Inteligență artificială

Litera A

A* Admissibility = Admisibilitatea algoritmului A*

Fie un algoritm A* care utilizeaza f(S) = g(S) + h(S). Daca

1. $g(S) = \sum_{k=i}^{n} cost\_arc(S_k, S_{k+1})$, cu $S_n = S$ si S_i starea initiala,
2. functia h satisface conditia de admisibilitate (0 <= h(S) <= h*(S)),
3. cost_arc(S,S') >= c, pentru orice doua stari S, S', unde c > 0 este o constanta si costul arcelor este finit

atunci algoritmul A* este *admisibil*, adica este garantat sa gaseasca calea de cost minim spre solutie.

S-a demonstrat ca algoritmul A* este o strategie completa, chiar si pentru spatii de cautare infinite.

A* Algorithm = Algoritmul A*

Vezi Strategia A*

A* Informedeness = Gradul de informare al algoritmului A*

Fie doi algoritmi (admisibili) A*, A_1 si A_2 pentru rezolvarea aceleiasi probleme, cu functiile de evaluare

- $f_1(S) = g(S) + h_1(S)$
- $f_2(S) = g(S) + h_2(S)$

Se spune ca algoritmul A_2 este *mai informat* decit algoritmul A_1 daca $h_2(S) > h_1(S)$ pentru orice stare S, S<>S_f cu S_f starea finala.

Se demonstreaza ca daca A_2 este mai informat decat A_1 atunci A_2 nu expandeaza niciodata mai multe stari decit A_1. Se demonstreaza de asemenea ca daca componenta h a functiei f a unui algoritm A* are propietatea de a fi *monoton crescatoare*, i.e. h(S) <= h(S') + cost_arc(S,S') pentru orice doua stari S, S', cu S' starea succesoare a lui S, atunci un nod, odata introdus in lista TERITORIU, nu va mai fi niciodata eliminat din TERITORIU si reinserat in FRONTIERA.

A* Strategy = Strategia A*

Strategia A* si algoritmul corespunzator urmareste determinarea caii de cost minim intre nodul asociat starii initiale si nodul asociat unei stari finale. Acest obiectiv include si problema gasirii caii spre solutie care contine un numar minim de arce, i.e. calea care implica aplicarea unui numar minim de operatori. Problema gasirii celei mai scurte cai este o particularizare a cazului general, particularizare in care costul arcelor este considerat unitar.

Algoritmul A* este o strategie de cautare informata de tip "best-first" pentru reprezentarea solutiei folosind spatiul starilor. Caracteristica distinctiva a algoritmului consta in modul de definire a functiei de evaluare w(S) care este notata in acest caz cu f(S). Nodul ales pentru expandare este nodul cu valoarea minima a functiei f. Deoarece f evalueaza nodurile din punct de vedere al gasirii solutiei de cost minim, f(S) include doua componente:

- g(S), o functie care estimeaza costul real g*(S) al caii de cautare intre starea initiala Si si starea S,
- h(S), o functie care estimeaza costul real h*(S) al caii de cautare intre starea curenta S si starea finala Sf.

In aceste conditii functia de evaluare se defineste f(S) = g(S) + h(S) si reprezinta o estimare a costului real f*(S) = g*(S) + h*(S) al unei solutii a problemei care trece prin starea S.

Functia g(S) este calculata ca fiind costul actual al drumului parcurs in cautare intre starea initiala S_i si starea S, deci

$$g(S) = \sum_{k=i}^{n} cost\_arc(S_k, S_{k+1})$$, cu $S_n = S$ si S_i

Daca spatiul starilor este un arbore, g(S) este o estimare perfecta a costului real g*(S). Daca spatiul de cautare este graf, g(S) poate numai sa supraestimeze costul real g*(S). In consecinta g(S) >= g*(S) pentru orice stare S.

Functia h(S) este functia purtatoare de informatie euristica si trebuie definita in raport cu caracteristicile problemei particulare de rezolvat. Pentru ca algoritmul A* sa gaseasca solutia optima, functia h trebuie sa fie nenegativa si sa subestimeze intotdeauna costul real h*(S) al caii care a mai ramas de parcurs pana in starea finala, deci să fie admisibilă.

Admissible Heuristic Function = Funcţie euristică admisibilă

O functie euristica h se numeste *admisibila* daca 0 <= h(S) <= h*(S) pentru orice stare S. Definitia stabileste *conditia de admisibilitate* a functiei h si este folosita pentru a defini *proprietatea de admisibilitate* a unui algoritm A*.

AND/OR Graph = Graf SI/SAU

Un *graf SI/SAU* este construit pe baza urmatoarelor reguli:

1. Fiecare nod reprezinta fie o singura problema fie o multime de probleme ce trebuie rezolvate.
2. Un nod ce reprezinta o singura problema nu are descendenti. Problema este fie o *problema elementara*, fie o *problema neelementara* care nu se mai poate descompune in subprobleme.
3. Nodurile ce reprezinta multimea de subprobleme in care s-a descompus o problema prin aplicarea unui operator de descompunere se numesc *noduri SI.*
4. Nodurile ce reprezinta descompuneri alternative ale unei probleme in subprobleme (prin aplicarea diversilor operatori de descompunere) se numesc *noduri SAU*. Aceste noduri au ca descendenti noduri SI.

Artificial Intelligence = Inteligență artificială

Inteligența artificială porneşte de la premisa conform căreia toate activitățile cognitive pot fi modelate ca procese de calcul. Această premisă are o tradiție de peste 2000 de ani, începând cu Aristotel şi Platon care credeau ca gândirea, la fel ca orice alt fenomen fizic, poate fi studiată folosind observația ştiințifică şi inferența logică. Mai târziu, Gottfried Leibnitz, prin cercetările sale în care privea gândirea drept calcul, pune bazele tratatului lui George Boole despre logica simbolică, intitulat semnificativ "Legile gândirii" (The Laws of Thought). Apariția calculatoarelor şi progresele făcute în domeniul calculului simbolic au condus la dezvoltarea unei noi ramuri a ştiinței calculatoarelor, previzionată de Alan Turing în articolul sau "Computing Machinery and Intelligence", respectiv inteligența artificială.

Inteligența artificială a fost inițiată ca o disciplină formală de studiu în timpul conferinței din 1956 de la Dartmouth College de către John McCarthy, Marvin Minsky, Allen Newell şi Herbert Simon, având ca scop ştiințific înțelegerea principiilor şi mecanismelor care permit acțiunea inteligentă şi ca scop ingineresc proiectarea sistemelor care permit rezolvarea problemelor de dificultate considerabilă, cu un nivel de competență ridicat.

De-a lungul scurtei dar prolificei existențe a domeniului s-au propus diverse definiții ale inteligenței artificiale, cele mai multe variind de-a lungul a două dimensiuni. Conform unei prime dimensiuni, inteligența artificială este văzută fie ca *studiul proceselor de gândire şi raționament*, fie ca *studiul modelării comportamentului inteligent*. Cea de a doua dimensiune impune alte două viziuni alternative ale domeniului, respectiv studiul şi dezvoltarea modelelor cu performanțe similare gândirii umane şi, alternativ, a modelelor performante din punct de vedere al unui concept ideal de inteligență, şi anume raționalitatea, văzută drept capacitatea de acțiune corectă în raport cu un scop dat. Aceste două dimensiuni definitorii identifică, prin intersecție, patru clase de definiții posibile ale inteligenței artificiale care stabilesc în acelaşi timp şi scopurile posibile ale disciplinei: sisteme care emulează gândirea umană, sisteme care emulează acțiunea umană, sisteme care simulează gândirea rațională şi sisteme care simulează acțiunea rațională.

Celebrul test Turing, conceput pentru a oferi o definiție operațională a inteligenței, vede inteligența artificială ca acea ramură

a ştiinţei calculatoarelor care încearcă să construiască *sisteme care emulează acţiunea umană*. Domeniul interdisciplinar al ştiinţelor cognitive, care include modele computaţionale şi psihologice, cât şi cercetările dedicate studiului gândirii omeneşti se încadrează în eforturile de a construi *sisteme care emulează gândirea umană*.

Principiile gândirii raţionale, prefigurate de Aristotel prin celebra lege a silogismului, stau la baza eforturilor de concepţie a *sistemelor care simulează gândirea raţională* şi au creat tradiţia abordării inteligenţei artificiale prin prisma logicii simbolice. Logica simbolică, ca limbaj universal acceptat, oferă atât un model de formalizare a cunoştinţelor cât şi o metodă de raţionament corect, valid, în rezolvarea problemelor. Cu toate acestea, inferenţele corecte, valide, reprezintă numai o parte a comportamentului raţional. Există situaţii în care atingerea unui scop se poate baza pe inferenţe nevalide, de exemplu inducţia sau obţinerea unor concluzii în condiţiile existenţei unor informaţii incomplete despre lume, şi alte situaţii în care comportamentul raţional nu implică nici un fel de inferenţă, de exemplu reflexele de protecţie. Aceste observaţii au dus la dezvoltarea cercetărilor de inteligenţă artificială dedicate construirii unor *sisteme care simulează acţiunea raţională* şi a căror principii includ legile de inferenţă validă din logica clasică dar nu se limitează numai la acestea. S-a dovedit destul de repede că această parte a acţiunii raţionale, care nu poate fi formalizată printr-un model valid cum ar fi logica cu predicate de ordinul întâi, este cel mai dificil de simulat într-un sistem artificial.

Litera B

Basic Search Strategies = Strategii de cautare de bază

Vezi Strategii de cautare neinformate

Best-first Search = Căutare "best-first"

Vezi Strategia de căutare "best-first"

Best-first Search Strategy = Strategia de căutare "best-first"

Ideea *strategiei de cautare "best-first"* este aceea de a selecta spre expandare cel mai bun nod din spatiul de cautare generat pe baza cunostintelor euristice, deci pe baza unei estimari. Calitatea unui nod, din punct de vedere al gasirii solutiei, poate fi estimata in diverse moduri. Se poate atribui nodului gradul de dificultate in solutionarea problemei reprezentata de acel nod. Se poate estima calitatea unei multimi de solutii candidate care contin acel nod, deci solutii partiale care contin o cale ce duce la acel nod. O a treia alternativa este aceea de a evalua cantitatea de informatie care poate fi obtinuta prin expandarea acelui nod si importanta acestei informatii in ghidarea procesului de cautare. In toate aceste cazuri calitatea unui nod este estimata de *functia de evaluare euristica*, notata **w(n)** pentru nodul n, care poate depinde de descrierea lui n, de descrierea scopului si de cunostinte suplimentare despre problema.

Blind Search Strategies = Strategii de cautare oarbă

Vezi Strategii de cautare neinformate

Breadth-first Search = Căutarea pe nivel

Vezi Strategia căutării pe nivel

Breadth-first Search Strategy = Strategia căutării pe nivel

Cautarea pe nivel, numita si cautare in latime, este o strategie care expandeaza starile urmatoare in ordinea apropierii fata de nodul stare initiala. Cu alte cuvinte, aceasta strategie considera intii toate secventele posibile de n operatori inaintea secventelor de n+1 operatori.

Cautarea poate fi uneori lunga si complex computationala din punct de vedere al spatiului utilizat deoarece pentru fiecare nivel sunt generate toate starile succesoare posibile. Cu toate acestea, strategia de cautare pe nivel garanteaza gasirea solutiei, in cazul in care aceasta exista (deci este o strategie completă) si, in plus, gaseste cel mai scurt drum spre solutie in termenii numarului de tranzitii de stari executate.

Litera C

CLOSED = TERITORIU

In implementarea strategiilor de cautare se folosesc de obicei doua liste:

- FRONTIERA - lista nodurilor evaluate
- TERITORIU - lista nodurilor expandate

In parcurgerea spatiului de cautare un nod poate fi:

- *necunoscut* - nodul apartine partii neexplorate a spatiului de cautare,
- *evaluat* - nodul este cunoscut dar fie nu se cunoaste nici un succesor al lui, fie se cunosc numai o parte din succesorii lui,
- *expandat* - nodul este cunoscut si, in plus, se cunosc toti succesorii lui.

Lista TERITORIU reprezinta deci partea cunoscuta a spatiului de cautare.

Common Sense Knowldge = Cunoştinţe de bun simţ sau cotidiene

O directie a cercetarilor de inteligenta artificiala a fost aceea a rezolvarii problemelor banale, cotidiene, care necesita *cunostinte de bun simt*. Aceste probleme includ rationamentul despre obiecte fizice si relatiile intre ele, si rationamentul despre actiuni si consecintele acestora. Oricine stie, de exemplu, ca un obiect nu poate sa fie simultan in doua locuri diferite sau ca nu trebuie sa dea drumul unui pahar din mina deoarece poate sa cada si sa se sparga. Aceste comportamente pot fi greu caracterizate ca necesitind inteligenta si, totusi, ele sunt cele mai greu de modelat intr-un program. Cunostintele de bun simt sunt la indemana oricarui om dar ele trebuie reprezentate explicit intr-un program, iar volumul lor este impresionant. Surprinzator, cercetarile de inteligenta artificiala au avut rezultate cu mult mai bune in domenii ca rezolvarea problemelor formale dificile cum ar fi jocurile, demonstrarea teoremelor, sau a problemelor care necesita expertiza umana intr-un anumit domeniu, decit in domeniile care necesita cunostinte de bun simt. S-a reusit construirea unui program care sa demonstreze teoreme matematice complicate si care sa descopere chiar concepte matematice noi, dar nu s-a reusit construirea unui program care sa stie tot ceea ce stie un copil de doi ani!

Common Sense Reasoning = Raţionament de bun simţ

Vezi cunoştinţe de bun simţ.

Completeness of a Search Strategy = Completitudinea unei strategii de căutare

O strategie de căutare este *completă* dacă strategia garantează găsirea soluţiei problemei în cazul în care problema admite o soluţie. Strategia de căutare pe nivel este, de exemplu, o strategie completă.

Control Strategy = Strategie de control

Se numeste *strategie de control* procesul de aplicare repetata a regulii de inferenta pentru a ajunge la solutie, de preferinta cat mai repede. Regula de inferenta impreuna cu strategia de control formeaza nucleul motorului de inferenta al unui sistem bazat pe cunostinte. Datorita descoperirii si utilizarii unor strategii de control adecvate, programele de inteligenta artificiala au reusit sa rezolve probleme grele, deci NP-complete, intr-un timp acceptabil pentru dimensiuni semnificative ale intrarii.

Litera D

Deduction = Deducţie

Deductia este o forma fundamentala de rationament in planul conceptelor in care concluzia decurge cu necesitate din premise. Exemplul tipic de deductie este silogismul.

Default Reasoning = Raţionament implicit

Necesitatea de a face presupuneri este întâlnită frecvent în raţionamentul despre lumi incomplet specificate. Rezultatul

inferențelelor necesare realizării unui astfel de raționament pot fi văzute drept *convingeri* care pot fi modificate sau invalidate pe baza unor cunoştinţe (probe) obținute ulterior. Ele se numesc convingeri tocmai pentru a fi diferențiate de faptele sau relațiile ce sunt întotdeauna adevărate (cunoştinţe), şi au ca scop reprezentarea stării epistemice (posibil incomplete sau eronate) a unui agent rațional despre o lume dată.

Raționamentul implicit, întâlnit în multe situații, de exemplu raționamentul de bun simț, corespunde procesului de derivare a noi concluzii pe baza unor şabloane inferențiale de tipul "în absența oricărei informații contrare, putem presupune că ..." Să considerăm cazul în care am hotărât să mergem cu automobilul la servici şi ştim că traseul durează 30 de minute. Dacă planificăm să plecăm la ora 7:30 putem concluziona că vom ajunge la servici la ora 8:00. Făcând o astfel de inferență am presupus însă că automabilul nu este stricat, că drumul nu este aglomerat, că nu se produce nici o pană de cauciuc, etc. Aceasta înseamnă că am utilizat o regulă de inferență implicită de forma:

> *"Dacă A este adevărat şi dacă putem presupune că B este adevărat atunci putem concluziona C."*

Ce înseamnă faptul că putem presupune B adevărat? Prin aceasta se înțelege că B poate fi considerat adevărat atât timp cât nu avem sau nu putem infera informații contrare lui B. Regula anterioară se poate deci reformula astfel:

> *"Dacă A este adevărat, în absența oricăror informații contrare lui B (sau despre falsitatea lui B), putem concluziona C."*

Problema se reduce astfel la interpretarea frazei "în absența oricăror informații despre falsitatea lui B." Interpretarea considerată este aceea că este consistent să presupunem B, deci că B este consistent cu ceea ce se ştie sau cu ceea ce se poate deduce. În continuare, regula poate fi din nou reformulată astfel:

> *"Dacă A este adevărat şi dacă B este consistent cu ceea ce se cunoaşte atunci putem concluziona C."*

Ce înseamnă de fapt consistența lui B cu ceea ce se cunoaşte? A oferi o definiție formală a acestei cerințe de consistență este, aşa cum spune Reiter, cel mai dificil aspect în definirea unui formalism de reprezentare a raționamentului implicit.

Depth-first Search = Căutarea în adâncime

Vezi Strategia căutării în adâncime

Depth-first Search Strategy = Strategia căutării în adâncime

Cautarea in adancime este o strategie care expandeaza starile cel mai recent generate, cu alte cuvinte nodurile cu adancimea cea mai mare din lista FRONTIERA. In consecinta, aceasta strategie parcurge o cale de la starea initiala pana la o stare ce poate fi stare finala sau care nu mai are nici un succesor. In acest ultim caz strategia revine pe nivelele anterioare si incearca explorarea altor cai posibile.

Strategia cautarii in adancime nu garanteaza obtinerea unei solutii a problemei, chiar in cazul in care solutia exista. O astfel de situatie poate apare, de exemplu, in cazul unui spatiu de cautare infinit in care ramura pe care s-a plecat in cautare nu contine solutia. Din acest motiv se introduce de obicei o limita a adincimii maxime de cautare, AdMax. Daca s-a atins aceasta limita fara a se gasi solutia, strategia revine si inspecteaza stari de pe nivele inferioare lui AdMax dar aflate pe cai diferite. Solutia care s-ar gasi la o adancime de AdMax+p, de exemplu, ar fi pierduta. Daca strategia de cautare gaseste solutia, aceasta nu este neaparat calea cea mai scurta intre starea initiala si starea finala.

Algoritmul cautarii in adancime prezinta avantajul generarii unui numar de stari mai mic decat in cazul algoritmului de cautare pe nivel, deci consumul de spatiu este mult redus. Evident, algoritmul plateste pretul limitarilor indicate anterior.

Litera F

Frame Problem = Problema cadrului

Problema cadrului se referă la dificultatea reprezentării tuturor aspectelor lumii care nu se modifică în urma executării unei acțiuni. De exemplu, mutarea unui obiect dintr-un loc în altul nu modifică nici forma nici culoarea acelui obiect. Un obiect poate avea o mulțime de astfel de proprietăți invariante la o anumită tranziție de stare (acțiune) şi reprezentarea lor explicită

poate creşte în mod exagerat dimensiunea bazei de cunoştinţe. Raționamentul implicit propune soluții pentru tratarea coerentă şi eficientă a acestor dificultăți de reprezentare.

Litera H

Heuristic Knowledge = Cunoştinţe euristice

Cunostintele euristice reprezinta o forma particulara de cunostinte utilizata de oameni pentru a rezolva probleme complexe. Ele reprezinta cunostintele utilizate pentru a judeca corect, pentru a lua o decizie, pentru a avea un comportament dupa o anumita strategie sau a utiliza trucuri sau reguli de bun simt. Acest tip de cunostinte nu este nici formalizat, nici demonstrat a fi efectiv si citeodata nici corect, dar este frecvent utilizat de oameni in numeroase situatii. Judea Pearl, in lucrarea sa "Heuristics. Intelligent Search Strategies for Computer Problem Solving", defineste cunostintele euristice astfel:

> *"Euristicile sunt criterii, metode sau principii pentru a alege intre diverse alternative de actiune pe aceea care promite a fi cea mai eficienta in realizarea unui scop. Ele reprezinta compromisuri intre doua cerinte: necesitatea de a lucra cu criterii simple si, in acelasi timp, dorinta de a vedea ca aceste criterii fac o selectie corecta intre alternativele bune si rele."*

Heuristic Search Strategies = Strategii de cautare euristice

Vezi Strategii de cautare informate

Litera I

Induction = Inducție

Inductia este o forma de rationament care trece de la particular la general, de la fapte la concepte. Exista doua tipuri de inductie: inductia completa, daca se enumera toate cazurile existente, si inductia incompleta, daca se enumera numai o parte din cazurile existente. Inferenta inductiva sta la baza majoritatii proceselor de invatare.

Inference = Inferență

Conform dictionarului enciclopedic al limbii romane, *inferenta* este o forma de rationament prin care se trece de la un enunt la altul in mod deductiv sau inductiv direct, caz in care se numeste inferenta imediata, sau indirect, caz in care se numeste inferenta indirecta.

O definiție alternativă urmează. Se numeste *inferenta* sau *regulă de inferenta*, sau pe scurt inferenta, procedura de obtinere la un moment dat, a noi elemente (fapte) implicate in mod direct de elementele particulare reprezentarii. Fiecare model de reprezentare a cunostintelor are metode de inferenta specifice. Pentru a putea ajunge la solutia unei probleme este necesar, de cele mai multe ori, o aplicare repetata a metodei de inferenta.

Inference Rule = Regulă de inferență

Vezi inferență.

Informed Search Strategies = Strategii de cautare informate

Strategiile de cautare *informate*, numite şi strategii euristice, consideră starile urmatoare de inspectat pe baza unor funcții de evaluare sau dupa criterii euristice. Strategia foloseste o functie de evaluare a situatiei globale sau locale care indica starea urmatoare cea mai promitatoare din punct de vedere al avansului spre solutie.

Spre deosebire de strategiile de cautare neinformata care inspecteaza sistematic toate starile spatiului de cautare pana in momentul gasirii starii finale, strategiile de cautare euristice incearca reducerea numarului de stari din spatiul de cautare inspectate pana la atingerea starii finale, pe baza diverselor criterii, cum ar fi functiile euristice. Strategia alpinistului este un exemplu de cautare informata. Alte exemple sunt strategia de cautare "best-first", algoritmul A^* si algoritmul AO^*. Algoritmii A^* si AO^* urmaresc in principal, pe linga reducerea numarului de stari inspectate, gasirea solutiei optime.

Irrevocable Search Strategy = Strategie de căutare irevocabilă

Strategiile de cautare *irevocabile* sunt strategiile pentru care nu există posibilitatea revenirii într-o stare anterior parcursă în căutare. Un operator aplicabil este selectat, acest operator se aplica unei stari pentru a obtine o noua stare, iar starea anterioara este uitata (nu este memorata).

O strategie irevocabila este *strategia de cautare a alpinistului*, bazata pe criterii de optim local. Aceasta strategie se numeste a alpinistului deoarece, la fel ca un alpinist care doreste sa ajunga repede pe virful unui munte, alege starea urmatoare de nivel maxim pe baza unei functii de evaluare a starilor. Strategia este irevocabila deoarece pentru o stare curenta, se genereaza starile urmatoare, se alege starea de nivel maxim ca stare urmatoare si atat starea curenta cat si celelalte stari de pe nivelul starii urmatoare sunt uitate. Selectia se face irevocabil, deci nu se mai poate reveni intr-una din starile anterioare starii curente sau intr-una din alternativele starii curente. Strategia alpinistului, desi simpla si putin consumatoare de memorie, prezinta o serie de limitari. De exemplu, daca problema cere determinarea starii cu o valoare maxima a functiei de evaluare, maximul global poate sa nu fie niciodata atins, cautarea blocandu-se intr-un maxim local.

Iterative Deepening Search = Căutarea cu nivel iterativ

Vezi Strategia căutării cu nivel iterativ

Iterative Deepening Search Strategy = Strategia căutării cu nivel iterativ

Strategia de cautare in adancime cu nivel iterativ elimina multe din dezavantajele cautarii pe nivel si a cautarii in adancime. Algoritmul de cautare in adancime cu nivel iterativ realizeaza la inceput o cautare in adancime in spatiul starilor cu o adancime maxima de cautare $\mathrm{AdMax}=1$. Daca starea finala nu a fost gasita, se reia cautarea in adancime cu $\mathrm{AdMax}=2$ si se continua in acest fel, crescand adancimea de cautare la fiecare iteratie. La fiecare iteratie algoritmul realizeaza o cautare in adancime completa cu adancimea de cautare egala cu valoarea curenta AdMax. Intre doua iteratii succesive nu se retine nici o informatie despre spatiul de cautare. Deoarece algoritmul de cautare in adancime cu nivel iterativ realizeaza de fapt o cautare pe nivel, el este garantat sa gaseasca solutia, daca aceasta exista, si, in plus, determina drumul cel mai scurt la solutie. Deoarece strategia este de adancime, numarul de stari generate la fiecare iteratie este mai mic decit cel in cazul cautarii pe nivel.

Litera K

Knowledge = Cunoştinţe

Cunoştinţele reprezintă o colecţie de fapte, evenimente, reguli şi convingeri organizate sistematic. Modul de reprezentare si organizare a cunostintelor este un element esential al oricarui sistem inteligent. Cunostintele dintr-un sistem bazat pe cunostinte trebuie sa posede urmatoarele caracteristici:

- Cunoştintele trebuie sa fie generale. Situatiile care prezinta proprietati comune trebuie sa poata fi reprezentate prin structuri simbolice comune si nu punctual, fiecare in parte. Daca cunostintele nu au aceasta proprietate cantitatea de memorie necesara descrierii acestora poate deveni imensa.
- Cunostintele unui sistem inteligent sunt in continua schimbare deoarece ele reflecta schimbarile din lumea inconjuratoare. Reprezentarea cunostintelor in sistem trebuie sa poata modela usor aceste schimbari si sa permita dezvoltarea incrementala a bazei de cunostinte.
- Reprezentarea cunostintelor trebuie sa faciliteze achizitia lor. Multe din cunostintele necesare unui sistem inteligent sunt cunostinte greu sau imposibil de formalizat si ele trebuie obtinute de la oamenii care le poseda. O reprezentare poate facilita sau ingreuna acest proces.
- Cunostintele trebuie sa poata fi utilizate in orice instanta a problemei de rezolvat, chiar daca sunt incomplete sau partial incorecte.
- Cunostintele intr-un sistem inteligent trebuie sa fie organizate intr-o structura care sa corespunda modului in care acestea vor fi utilizate.
- Cunostintele trebuie sa fie astfel reprezentate, organizate si utilizate incit sa permita transparenta sistemului. Expertul sau utilizatorul trebuie sa poata inspecta cunostintele utilizate in rezolvarea unei probleme si inferentele pe baza carora problema a fost rezolvata.

Aceste caracteristici pun in evidenta diferenta existenta intre cunostinte si date. Feigenbaum si McCorduck ilustreaza aceasta diferenta prin urmatorul exemplu. Un medic care trateaza un pacient foloseste cunostinte si date. Datele sunt reprezentate de

fisa pacientului: simptome, boli anterioare, tratament prescris, reactie la tratament, etc. Cunostintele utilizate in tratarea pacientului reprezinta tot ceea ce medicul a invatat in facultate si in decursul intregii lui cariere prin practica, studiu, experienta, in legatura cu modul de vindecare a bolii. Aceste cunostinte se refera la fapte, teorii, tratament si, cel mai important, la cunostinte euristice. Astfel, cunostintele necesita si includ utilizarea datelor dar sunt mai mult decit acestea.

Vezi şi sisteme bazate pe cunoştințe.

Knowledge-based System = Sistem bazat pe cunoştințe

Un *sistem bazat pe cunoştințe* este un sistem de prelucrarea cunoştințelor pentru rezolvarea problemelor dintr-un anumit domeniu particular sau aplicație pe baza executării inferențelor. În general, un astfel de sistem este format dintr-o bază de cunoştințe şi un motor de inferență.

La inceputul anilor '70 s-a produs o modificare fundamentala in modul de abordare si dezvoltare a sistemelor de inteligenta artificiala. Cercetatorii au ajuns la concluzia ca realizarea unui sistem inteligent capabil sa rezolve probleme reale, complexe, necesita un volum substantial de *cunostinte* specifice domeniului. In legatura cu aceasta idee a fost frecvent citata maxima lui Francis Bacon: "Scientia et potentia in idem coincidunt." Perspectiva importantei cunostintelor in programele de inteligenta artificiala este sustinuta si de observatia ca un specialist intr-un anumit domeniu nu poate lucra performant in alte domenii, oricit de puternica ar fi capacitatea lui de rationament. Descoperirea rolului cunostintelor specifice domeniului in rezolvarea unei probleme dificile a generat o noua conceptie a sistemelor de inteligenta artificiala: *sistemele bazate pe cunostinte*. Eduard Feigenbaum a rezumat aceasta noua conceptie intr-o lucrare prezentata la "The International Joint Conference on Artificial Intelligence" in 1977. El a pus in evidenta faptul ca adevarata putere de rezolvare a unui program este determinata in primul rind de cantitatea de cunostinte pe care o poseda si numai in al doilea rind de modalitatile de rationament general utilizate.

Sistemul expert DENDRAL este unul din primele exemple de sisteme bazate pe cunostinte. El a fost caracterizat de Feigenbaum ca "o masina de aplicare a cunostintelor." Sistemul MYCIN, sistem expert pentru diagnosticarea infectiilor bacteriene ale singelui, a carui dezvoltare a inceput la Stanford University in jurul anilor '74-'75, este un alt exemplu de sistem care a pus in evidenta rolul important al cunostintelor specifice domeniului. Incepand din aceasta perioada si pana in prezent, toata comunitatea cercetatorilor in domeniul inteligentei artificiale a recunoscut rolul esential al cunostintelor in rezolvarea inteligenta a problemelor.

Vezi şi cunoştințe.

Knowledge Engineering = Ingineria cunoştințelor

Ingineria cunostintelor se ocupa de studiul metodelor si tehnicilor de achizitie, reprezentare si organizare a cunostintelor in sistemele bazate pe cunostinte. Ingineria cunostintelor incearca sa rezolve una dintre problemele considerate cheie din punct de vedere al limitarii timpului de dezvoltare a unui sistem bazat pe cunostinte: transferul cunostintelor specializate ale expertului uman in sistemul bazat pe cunostinte. Feigenbaum defineste ingineria cunostintelor dupa cum urmeaza.

> *"Ingineria cunostintelor practica arta de a utiliza principiile si instrumentele cercetarilor de inteligenta artificiala in rezolvarea problemelor aplicative care necesita cunostinte experte. Aspectele tehnice ale achizitiei acestor cunostinte, ale reprezentarii acestora si ale utilizarii lor adecvate pentru construirea si explicarea liniilor de rationament, sunt probleme importante in proiectarea sistemelor bazate pe cunostinte. Arta de a construi agenti inteligenti este in acelasi timp o parte din si o expresie a artei programarii. Este arta de a construi programe de calcul complexe care reprezinta si rationeaza cu cunostintele lumii inconjuratoare."*

Litera M

Monotonicity = Monotonicitate

Minsky spune:

> "Monotonicitate: *Chiar dacă formulăm restricții de relevanță, sistemele logice prezintă o problemă dacă sunt utilizate în inteligență artificială. În orice sistem logic, toate axiomele sunt cu necesitate permisive - ele permit obținerea de noi date inferate. Fiecare axioma adaugată înseamnă mai multe teoreme şi nici una nu poate fi eliminată. Nu există nici o modalitate de adăugare a unei informații care să spună că anumite concluzii nu trebuie*

inferate! Spunând mai simplu: dacă adoptăm un numar suficient de axiome pentru a deduce ceea ce este necesar, deducem cu mult prea multe aserţiuni suplimentare. Dar dacă încercăm să schimbăm acest lucru prin adăugarea unor axiome de relevanţă, tot producem toate aserţiunile nedorite irelevante.

Deoarece logicienii nu sunt preocupaţi de sisteme care vor fi extinse ulterior, ei pot proiecta axiome care permit numai inferarea concluziilor pe care le doresc. În dezvoltarea sistemelor de inteligenţă artificială situaţia este diferită. Trebuie să învăţăm ce caracteristici ale unei situaţii sunt importante şi ce fel de deducţii nu trebuie luate în considerare."

Minsky mai adaugă de asemenea că cerinţa de consistenţă a logicii clasice nu poate fi întotdeauna aplicată în rezolvarea problemelor din inteligenţă artificială, această cerinţă neputând fi întotdeauna atinsă şi, de altfel, nici de dorit, datorită caracterului problemelor de inteligenţă artificială, de multe ori incomplet definite.

A se vedea şi raţionament nemonoton şi logici nemonotone.

Litera N

Node Depth = Adâncimea unui nod

Intr-o reprezentare a solutiei problemei prin spatiul starilor *adancimea unui nod* se defineste astfel:

1. Adancime (S_i) = 0, unde S_i este nodul stare initiala,
2. Adancime (S) = Adancime (S_p) + 1, unde S_p este nodul predecesor nodului S.

Nonmonotonic Reasoning = Raţionament nemonoton

În *raţionamentul nemonoton*, regulile de inferenţă sunt extinse astfel încât să permită raţionamentul pe baza cunoştinţelor incomplete sau contradictorii. Un astfel de model oferă instrumente capabile să extindă în mod consistent o mulţime incompletă de convingeri despre lume, să elimine contradicţiile în mod preferenţial şi să revizuiască convingerile în cazul obţinerii de noi cunoştinţe.

Raţionamentul nemonoton este strâns legat de dorinţa realizării raţionamentelor de bun simţ, cotidiene, în inteligenţa artificială. Termenul "raţionament nemonoton" poate fi atribuit probabil lui Minsky. Minsky defineşte informal noţiunea de cadru (care nu are legatură cu problema cadrului) şi spune că:

"Un cadru este o structură de date pentru reprezentarea unei situaţii stereotipice, cum ar fi cazul în care ne aflăm într-o camera sau în care mergem la o petrecere. Diverse tipuri de informaţii sunt ataşate fiecărui cadru. Unul dintre aceste tipuri este modul în care se foloseşte cadrul. Alt gen de informaţie se referă la ceea ce presupunem că se va întâmpla în continuare. şi altul se refera la ceea ce trebuie să facem dacă aceste presupuneri nu se confirmă."

Enunţurile "ceea ce presupunem că se va intâmpla" şi "ceea ce trebuie să facem dacă presupunerile nu se confirma" reprezintă de fapt un fel de reguli de raţionament nemonoton.

A se vedea şi logici nemonotone şi monotonicitate.

Nonmonotonic Logics = Logici nemonotone

Logicile nemonotone sunt o formalizare a raţionamentului nemonoton. Formalizarea raţionamentului nemonoton aşa cum este cunoscută la ora actuală a început în anii 1975-1976 şi primele lucrări au fost publicate în perioada 1977-1979. O primă lucrare importantă din această perioada este lucrarea în care Reiter propune o regulă a negaţiei numită *ipoteza lumii inchise*. Ipoteza lumii închise se aplică teoriilor logice Horn şi afirmă că dacă nu se poate demonstra un atom p atunci se poate presupune că p este fals, deci **non** p este adevărat. Un exemplu tipic de ipoteză a lumii închise este acela al unei baze de date. Numai informaţiile pozitive despre problemă trebuie reprezentate explicit în baza de date, informaţiile negative fiind inferate pe baza lipsei celor pozitive. Alt exemplu de aplicare a ipotezei lumii închise este limbajul Prolog, menţionat anterior, şi interpretarea operatorului **not** ca "negaţie prin insucces" (reuşita lui **non** p înseamna insuccesul demonstrării lui p).

A doua lucrare importantă din acea perioadă este cea a lui Clark care leaga negația de partea **only if** a instrucțiunilor **if** într-un program logic. Instrucțiunile **if-and-only-if** formează o teorie în care atomii negați pot fi demonstrați cu ajutorul unui demonstrator de teoreme complet. Aceste două reguli ale negației (cea a lui Reiter şi cea a lui Clark) sunt primele formalizări ale raționamentului nemonoton. În 1977 McCarthy dezvoltă teoria circumscrierii iar în 1978 Reiter publică un material preliminar despre raționamentul implicit.

O primă categorie de abordări nemonotone îmbogățesc limbajul logicii, fie cel al logicii propozițiilor fie cel al logicii predicatelor, cu construcții suplimentare care să surprindă caracteristica raționamentului nemonoton. Unele dintre aceste abordări introduc operatori modali speciali, de exemplu modalitatea **M** a lui McDermott şi Doyle, modalitatea **L** a lui Moore. Alte abordări din aceeasi categorie adauga operatori logici suplimentari, cum ar fi operatorul **:/** din regulile implicite ale lui Reiter, preluat în toate dezvoltările ulterioare bazate pe această teorie, sau operatorii de consecintă logică condițională sau preferențială.

A doua categorie de abordări pastrează nemodificat formalismul logic de bază, respectiv calclul cu predicate, şi schimbă numai modul în care logica este folosită. În această categorie se încadrează teoria circumscrierii cu toate dezvoltările ei ulterioare, modelul lui Poole şi cel al lui Gardenfors şi Makinson.

Dintr-o altă perspectivă se pot distinge teorii care au în comun noțiunea de punct fix utilizată în definirea extensiilor acestor teorii, deci în specificarea modului de completare a unui set de formule ce reprezintă cunoştințele incomplete ale unui agent despre lume. Să presupunem că un agent rațional ideal doreşte să decidă ce mulțime de propoziții crede. Să presupunem că agentul are o mulțime inițială de convingeri şi un set de reguli care permit derivarea a noi convingeri. Atunci răspunsul natural ar fi ca agentul să considere închiderea logică a mulțimii inițiale de convingeri pe baza setului de reguli pe care le poseda. Cu toate acestea, nu este întotdeauna clar cum trebuie definită noțiunea de "închidere logică". Dacă modelul este logica propozițiilor sau logica predicatelor de ordinul intâi atunci este clar ce înseamna "închidere logică". Dacă considerăm însă cazul logicilor nemonotone, regulile pot avea forme mai complicate. Acceptabilitatea unei reguli poate depinde, în particular, şi de prezența sau absența unor convingeri ale agentului. O modalitate de a construi închideri logice în astfel de situații este aceea de a utiliza punctele fixe. În acest caz agentul presupune o mulțime inițială de convingeri şi refera toate regulile de derivare a noi convingeri la această mulțime inițială. Dacă, pe baza mulțimii inițiale de convingeri şi a regulilor aplicate, agentul obtine o mulțime de convingeri care coincide cu cea presupusă anterior, atunci această mulțime este, din punctul lui de vedere, "justificată" şi poate reprezenta un candidat pentru mulțimea totală de convingeri pe care agentul le poate avea. Printre teoriile de punct fix se pot menționa cele implicite şi cele modale.

Se poate identifica, de asemenea, o clasă de teorii ce provin dintr-o aceeaşi viziune asupra raționamentului nemonoton, respectiv teoriile minimale şi preferențiale. În logica traditională, semnificația unei formule este mulțimea interpretărilor care satisfac acea formulă, sau mulțimea ei de modele. Interpretarea poate fi văzută fie ca o atribuire de valori de adevăr atomilor din formulă fie ca o pereche <interpretare Kripke, lume> în abordarea lumilor posibile. Teoriile minimale şi preferențiale obțin o logică nemonotonă prin focalizarea atenției numai asupra unui subset al acestor modele, respectiv acelea care sunt minimale sau preferate dupa un anumit criteriu. Motivul pentru care această abordare face ca logica rezultată să fie nemonotonă este următorul. În logica clasică $\alpha\Phi \subseteq \beta$ dacă *beta* este adevărată în toate modelele lui *alfa*. Deoarece toate modelele formulei $\alpha \wedge \delta$ sunt de asemenea modele ale lui *alfa*, rezultă că $\alpha \wedge \delta\Phi \subseteq \beta$ şi astfel logica clasică este monotonă. În abordarea minimală sau preferențială, $\alpha\Phi \subseteq \beta$ dacă *beta* este adevărată în toate modelele minimale, respectiv preferențiale ale lui *alfa*, dar $\alpha \wedge \delta$ poate avea modele preferate (minimale) care nu sunt şi modele preferate (minimale) ale lui *alfa*. De fapt, clasa modelelor preferate ale lui $\alpha \wedge \delta$ şi cea a modelelor preferate ale lui *alfa* pot fi complet disjuncte. Astfel se obtine o logică nemonotonă.

O altă dimensiune posibilă în clasificare este tipul de raționament nemonoton modelat de diverse teorii. Anumite abordări încearcă modelarea atitudinii introspective a unui agent şi a stării cunoştințelor sale despre lume. În această categorie pot fi incluse logicile modale nemonotone cât şi toate variantele de circumscriere. Alte teorii surprind raționamentul implicit, deci încearcă să reprezinte cunoştințe în general adevărate. Această abordare, inițiată de teoria lui Reiter, poate fi considerată comună şi teoriilor preferențiale.

Litera O

OPEN = FRONTIERA

In implementarea strategiilor de cautare se folosesc de obicei doua liste:

- FRONTIERA - lista nodurilor evaluate
- TERITORIU - lista nodurilor expandate

In parcurgerea spatiului de cautare un nod poate fi:

- *necunoscut* - nodul apartine partii neexplorate a spatiului de cautare,
- *evaluat* - nodul este cunoscut dar fie nu se cunoaste nici un succesor al lui, fie se cunosc numai o parte din succesorii lui,
- *expandat* - nodul este cunoscut si, in plus, se cunosc toti succesorii lui.

Lista FRONTIERA reprezinta deci frontiera spatiului de cautare parcurs (explicitat) spre partea necunoscuta a spatiului de cautare.

Problem Decomposition Space = Spațiul descompunerii problemei în subprobleme

Exista probleme a caror rezolvare poate fi convenabil reprezentata printr-o tehnica numita *reducerea problemei la subprobleme*. Caracteristica comuna a acestei clase de probleme este aceea ca orice problema pusa de un obiect candidat la solutie poate fi vazuta ca o conjunctie de subprobleme ce pot fi rezolvate independent unele de altele. Rezolvarea problemelor din aceasta clasa poate fi abordata in urmatorul mod: se descompune problema in subproblemele care trebuie rezolvate, subproblemele se descompun la rindul lor in alte subprobleme si asa mai departe, pana cand se obtine o descompunere a problemei initiale in subprobleme elementare, adica banal de rezolvat.

Spatiul de cautare a unei astfel de rezolvari a problemei are forma unui graf SI/SAU. Un graf SI/SAU este un caz particular al unui hipergraf. Un *hipergraf* este format dintr-o multime de noduri si o multime de hiperarce definite prin perechi ordonate in care primul element este un nod din multimea de noduri a hipergrafului si cel de al doilea element este o submultime de noduri. Un graf obisnuit este un caz particular al unui hipergraf in care cel de al doilea element al hiperarcelor este o multime formata dintr-un singur element.

O reprezentare a solutiei problemei prin grafuri SI/SAU este formata dintr-un triplet

(S_i , O , P_e)

cu urmatoarea semnificatie:

- S_i reprezinta descrierea problemei initiale,
- O reprezinta multimea de operatori de transformare (descompunere) a problemei in subprobleme,
- P_e reprezinta descrierea unei multimi de probleme elementare.

Vezi si grafuri SI/SAU

Litera Q

Qualification Problem = Problema calificării

Problema calificării apare deoarece este dificil, în lumea reală, să se specifice toate condițiile în care este posibilă executarea unei acțiuni. De exemplu, mutarea unui obiect poate fi împiedicată dacă obiectul este solidar cu podeaua, dacă trebuie să treacă printr-un spațiu mai mic decât dimensiunea obiectului, etc. Raționamentul implicit propune soluții pentru tratarea coerentă şi eficientă a acestor dificultăți de reprezentare.

Litera R

Ramification Problem = Problema ramificării

Problema ramificării se referă la proliferarea consecințelor implicite ale unei acțiuni şi la dificultatea reprezentării tuturor acestor consecințe. De exemplu, dacă obiectul care se mută este acoperit de praf, mutarea obiectului va implica şi mutarea particulelor de praf de pe obiect sau obiectul poate acoperi o gură de aerisire după mutare. Raționamentul implicit propune

soluții pentru tratarea coerentă şi eficientă a acestor dificultăți de reprezentare.

Litera S

Search Space = Spațiu de căutare

Descrierea initiala a problemei si a obiectelor candidate la solutie obtinute pe parcursul rezolvarii, deci structurile simbolice care specifica universul problemei, pot fi asimilate cu o *multime de stari*. Multimea de operatori (reguli) de transformare indica modul de transformare a universului problemei dintr-o stare initiala intr-o stare finala. *Starea initiala* descrie conditiile initiale ale problemei iar *starea finala* reprezinta solutia problemei. Starea finala poate fi definita explicit, prin descrierea acesteia, sau implicit, printr-o multime de conditii pe care o stare trebuie sa le satisfaca pentru a fi stare finala, adica solutie a problemei. Rezolvarea problemei poate cere fie determinarea starii finale, fie stabilirea intregului drum de la starea initiala la starea finala. Multimea starilor investigate pana in momentul ajungerii in starea finala formeaza *spatiul de cautare* a solutiei problemei.

Solved node (in an AND/OR Graph) = Nod rezolvat (într-un graf SI/SAU)

Intr-un graf SI/SAU un *nod* este *rezolvat* daca:

1. este un nod terminal etichetat cu o problema elementara
2. este un nod SI si toti succesorii lui sunt noduri rezolvate
3. este un nod SAU si cel putin un succesor al acestuia este rezolvat

State Space = Spațiul stărilor

O reprezentare a solutiei problemei prin spatiul starilor este formata dintr-un triplet

(S_i , O , S_f)

cu urmatoarea semnificatie:

- S_i reprezinta multimea starilor initiale,
- O reprezinta multimea de operatori posibil de aplicat asupra starilor universului problemei pentru a ajunge in noi stari; in fiecare stare data, numai o parte din operatori sunt legal aplicabili,
- S_f reprezinta multimea starilor finale sau stari scop. Multimea starilor finale poate contine si o singura stare.

In reprezentarea solutiei problemei prin spatiul starilor, spatiul de cautare are forma unui graf orientat in care nodurile sunt identificate prin stari, iar arcele reprezinta aplicarea operatorilor pentru a transforma o stare in starea urmatoare. O *solutie a problemei* este o secventa de operatori care transforma starea initiala in stare finala si reprezinta o cale intre aceste doua stari in graf. Graful spatiului de cautare este specificat implicit de reprezentare prin tripletul (S_i , O , S_f). Pe parcursul avansului in cautare o portiune din acest graf devine explicita, portiunea din graful spatiului de cautare astfel construita reprezentind partea explorata a spatiului de cautare.

Statistical Reasoning = Raționament statistic

În *raționamentul statistic*, reprezentarea cunoştinţelor este extinsă cu o măsură numerică a încrederii sau a incertitudinii asociată faptelor. Prin definirea unor reguli de combinare a acestor măsuri ale incertitudinii, raționamentul statistic permite rezolvarea problemelor care implică cunoştinţe nesigure, incerte. Unii cercetători consideră cunoştinţele incerte tot ca o formă de cunoştinţe incomplete şi privesc raționamentul nemonoton şi cel statistic ca soluții alternative în rezolvarea problemelor. John McCarthy, în articolul sau "History of Circumscription", susține ideea conform căreia modelele de raționament nemonoton şi cele de raționament statistic sunt formalisme complementare şi nu rivale. Metodele nemonotone sunt necesare pentru a reprezenta o lume în schimbare la care, ulterior, se pot adăuga probabilități sau măsuri ale incertitudinii conform modelelor statistice.

Syllogism = Silogism

Se numeste *silogism* un tip de rationament deductiv alcatuit din trei judecati:

1. *premisa majora* sau termen major care contine predicatul concluziei,
2. *premisa minora* sau termen minor care contine subiectul concluziei, si
3. *concluzia*, derivata cu necesitate din primele doua.

Legatura intre (1) si (2) este mijlocita de *termenul mediu* care figureaza in ambele premise dar nu si in concluzie. Termenul de silogism impreuna cu definitia de mai sus au fost date de Aristotel, acesta fiind considerat fondatorul teoriei deductiei. Ulterior deductia a fost dezvoltata de Descartes, Leibniz si de logica simbolica, in care silogismul ia forma regulii de inferenta Modus Ponens.

Litera T

Tentative Search Strategy = Strategie de căutare tentativă

Strategiile de cautare *tentative* sunt strategii capabile să revină în stări anterior parcurse în căutare. La aplicarea unui operator intr-o stare curenta se memoreaza aceasta stare astfel incit procesul de cautare sa poata ulterior reveni in starile anterioare aplicarii operatorilor.

Strategii de căutare tentative sunt, de exemplu, căutarea pe nivel sau căutarea în adâncime.

Turing Test = Testul Turing

Cel mai celebru criteriu de apreciere a inteligentei unui program este *testul Turing*. In 1950, Alan Turing (1912-1954), celebru matematician britanic si unul dintre intemeietorii stiintei calculatoarelor, a propus un test pentru a determina daca o masina poate avea sau nu un comportament inteligent. In articolul sau "Computing Machinery and Intelligence", Turing a pus pentru prima oara problema posibilitatii simularii gandirii umane cu ajutorul calculatorului. In acelasi articol, Turing descrie si testul care, afirma el, poate discerne intre o comportare inteligenta si una neinteligenta.

Testul Turing consta in principiu din urmatorul experiment. Un om comunica prin intermediul unui terminal al calculatorului cu alte doua terminale, situate in camere invecinate, punand intrebari si obtinand raspunsuri. Raspunsurile sunt date de o persoana la unul din cele doua terminale ascunse celui de la care se intreaba si de un program la celalalt terminal. Daca persoana care intreaba nu poate stabili in urma dialogului care este terminalul de unde i-a raspuns omul si care este cel de la care i-a raspuns programul atunci programul are o comportare inteligenta. Programul poate fi astfel facut incit sa incerce sa pacaleasca pe cel care intreaba. De exemplu, daca programul este intrebat cat fac 12.120*32.425, se asteapta citeva minute pana se primeste raspunsul.

Pana la ora actuala nici un program nu a reusit sa treaca testul Turing. Cu toate acestea, testul Turing sugereaza o masura a performantelor unui program daca se considera domenii restrinse ale discursului. De exemplu, un program de jucat sah poate ajunge sa aiba performante similare cu cele ale unui maestru iar oponentul sa nu stie daca joaca sah cu o persoana sau cu un program. Si in alte domenii este posibil sa se compare performantele rezolvarii unei probleme de catre un program cu cele ale unui expert in domeniu, atat din punct de vedere al calitatii solutiei cat si din punct de vedere al vitezei de rezolvare.

Testul Turing este insa vulnerabil, in ciuda sarmului lui intelectual, din mai multe motive. In primul rand testul verifica numai capacitatile pur simbolice de rezolvare a problemelor, neluind in considerare aspecte cum ar fi perceptia sau dexteritatea manuala. Pe de alta parte, exista posibila obiectie ca acest test impune ca inteligenta masinilor sa modeleze inteligenta umana. Unii cercetatori sustin ca inteligenta masinilor este o forma diferita de inteligenta si ca este o gresala sa incercam a o evalua in termenii inteligentei umane. Toate aceste intrebari raman pentru moment fara un raspuns definitiv, unele depasind cu mult sfera inteligentei artificiale.

Litera U

Uninformed Search Strategies = Strategii de cautare neinformate

Strategiile de cautare *neinformate*, numite şi strategii de căutare de bază, consideră starile urmatoare de inspectat dupa o ordine arbitrara, anterior stabilita. Exemple de astfel de strategii sunt:

- căutarea pe nivel - se inspecteză stările în ordinea aproprierii față de starea curentă, deci cele mai aproape mai întâi;
- căutarea în adâncime - se inspecteză stările în ordinea depărtării față de starea curentă, deci cele mai departe mai întâi;
- căutarea cu backtraking - asemănătoare cu căutarea în adâncime dar starile anterioare la care se poate reveni in timpul cautarii se afla numai pe calea curenta intre starea initiala si starea finala.

Unsolvable node (in an AND/OR Graph) = Nod nerezolvabil (într-un graf SI/SAU)

Intr-un graf SI/SAU un *nod* este *nerezolvabil* daca:

1. este un nod terminal etichetat cu o problema neelementara, deci care nu se mai poate descompune in subprobleme
2. este un nod SI cu cel putin un succesor nerezolvabil
3. este un nod SAU cu toti succesorii nerezolvabili.

O *solutie a problemei* reprezentata prin grafuri SI/SAU este secventa de operatori de descompunere care determina ca nodul problema initiala sa devina rezolvat. Altfel spus, solutia problemei este subgraful SI/SAU care face ca nodul problema initiala sa devina rezolvat.