].

xx Voor het interactief kunnen werken met een database is er minstens één inverteringsmechanisme nodig (b.v. hash-tabel of geïnverteerde file), die een correspondentie legt tussen een externe karakterisering en een interne sleutel van het bijbehorende record (tabelingang/graspol/bloem). Met meerdere inverteringsmechanismen kan men vermijden dat, voor het opsporen van een sleutel met een alternatieve karakterisering of zelfs met een omschrijving het hele bestand doorlopen moet worden. Een nadeel van deze redundantie is dat bij een mutatie extra aandacht geschonken moet worden aan het bijwerken van de additionele data. Belangrijk onderzoek zou gedaan kunnen worden om te bepalen hoe een interactieve database-manager op grond van actuele sessies aanbevelingen zou kunnen genereren voor het toevoegen van extra inverteringsmechanismen en in een volgend stadium zelfs hoe deze extra toegangskanalen automatisch toegevoegd kunnen worden.

Literatuur

(1) Academische Raad, Struktuurplan Informatica (W.O.), december 1974;
(2) ASTUR, De opleiding in de Informatica aan de Universiteit van Amsterdam, december 1972;
(3) Bailey, D. E., Sociaal Wetenschappelijke Informatica, Technisch Centrum F.S.W., Universiteit van Amsterdam, mei 1978;
(4) Bradzil, P., Experimental Learning Model, Proceedings of the AISB/GI Conference, Hamburg 1978, pp 46-50;
(5) Centrale Informatica Commissie, Inventarisatierapport betreffende Onderwijs en Onderzoek in de Informatica aan de Universiteit van Amsterdam, oktober 1976;
(6) Champeaux, D. de, Contexten en Processen, rapport vakgroep Bedrijfsinformatica, Universiteit van Amsterdam, november 1976;
(7) Davis, R. et al, Production Rules as Representation for a Knowledge-Based Consultation Program, Artificial Intelligence, vol. 8, no. 1, Feb 1977, pp 15-45;
(8) Davis, R., Interactive Transfer of Expertise: Acquisition of New Inference Rules, Proceedings of IJCAI-77, MIT, pp 321-328;
(9) Davis, R. & Buchanan, B. G., Meta-Level Knowledge: Overview and Applications, Proceedings of IJCAI-88, MIT, pp 920-927;
(10) Martelli, A. & Montanari, U., Unification in Linear Time and Space, Nota Interna B76-16, Luglio 1976, Pisa;
(11) McCarthy, J. et al, LISP1.5 Programmer's Manual, MIT, 1962;
(12) Robinson, J. A., A Machine-Oriented Logic Bases on the Resolution Principle, J.ACM, vol. 12, no. 1, Jan 1965, pp 23-41;
(13) Waltz, D. L., An English Language Question Answering System for a Large Relational Database, Comm.ACM, vol. 21, no. 7, July 1978, pp 526-539;
(14) Woods, W. A., Transition Network Grammars for Natural Language Analysis, Comm.ACM, vol. 13, no. 10, October 1970, pp 591-606;
(15) Woods, W. A. et al, The Lunar Sciences Natural Language Information System, BBN Report 2378, June 1972, NTIS no. N72-28984.