BLIA-MAS Laborator 06

Cuprins:

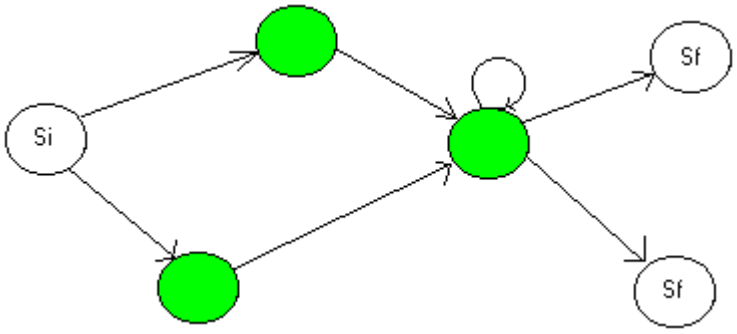
a. problem solving agents
b. exemple de probleme de cautare
c. agentul si baza de cunostinte
d. mediul lumii wumpus
e. decizii si comportare in mediul lumii wumpus - agent in mediul lumii wumpus
f. baza de cunostinte
g. agenti reflexivi -agenti bazati pe un model sau scop

Problem-Solving Agents

In acest laborator vom discuta despre modul de rezolvare a diferitelor probleme de catre agenti. Rezolvarea unei probleme de catre un agent va reprezenta de fapt atingerea unui scop (rezultat final) de catre agentul respectiv. Agentii inteligenti trebuie sa se comporte astfel incat sa-si atinga scopul. Acest scop va fi realizat prin trecerea agentului printr-o succesiune de stari optime. Atunci cand unui agent i se va da o problema de rezolvat, agentul ar trebui sa identifice scopul final (rezolvarea problemei) si sa gaseasca o cale, o succesiune de stari, care sa-l ajute sa obtina acel scop.

Formularea scopului final reprezinta primul pas in rezolvarea problemei curente. Acest pas se va face pe baza informatiilor disponibile, pe situatia la momentul respectiv.

Vom considera scopul final ca fiind un set de stari, stari in care scopul este satisfacut. Actiunile reprezinta tranzitiile de la o stare la alta. Putem reprezenta aceste stari si tranzitii sub forma unui graf: starile=nodurile grafului, iar tranzitiile ca arcele care leaga graful.



Un punct important in rezolvarea problemei, este definirea tipurilor de actiuni pe care agentul poate sa le execute la un moment dat. Formularea problemei reprezinta modalitatea de a specifica ce actiuni si stari ar trebui considerate.

Un agent care la un moment dat are posibilitatea sa treaca in mai multe stari (de valori necunoscute), din starea in care se afla, fara a sti rezultatul imediat al actiunilor sale, poate sa decida ce sa faca prin analizarea diferitelor secvente de actiuni care il vor duce in stari (cu valoare cunoscuta), si apoi sa o aleaga pe cea mai optima. Acest proces se numeste cautare.

Un algoritm de cautare are ca intrare o problema si ca rezultat o solutie a problemei respective. Aceasta solutie reprezinta de fapt o secventa de actiuni pe care agentul trebuie sa le execute. Aceasta faza se numeste faza de executie. Cand o solutie a fost realizata, agentul va incerca sa obtina o noua solutie pe care s-o rezolve.

Vom prezenta mai jos un algoritm simplu pentru problem-solving agent (*Russel, J., Norvik, P. - Artificial Intelligence A modern Approach*)

function SIMPLE-PROBLEM-AGENT(*p*) **returns** an action
    inputs: *p*, a percept
    static: *s*, an action sequence, initially empty
        g, a goal, initially null
        problem, a problem formulation

    state – UPDATE-STATE(*state, p*)
    if *s* is empty **then**
        g – FORMULATE-GOAL(state)
        problem – FORMULATE-PROBLEM(*state, g*)
        s – SEARCH(problem)
    action – RECOMMENDATION(*s, state*)
    s – REMAINDERS(*s, state*)
    return *action*

RECOMMENDATION – preia prima actiune din secventa
REMAINDERS – returneaza restul

Tipuri de probleme:

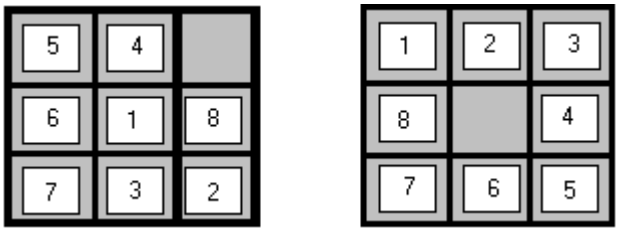
a. **single-state problem**, atunci cand un agent stie in ce stare se afla, ce actiuni poate intreprinde, rezultatul fiecarei actiuni, si poate calcula exact starea in care se va afla dupa un numar finit de actiuni pe care le va intreprinde
b. **multiple-state problem**, cand agentul stie rezultatul actiunilor lui, dar are foarte putine cunostinte asupra lumii inconjuratoare.
c. **contingency problem**, cand predinctia exacta a rezultatului nu este posibila si trebuie calculat intregul arbore de solutii
d. exploration problem, cand agentul nu are nici o informatie referitoare la efectul actiunilor sale

Exemple de probleme de cautare

Definitii:

Starea initiala: starea in care agentul stie ca se afla la momentul zero
Operatori: un set de actiuni posibile care sunt accesibile agentului
Functia de succesiune S: dandu-se o stare particulara *x, S(x)* intoarce setul de stari care sunt accesibile din *x* printr-o singura actiune
Spatiul unei stari a problemei: multimea starilor accesibile din starea initiala printr-o succesiune finita de actiuni
Drum:orice succesiune de actiuni dintr-o stare intr-alta
Testul de scop: se aplica unei singure stari si trebuie sa determine daca in starea respectiva s-a atins scopul problemei
Costul drumului: este o functie care asigneaza un cost unui drum dat; reprezinta suma costurilor actiunilor individuale care apartin drumului respectiv.

Problema puzzle

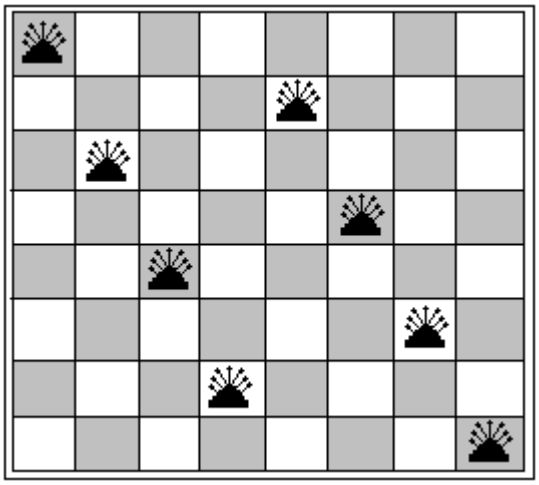


Starea initiala Starea scop

(*Russel, J., Norvik, P. - Artificial Intelligence A modern Approach*)

- **Stari**: descrierea unei stari specifica locatia fiecarei celule cu numere intr-una din cele 9 lacasuri. Pentru usurinta ar trebui inclusa si locatia celulei goale.
- **Operatori**: celula goala se muta la stanga, dreapta, sus, jos
- **Testul de scop**: starea se identifica cu scopul atunci cand se ajunge la configuratia din imaginea de mai sus
- **Costul drumului**: fiecare pas costa 1 unitate, deci costul drumului reprezinta lungimea drumului.

Problema celor 8 regine



Solutie relativa a problemei
(*Russel, J., Norvik, P. - Artificial Intelligence A modern Approach*)

- *Testul de scop**: cele 8 regine sunt pe tabla de sah, si nu se ataca***
- **Costul drumului**: zero
- **Starile**: orice aranjament de 0 sau 8 regine pe tabla de sah
- **Operatori**: adaugarea unei regine intr-o celula

Conform acestei formulari, avem 64^8 secvente posibile de investigat. O alta varianta ar fi sa verificam daca in locatia in care vom plasa regina exista posibiltiatea ca aceasta din urma sa fie atacata de reginele deja dispuse pe tabla de sah.

- **Stari**: aranjamente de 0 pana la 8 regine care nu se ataca
- **Operatori**: plaseaza o regina in cea mai din stanga coloana libera in asa fel incat sa nu fie atacata de nici o alta regina

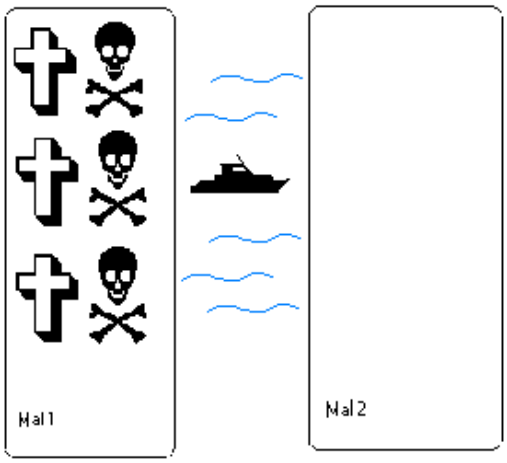
Problema criptarii aritmetice

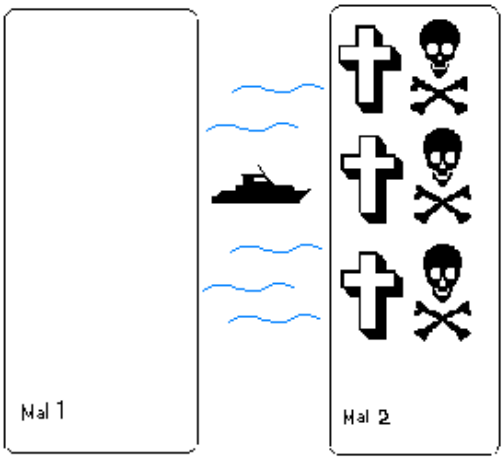
FORTY	Solutie:	29786	F=2, O=9, R=7, etc.
+ TEN		850	
+ TEN		850	
---		---	
SIXTY		31486	

(*Russel, J., Norvik, P. - Artificial Intelligence A modern Approach*)

- **Stari**: un puzzle criptoaritmetic cu litere inlocuite de cifre
- **Operatori**: se inlocuiesc toate aparitiile unei litere cu o cifra care nu a aparut deja in puzzel

- **Testul scop**: puzzelul contine doar cifre, care reprezinta suma exacta
- **Costul drumului**: zero. Toate solutiile sunt valide in egala masura.





Misionarii si canibalii

Stare initiala ***Stare finala***

Cu ☠ am notat canibalii, iar cu ✝ am notat misionarii.

- **Stari**: o stare consta dintro secventa ordonata de trei numere reprezentand numarul de misionari, canibali, si barci de pe un mal al raului, de unde pornesc. In cazul nostru (3, 3, 1)
- **Operatori**: din orice stare, operatorii posibili sunt sa fie transportati un misionar, un canibal, doi misionari, doi canibali, sau un canibal si un misionar, cu ajutorul barcii
- **Testul scop**: sa ajungem in starea (0, 0, 0)
- **Costul drumului**: numarul de traversari.

Agent si baza de cunostinte

Baza de cunostinte reprezinta un set de reprezentari de fapte despre lumea inconjuratoare. Fiecare reprezentare individuala se numeste propozitie. Propozitiile sunt reprezentate intr-un limbaj denumit limbaj de preprezentare cunostinte.

Componenta principala a unui agent care are o baza de cunostinta este baza de cunostinte (knowledge base, KB).

Interactionarea cu baza de cunostinte se face prin intermediul unor actiuni: TELL si ASK. TELL este pentru a adauga in baza de cunostinte iar ASK pentru a intreba. Modul de a determina ce urmeaza cui in baza de cunostinte se face cu ajutorul motorului de inferenta. Motorul de inferenta reprezinta componenta principala a unui agent care are o baza de cunostinte.
(*Russel, J., Norvik, P. - Artificial Intelligence A modern Approach*)
function KB-AGENT(*percept*) **returns** an *action*
static: *KB*, a knowledge base
t, a counter, initially 0, indicating time

TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
action – ASK (*KB*, MAKE-ACTION-QUERY(*t*))
TELL (*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
t – *t* + 1
return *action*

De fiecare data cand programul agentului este apelat, el spune bazei de cunostinte ce simte/vede referitor la lumea inconjuratoare. Apoi intreaba baza de cunostinte ce trebuie sa faca. Apoi agentul executa actiunea propusa de baza de cunostinte.

La orice moment de timp un agent (care are o baza de cunostinte) se poate afla in unul din urmatoarele nivele:

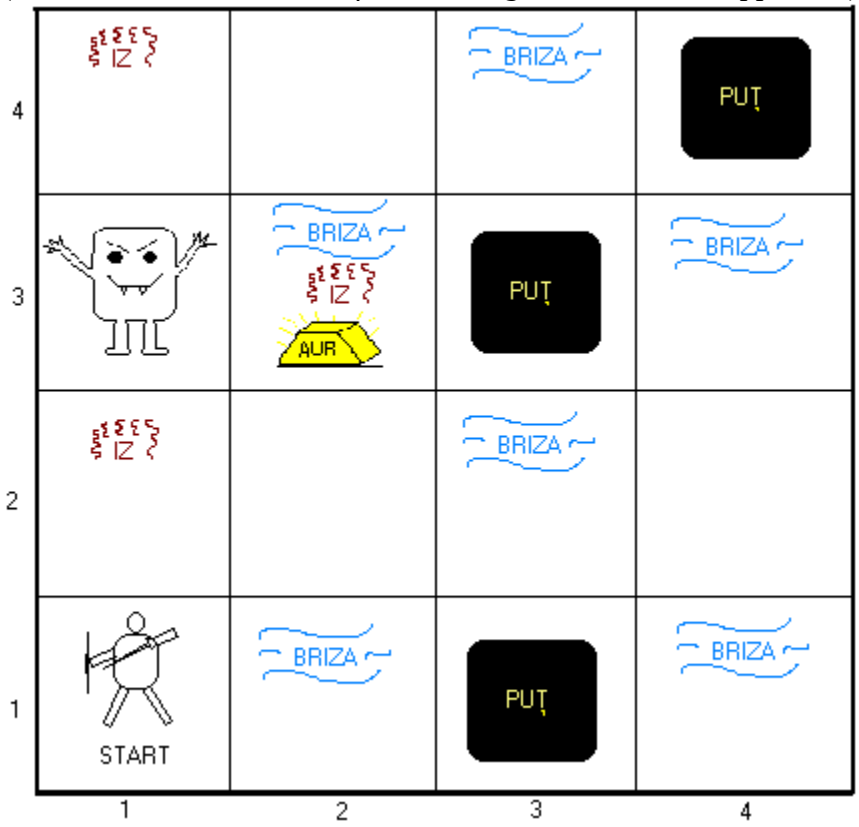
- nivelul cunostintelor; un agent se afla la acest nivel atunci cand simte/vede lumea inconjuratoare, el stie
- nivelul logic este nivelul la care cunostintele sunt codate in propozitii.
- nivelul implementarii; la acest nivel reprezentarile fizice devin propozitii de la nivelul logic.

Mediul lumii wumpus

Wumpus a fost un joc pe calculator, bazat pe un agent care trebuia sa exploreze incaperi conectate prin pasaje. Una dintre aceste camera continea un wumpus, o creatura infricosatoare, care omora pe oricine intra in camera respectiva. Alte camera contineau puturi adanci din care nu se mai putea iesi (cu exceptia wumpusului care era suficient de mare ca sa nu cada in put), si oricine cadea in ele murea. Alte camere aveau lazi cu aur. In camerele care erau adiacentei (nu pe diagonala) camerei in care se afla wumpusul, se

simte/percepe un iz specific. In camerele adiacente (nu pe diagonala) camerei in care se afla un put, se simte/percepe o briza batand. In camera unde se afla aurul, agentul percepea o stralucire. Cand wumpusul este omorat se va auzi in orice camera de pe harta un tipat infricosator. Aceasta lume este delimitata de un zid. Cand agentul va intra in zid va percepe o busitura. Modul de a percepe un agent va fi o lista de cinci simboluri. De exemplu daca in ceula respectiva avem un iz, o briza, o stralucire, dar nu avem busitura si nu avem tipat, atunci agentul va primi [Iz, Briza, Stralucire, Nimic, Nimic]. Agentul nu-si poate percepe locatia proprie.

(*Russel, J., Norvik, P. - Artificial Intelligence A modern Approach*)

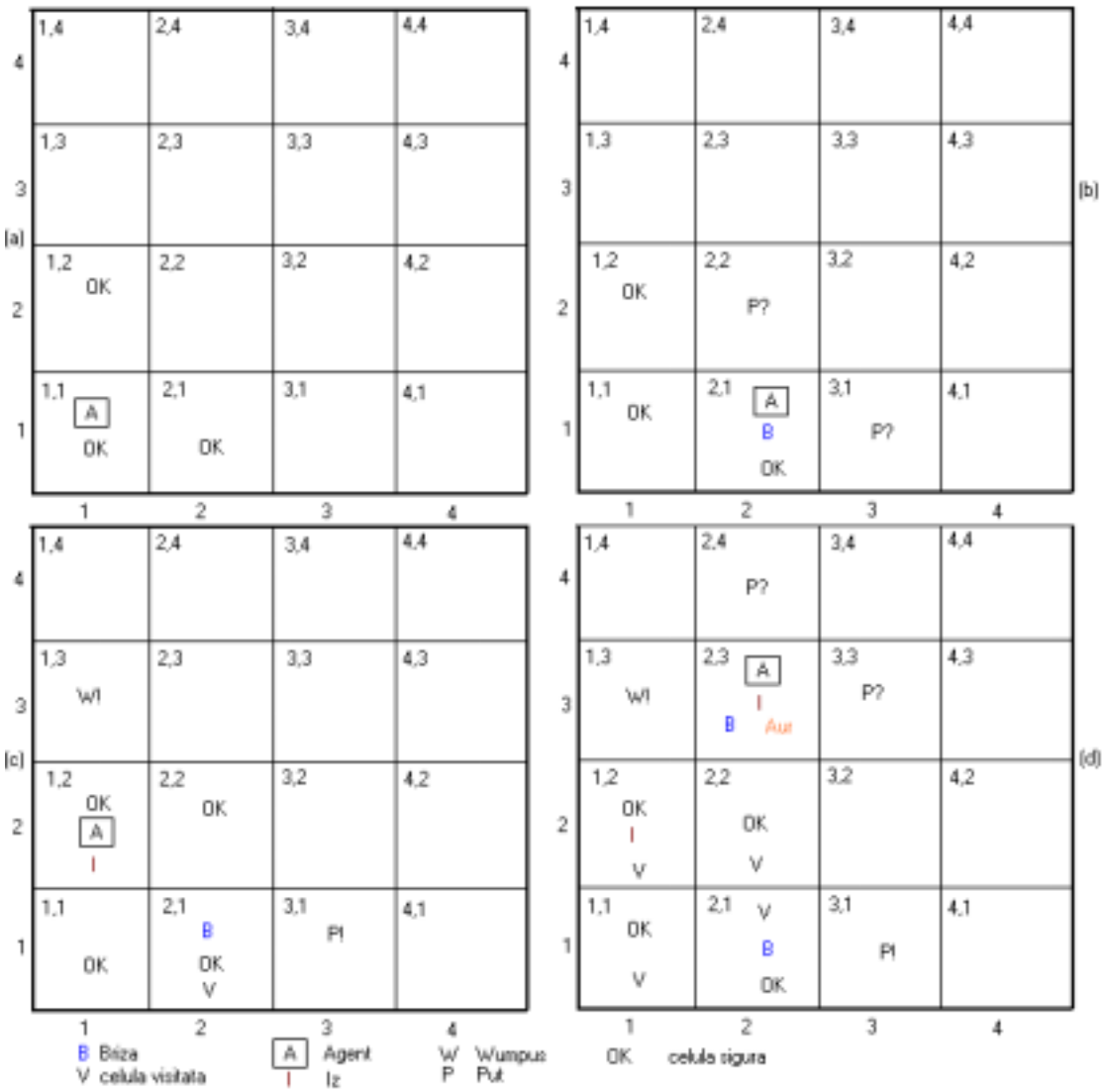


Avem urmatoarele actiuni posibilite pe care un agent ar putea sa le execute:

- sa se miste cu 90º la stanga,
- sa se miste cu 90º la dreapta,
- sa mearga inainte,
- sa apuce ceva de jos (un obiect care se afla in aceeasi casuta cu agentul)
- sa traga cu o sageata in directia in care este orientat agentul; sageata va continua pana cand fie loveste peretele, fie omoara wumpusul; agentul dispune de o singura sageata, deci doar prima actiune de a trage cu sageata are efect

Agentul va muri atunci cand intra intr-o celula cu un put sau c-un agent viu. Agentul poate sa intre intr-o celula c-un wumpus. Scopul agentului este sa gaseasca aurul si sa-l aduca la celula de start cat mai repede, fara a muri. Se dau 1000 de puncte pentru a ajunge cu aurul la celula de start, 1 punct penalitate pentru fiecare actiune intreprinsa si 10000 puncte penalitate atunci cand agentul este omorat.

Decizii si comportare in mediul lumii wumpus - agent in mediul lumii wumpus



(*Russel, J., Norvik, P. - Artificial Intelligence A modern Approach*)

(a) starea initiala, agentul percepe [Nimic, Nimic, Nimic, Nimic, Nimic]
(b) dupa o mutare, agentul va percepe [Nimic, Briza, Nimic, Nimic, Nimic]
(c) dupa doua mutari, agentul percepe [Iz, Nimic, Nimic, Nimic, Nimic]
(d) dupa inca doua mutari, agentul percepe [Iz, Briza, Stralucire, Nimic, Nimic]

Consideram starea initiala celula [1,1] (a). Deoarece nu avem nici o briza in celula [1,1], agentul poate infera ca [1,2] si [2,1] sunt celule sigure. Ele vor fi marcate cu Ok pentru a arata acest lucru.Din moment ce agentul este inca in viata, se poate infera ca celula [1,1] este OK. Un agent prudent se va muta doar intr-o celula apropiata care este OK. Sa presupunem ca agentul decide sa se mute in celula [2,1] (b). Agentul detecteaza o briza in [2,1], deci ar trebui sa fie un put intr-o ceula invecinata, ori [2,2], ori [3,1]. Notatia P? Indica un posibil put. Putul nu poate fi in [1,1]deoarece agentul a fost deja in aceasta ceula si nu a murit. In acest moment avem o singura celula marcata cu OK care nu a fost

visitata inca. Deci, agentul prudent se va intoarce in celula [1,1] si va merge in celula [1,2] (c). Agentul detecteaza un iz in [1,2] care il determina sa concluzioneze ca exista un wumpus in apropiere. Wumpusul nu poate fi in [1,1] (l-ar fi mancat pe agent din start), nu poate fi in [2,2] (agentul ar fi detectat un iz cand era in [2,1]). De aceea, agentul poate infera ca wumpusul se afla in [1,3] (vom folosi notatia W! pentru a arata acest lucru). Lipsa brizei in [1,2] determina agentul sa concluzioneze ca avem un put in [3,1] (de fapt lipsa brizei din [1,2] ne arata ca nu avem un put in [2,2], si deoarece avem o briza in [2,1] atunci singura posibilitate era ca putul sa fie in [3,1]). Avem in acest moment o singura celula sigura cu Ok, [2,2]. Agentul se muta in aceasta. Vom presupune ca agentul intoarce si se va muta in [2,3] (d).In [2,3] agentul va detecta o stralucire, deci va apuca aurul si se va indrepta spre casa. La intoarcere agentul va avea grija sa treaca doar prin celulele descoperite deja si care sunt insemnate cu OK.

Baza de cunostinte

Vom folosi urmatoarele simboluri pentru reprezentarea cunostintelor in baza de cunostinte. De exzemplu: $I_{1,2}$ reprezinta “Exista un iz in celula [1,2]”. Similar $B_{2,1}$ inseamna “Avem o briza in [2,1]”.

La un moment dat vom avea:

$\neg I_{1,1}$ $\neg B_{1,1}$
$\neg I_{2,1}$ $B_{2,1}$
$\neg I_{1,2}$ $\neg B_{1,2}$

Pentur inceput agentul va stii ca daca o celula nu are miros/iz, atunci nici una dintre celulele adiacente ei nu poate contine un wumpus.

R_1: $\neg I_{1,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$
R_2: $\neg I_{2,1} \Rightarrow \neg W_{1,1} \wedge \neg W_{2,1} \wedge \neg W_{2,2} \wedge \neg W_{3,1}$
R_3: $\neg I_{1,2} \Rightarrow \neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,2} \wedge \neg W_{1,3}$

Analog, daca avem un iz in celula [1,2] atunci ar trebui sa fie un wumpus in ceula [1,2] sau in celulele adiacente.
R_4: $I_{1,2} \Rightarrow W_{1,1} \wedge W_{1,2} \wedge W_{2,2} \wedge W_{1,3}$
Luam regula R1. Aplicam Modus Ponens cu $\neg I_{1,1}$ si obtinem

$\neg W_{1,1} \wedge \neg W_{1,2} \wedge \neg W_{2,1}$

Aplicam apoi Eliminarea lui SI, si se obtine

$\neg W_{1,1}$ $\neg W_{1,2}$ $\neg W_{2,1}$

Analog aplicam Modus Ponens pe regula R2 si apoi Eliminarea lui SI:

$\neg W_{1,1}$ $\neg W_{2,1}$ $\neg W_{2,2}$ $\neg W_{3,1}$

Aplicam regula rezolutiei, unde α este $W_{1,2} \cup W_{2,2} \cup W_{1,3}$ si β este $W_{1,1}$(obtinusem $\neg W_{1,1}$ mai sus):

$W_{1,2} \wedge W_{2,2} \wedge W_{1,3}$

Aplicam rezolutia din nou unde α este $W_{1,2} \cup W_{1,3}$ si β este $W_{2,2}$(obtinusem $\neg W_{2,2}$ mai sus):

$W_{1,2} \wedge W_{1,3}$

Aplicam rezolutia din nou unde α este $W_{1,3}$ si β este $W_{1,2}$ (obtinusem $\neg W_{1,2}$ mai sus):

$W_{1,3}$

Deci wumpusul se afla in celula [1,3].
Pentru a determina agentul sa actioneze ar trebui introduse noi regulicare leaga starea curenta a lumii de actiuni pe care agentul ar trebui sa le intreprinda. De exemplu daca wumpusul se afla intr-o celula din fata, este o idee proasta ca agentul sa execute actiunea *Inainte*. Pentru cazul in care agentul se afla in celula [1,1] si este orientat catre est, avem urmatoarea regula:

$A_{1,1} \wedge Est_A \wedge W_{2,1} \Rightarrow \neg Inainte$

Logica propozitionala ne permite sa trecem prin toate punctele importante referitoare la ce este logica si cum poate fi folosita pentru a se realiza inferenta care rezulta eventual in urma actiunilor intreprinse. Problema principala este ca sunt prea multe propozitii de tratat. Regula simpla “Nu te duce inainte daca exista un wumpus in fata ta” poate fi reprezentata in logica propozitionala printr-un set de 64 de reguli (16 patrate x 4 orientari pentru agent).

function PROPOSITIONAL-KB-AGENT(*percept*) **returns** an *action*
    static: *KB*, a knowledge base
        t, a counter, initially 0, indicating time

    TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
    for each *action* **in** the list of possible actions **do**
        if ASK(KB, MAKE-ACTION-QUERY(*t, action*)) **then**
            $t - t+1$
            return *action*
end
(Russel, J., Norvik, P. - Artificial Intelligence A modern Approach)

Totusi, o alta problema importanta sunt schimbari survenite in timp. La un moment dat in timp anumite propozitii din baza de cunostinte erau adevarate, iar la alt moment o perte din ele sunt false. Pentru a ocoli posibilele confuzii aparute, vor trebui diferite simboluri propozitionale pentru locatia agentului la fiecare moment de timp.

$A^0_{1,1} \wedge Est^0_A \wedge W^0_{2,1} \Rightarrow \neg Inainte^0$
$A^1_{1,1} \wedge Est^1_A \wedge W^1_{2,1} \Rightarrow \neg Inainte^1$
$A^2_{1,1} \wedge Est^2_A \wedge W^2_{2,1} \Rightarrow \neg Inainte^2$
.....
$A^0_{1,1} \wedge Nord^0_A \wedge W^0_{2,1} \Rightarrow \neg Inainte^0$
$A^1_{1,1} \wedge Nord^1_A \wedge W^1_{2,1} \Rightarrow \neg Inainte^1$
$A^2_{1,1} \wedge Nord^2_A \wedge W^2_{2,1} \Rightarrow \neg Inainte^2$
.....

Agenti reflexivi -Agenti bazati pe un model sau scop

Agent reflexiv = agent care isi clasifica perceptele si actiunile proprii in mod corespunzator.

Agent bazat pe un model = agent care isi construieste reprezentarea interna a lumii si o foloseste pentru a actiona.

Agent bazat pe scop = agent care isi formeaza scopuri si incearca sa le obtina.