Cuvânt înainte

Despre inteligenţa artificială se vorbeşte din ce în ce mai mult. Ca urmare, cei care încep să studieze inteligenţa artificială sunt mai eterogeni şi au metode şi profile de învăţare diferite. Exerciţiile din această carte vin sub forma unor jocuri din două motive. Pe de o parte, limbajul universal al jocului permite mai multor tipuri de studenţi să se aproprie de domeniul vast şi deloc simplu al inteligenţei artificiale. Pe de altă parte, jocul este un mecanism încă eficient în creşterea motivaţiei studenţilor care rezonează tot mai puţin cu învăţarea doar pe baza unui curs. Jocurile se potrivesc profilului studentului contemporan care nu respinge competiţia, care doreşte un câştig imediat chiar dacă este mic sau care este adeseori ghidat de imperativul distracţiei.

Această carte este un instrument ajutător celor care studiază inteligenţa artificială utilizând manualul *Artificial Intelligence - A Modern Approach* (AIMA) a lui Russell şi Norvig [1]. AIMA este văzută de către studenţi ca extrem de densă. Pentru a sparge această nucă tare numită AIMA, am dorit oferirea către studenţi a unui instrument complementar care să-i ajute în procesul de construire a hărţii cognitive proprii pe domeniul inteligenţei artificiale. Pentru crearea de conexiuni între numeroasele metode prezentate în AIMA mă bazez pe două tipuri de jocuri: 1) jocuri de tip Kahoot (www.kahoot.com), respectiv 2) rebusuri în inteligenţă artificială.

Jocurile de tip Kahoot reprezintă un test rapid, în limita a 5 minute, la care studenţii participă la finalul fiecărui curs de inteligenţă artificială. Sunt relevante corectitudinea răspunsului şi timpul de gândire. Ipoteza este că un student atent la discuţiile din timpul cursului ar putea răspunde la întrebări. Exerciţiile propuse în această carte sunt un antrenament pentru testele din timpul cursurilor. Formatul tipărit poate reproduce unele proprietăti ale jocului precum timpul alocat fiecărei întrebări, dar pierde elemente specifice interacţiunii directe: utilizarea dispozitivelor mobile, interacţiunea şi suportul colegilor vecini, efectele grafice şi muzica specifice platformei Kahoot.

Jocuri de tip rebus au fost dezvoltate pentru fiecare capitol din AIMA. Studenţii trebuie să vâneze cuvintele cerute în textul sursă. Se încurajează astfel identificarea unor concepte din AIMA, uneori complementare celor prezentate la curs. Ipoteza este că un student curios ar putea citi mai mult despre metodele din inteligenţa artificială care sunt prezentate doar într-o singură propoziţie în AIMA. Aceste metode ar putea trece neobservate de către studenţii care citeşc din AIMA doar părţile relevante pentru întelegerea prezentărilor din timpul cursului.

Prin această abordare oarecum inedită pentru un manual de inteligenţă artificială îl invit pe cititor în largirea prin joc a universului său de cunoaştere.

Partea I

Exerciţii

Capitolul 1

Întrebări cu răspuns multiplu

1. **Introducere în inteligenţă artificială**

(a) Când s-a născut inteligenţa artificială? (30s)

- ◯ 1956 - întâlnirea de la Dartmouth
- ◯ 1943 - modelarea creierului cu circuite booleene
- ◯ 1950 - articolul lui Turing *Computing Machinery and Intelligence*
- ◯ 1956 - algoritmul complet al lui Robinson pentru raţionare logică

(b) Testul lui Turing tratează IA din perspectiva (30s)

- ◯ gândirii umane
- ◯ gândirii raţionale
- ◯ comportamentului uman
- ◯ comportamentului raţional

(c) Perspectiva AIMA asupra IA impune agenţilor să (30s)

- ◯ gândească uman
- ◯ gândească raţional
- ◯ acţioneze uman
- ◯ acţioneze raţional

(d) Numele primului calculator care a învins campionul mondial la şah? (30s)

- ◯ Deep Purple
- ◯ Windows Azure
- ◯ Deep Blue
- ◯ Black Friday

(e) Cine nu apare pe coperta AIMA? (30s)

- ◯ Alan Turing (1912 1954)
- ◯ Aristotle (384 322 I.C.),
- ◯ Thomas Bayes (1702 1761)
- ◯ Marvin Minski (1927 2016)

(f) Testul Turing total include (30s)

- ◯ învăţare automată şi viziune computerizată
- ◯ viziune computerizată şi robotică
- ◯ procesarea limbajului natural
- ◯ înţelegerea limbajului şi robotică

(g) Predicţii legate de testul Turing (30s)

- ◯ până în 2000, 30% şanse ca cineva să fie indus în eroare pentru 5 minute
- ◯ până în 2000, 30% şanse ca un expert să fie indus în eroare
- ◯ până în 2000, să fie păcălită orice persoană
- ◯ 20% şanse să fie păcălită orice persoană

(h) Sistemele multi-agent s-au dezvoltat începând cu (30s)

- ◯ 1989
- ◯ 2001
- ◯ 2003
- ◯ 1995

(i) Ultimul joc la care agentul uman a avut supremaţia a fost (30s)

- ◯ Şah
- ◯ Go
- ◯ Spinner
- ◯ Game of Thrones

2. **Agenţi**

(a) Raţionalitatea implică (30s)

- ◯ omniscienţă, învăţare, autonomie
- ◯ explorare, învăţare, autonomie
- ◯ culegere de informaţii, învăţare, omniscienţă
- ◯ succes, învăţare, autonomie

(b) PEAS provine de la (30s)

- ◯ Planning, Environment, Actuators, Sensors
- ◯ Performance measure, Environment, Actuators, Search
- ◯ Performance measure, Environment, Agents, Sensors
- ◯ Performance measure, Environment, Actuators, Sensors

(c) Mediul este semi-static dacă (30s)

- ◯ mediul şi funcţia de performanţă se schimbă
- ◯ mediul se schimbă, funcţia de performanţă nu
- ◯ mediul nu se schimbă, funcţia de performantă se schimbă
- ◯ mediul şi funcţia de performanţă nu se schimbă

(d) În medii nondeterminste stărilor succesoare le sunt ataşate probabilităţi (30s)

- ◯ Adevărat
- ◯ Fals

(e) Mediul în jocurile de tip *rebus* este (30s)

- ◯ deterministic şi continuu
- ◯ static şi parţial observabil
- ◯ multi-agent şi secvenţial
- ◯ nicio variantă din cele anterioare

(f) Mediul pentru şah fără ceas este (30s)

- ◯ semi-static şi complet observabil
- ◯ static şi complet observabil
- ◯ continuu şi secvenţial
- ◯ episodic şi multi-agent

(g) Mediul pentru şoferul de taxi este (30s)

- ◯ continuu şi complet observabil
- ◯ dinamic şi complet observabil
- ◯ continuu şi secvenţial
- ◯ episodic şi multi-agent

(h) Un agent reflex bazat pe model menţine o stare interioară (30s)

- ◯ Adevărat
- ◯ Fals

(i) Agenţii bazaţi pe utilitate (30s)

- ◯ utilizează reguli de tip *conditie* → *actiune*
- ◯ aleg acţiunea care îi conduce la cea mai bună utilitate expectată
- ◯ folosesc planificarea pentru a atinge starea ţintă

(j) Un agent care învaţă îşi poate îmbunătăţi (30s)

- ◯ doar cunoştinţele iniţiale
- ◯ doar funcţia de evaluare a performanţei
- ◯ doar generatorul de probleme utilizat în învăţare
- ◯ toate elementele sale

3. **Căutare neinformată**

(a) Dacă agentul nu ştie care este starea lui iniţială, problema de rezolvat este de tipul (30s)

- ◯ stare unică (single state)
- ◯ conformant
- ◯ contingent

(b) Complexitatea în timp a unui algoritm reprezintă (30s)

- ◯ timpul în secunde pentru calcularea soluţiei
- ◯ numărul mediu de noduri în memorie
- ◯ numărul total de noduri generate/expandate
- ◯ numărul maxim de noduri în memorie în timpul căutării

(c) Un algoritm care găseşte întotdeauna soluţia dacă aceasta există este (30s)

- ◯ complet
- ◯ optimal

(d) Care este cea mai bună strategie de căutare pe un calculator cu memorie puţină? (30s)

- ◯ în adâncime
- ◯ în lăţime
- ◯ cost uniform

(e) Căutarea în adâncime găseşte cea mai bună soluţie (30s)

- ◯ adevărat
- ◯ fals

(f) A* este un algoritm de tip greedy ghidat de functia de estimare a costului $f(n) =$ (30s)

- ◯ costul de la nodul n la ţintă
- ◯ costul de la starea iniţială la nodul n
- ◯ maximul dintre costul de la starea iniţială la n şi costul de la n la ţintă
- ◯ nicio variantă din cele enumerate

(g) Care nu este un algoritm de căutare informată? (20s)

- ◯ cost uniform
- ◯ greedy-best search
- ◯ A*

(h) În problema jocului culisant, care nu sunt euristici admisibile? (30s)

- ◯ $h_1 =$ numărul de piese care nu se află în poziţia finală
- ◯ $h_2 =$ suma distanţelor de la fiecare piesă la poziţia ei finală
- ◯ $h_3 = max(h_1, h_2)$
- ◯ $h_4 = sum(h_1, h_2)$

4. **Dincolo de căutarea clasică**

(a) Care dintre următorii algoritmi este complet? (30s)

- ◯ hill climbing
- ◯ stochastic hill climbing
- ◯ first choice hill climbing
- ◯ random-restart hill climbing

(b) Când o stare are mii de stări succesoare, vei utiliza în căutare (60s)

- ◯ hill climbing
- ◯ stochastic hill climbing
- ◯ first choice hill climbing

(c) Algoritmul de *răcire controlată* păstrează în memorie (20s)

- ◯ 1 stare
- ◯ k stări, $k > 1$

(d) Algoritmul de *răcire controlată* porneşte cu (20s)

- ◯ temperatură ridicată
- ◯ temperatură joasă

(e) Algoritmul de *răcire controlată* (30s)

- ◯ nu permite selectarea stărilor cu utilitate mai mică decât cea curentă
- ◯ permite selectarea stărilor mai proaste cu probabilitatea $P = \frac{1}{e^{\frac{\Delta E}{T}}}$
- ◯ permite selectarea stărilor mai proaste cu probabilitatea $P = \frac{1}{e^{T}}$
- ◯ nicio variantă dintre acestea

(f) *Local beam search* cu k stări este similar cu k algoritmi hill climbing rulaţi în paralel. (30s)

- ◯ Adevărat
- ◯ Fals

(g) Care nu este un operator genetic în algoritmii genetici? (30s)

- ◯ ADN
- ◯ mutaţie
- ◯ încrucişare
- ◯ clonare

(h) Într-un nod de tip AND, ramificaţia este introdusă de (20s)

- ◯ alegerile mediului
- ◯ alegerea facută de agent

(i) *Belief state* reprezintă: (30s)

- ◯ setul stărilor posibile în care un agent poate fi
- ◯ spaţiul de căutare al unui agent religios
- ◯ cea mai probabilă stare în care agentul poate fi
- ◯ spaţiul de căutare al unui agent ateist

5. **Căutare adversarială**

(a) Minimax execută în arborele jocului o căutare de tipul (20s)

- ◯ în adâncime
- ◯ în lăţime
- ◯ cost uniform
- ◯ A*

(b) Ordinea de analiză a mutărilor nu afectează eficienţa eliminării nodurilor (pruning) (30s)

- ◯ fals
- ◯ adevărat

(c) Tăierea cu alpha-beta nu afectează rezultatul final (10s)

- ◯ adevărat
- ◯ fals

(d) În practică, pentru evaluarea poziţiei se utilizează o combinaţie de atribute (20s)

- ◯ liniară
- ◯ nonliniară

(e) Căutarea de tip *quiescent* analizează poziţiile interesante la o adâncime mai mare (10s)

- ◯ fals
- ◯ adevărat

(f) Ce se poate utiliza pentru a implementa *tăierea înainte (forward pruning)*? (30s)

- ◯ hill climbing
- ◯ beam search
- ◯ hill climbing stocastic
- ◯ răcire controlată (simulating annealing)

(g) Valoarea expectată a unei poziţii în jocuri stocastice este (30s)

- ◯ media tutoror rezultatelor posibile ale nodurilor de tip *min* şi *max*
- ◯ suma tutoror rezultatelor posibile ale nodurilor de tip *frunză*
- ◯ media tutoror rezultatelor posibile ale nodurilor de tip *şansă*
- ◯ suma tutoror rezultatelor posibile ale nodurilor de tip *şansă*

(h) În jocuri stocastice, comportamenul se prezervă pentru orice transformare a funcţiei de evaluare care este (30s)

- ◯ monotonă
- ◯ nonmonotonă
- ◯ liniar pozitivă
- ◯ liniar negativă

(i) În jocuri stocastice, căutarea la adâncimi mari este esenţială (30s)

- ◯ adevărat
- ◯ fals

6. **Probleme de satisfacere a constrângerilor**

(a) Algoritmul de backtracking poate fi îmbunătăţit prin: (30s)

- ◯ nu poate fi imbunătăţit!
- ◯ sortarea variabilelor în ordine lexicografică
- ◯ detectarea timpurie a eşecului inevitabil
- ◯ exploatând structura problemei

(b) Algoritmul de backtracking poate beneficia de euristici în alegerea (30s)

- ◯ domeniului unei variabile
- ◯ constrângerilor care să fie eliminate
- ◯ variabilei care se instanţiază la pasul următor
- ◯ valorii din domeniu care se asignează variabilei curente

(c) Euristica *valorile minime rămase* selectează (30s)

- ◯ variabilele cu cele mai multe constrângeri pe variabile rămase
- ◯ variabila cu cele mai puţine valori disponibile
- ◯ valoarea care elimină cele mai puţine valori ale variabilor învecinate
- ◯ valorile care introduc cele mai multe constrângeri în variabile învecinate

(d) Euristica grad selectează (30s)

- ◯ variabilele cu cele mai multe constrângeri pe variabile rămase
- ◯ variabila cu cele mai puţine valori disponibile
- ◯ valoarea care elimină cele mai puţine valori ale variabilelor învecinate
- ◯ valorile care introduc cele mai multe constrângeri în variabilele învecinate

(e) Euristica *cele mai puţine constrângeri* selectează (30s)

- ◯ variabilele cu cele mai multe constrângeri pe variabile rămase
- ◯ variabila cu cele mai puţine valori disponibile
- ◯ valoarea care elimină cele mai puţine valori ale variabilelor învecinate
- ◯ valorile cure introduc cele mai multe constrângeri în variabilele învecinate

(f) Pentru a detecta eşecul cât mai repede posibil se utilizează euristica (30s)

- ◯ valorile minime rămase
- ◯ grad
- ◯ cele mai puţine constrângeri

(g) Pentru a reduce factorul de ramificare se utilizează (30s)

- ◯ valorile minime rămase
- ◯ grad
- ◯ cele mai puţine constrângeri

(h) Verificarea înainte (forward checking) evită detectarea târzie a conflictului (20s)

- ◯ adevărat
- ◯ fals

(i) $X, Y, Z \in \{0, 1, 2, 3\}, X < Y < Z$ (30s)

- ◯ $X \in \{0, 1, 2\}$
- ◯ $X \in \{2, 3\}$
- ◯ $X \in \{1, 2\}$
- ◯ $X \in \{0, 1\}$

(j) Bound propagation $D_1 = [0, 5], D_2 = [0, 8], D_1 + D_2 = 10$ (30s)

- ◯ $D_1 = [5, 5], D_2 = [2, 8]$
- ◯ $D_1 = [0, 5], D_2 = [2, 8]$
- ◯ $D_1 = [2, 5], D_2 = [5, 8]$
- ◯ nicio variantă

7. **Agenţi logici**

(a) Lumea lui Wumpus este (30s)

- ◯ complet observabilă şi deterministă
- ◯ episodică şi observabilă local
- ◯ statică şi include un singur agent
- ◯ multi-agent şi secvenţială

(b) Care expresie nu este satisfiabilă? (30s)

- ◯ $\neg a \vee b$
- ◯ $a \rightarrow \neg a$
- ◯ $a \vee \neg a \rightarrow a \wedge \neg a$
- ◯ $a \wedge \neg a \vee b$

(c) $KB \vDash \alpha$ dacă şi numai dacă (20s)

- ◯ $M(KB) \subseteq M(\alpha)$
- ◯ $M(\alpha) \subseteq M(KB)$

(d) Propoziţia $p \rightarrow q \equiv \neg q \rightarrow \neg p$ este (30s)

- ◯ falsă
- ◯ adevarată
- ◯ depinde doar de p
- ◯ depinde doar de q

(e) Contrapoziţia spune că: (30s)

- ◯ $(a \rightarrow b) \equiv \neg b \rightarrow \neg a$
- ◯ $\neg(\neg a) \equiv a$
- ◯ $\neg(a \wedge b) \equiv (\neg a \vee \neg b)$
- ◯ $\neg(a \vee b) \equiv (\neg a \wedge \neg b)$

(f) Tabelul de adevăr al implicaţiei conţine (10s)

- ◯ 4 de zero
- ◯ 3 de zero
- ◯ 2 de zero
- ◯ 1 singur zero

(g) Care din următoarele propoziţii sunt tautologii? (20s)

- ◯ $a \wedge \neg a$
- ◯ $a \vee \neg a$
- ◯ $a \vee \neg a$
- ◯ $\neg a$

(h) Care din următoarele nu este o clauză Horn? (10s)

- ◯ $a \rightarrow b$
- ◯ $a \rightarrow \neg b$

◯ $a \to b \wedge c$

◯ $a \to b \vee c$

(i) Raţionarea înainte este (30s)

◯ ghidată de ţel

◯ ghidată de date

(j) Care expresie este în forma normal conjunctivă? (10s)

◯ $a \to b$

◯ $a \wedge b$

◯ $\neg a \vee b$

◯ $\neg(a \wedge b)$

(k) Pentru a demonstra α, rezoluţia arată că $KB \wedge \neg\alpha$ este (10s)

◯ nesatisfiabilă

◯ satisfiabilă

8. **Logica de ordinul întâi**

(a) Ce este FOL? (10s)

- ◯ First Order Language
- ◯ First order logic
- ◯ Focus on logic
- ◯ First order (Star Wars)

(b) Predicatul $between(x, y)$ este (20s)

- ◯ unar
- ◯ binar
- ◯ ternar

(c) Logica de ordinul include (20s)

- ◯ obiecte, relaţii, funcţii
- ◯ timp, credinţe, cunoştinţe
- ◯ constrângeri
- ◯ aceeaşi expresivitate cu logica propoziţională

(d) *Fiecăruia îi place îngheţata* este formalizată în FOL (20s)

- ◯ $place(x, inghetata)$
- ◯ $place \wedge inghetata$
- ◯ $\exists x, place(x, inghetata)$
- ◯ $\forall x, place(x, inghetata)$

(e) *Există o persoană care iubeşte pe toată lumea* este formalizată în FOL (20s)

- ◯ $\forall x\ \exists y, iubeste(x, y)$
- ◯ $\exists x\ \forall y, iubeste(x, y)$
- ◯ $\forall x\ iubeste(x, Persoana)$
- ◯ $\exists x\ iubeste(x, ToataLumea)$

(f) *Brothers are siblings* este formalizată în FOL (20s)

- ◯ $\exists x\ \exists y Brother(x, y) \rightarrow Sibling(x, y)$
- ◯ $\forall x\ \exists y Brother(x, y) \wedge Sibling(x, y)$
- ◯ $Brother(x) \rightarrow Sibling(y)$
- ◯ $\forall x\ \forall y Brother(x, y) \rightarrow Sibling(x, y)$

(g) *Oricine de la UTCN este isteţ* este formalizată în FOL (20s)

- ◯ $\forall x\ la(x, UTCN) \wedge istet(x)$
- ◯ $istet(x, TUCN)$
- ◯ $\forall x\ la(x, UTCN) \Rightarrow istet(x)$
- ◯ $\exists x\ la(x, UTCN) \Rightarrow istet(x)$

(h) Relaţia *sibling* este simetrică (20s)

- ◯ $Sibling(x)$

◯ $\forall x, y\ Sibling(x, y) \leftrightarrow Sibling(y, x)$
◯ $Sibling(x, y) \vee Sibling(y, x)$
◯ $\exists x, y\ Sibling(x, y) \leftrightarrow Sibling(y, x)$

(i) *O mamă este părintele de sex feminin* (30s)

◯ $\exists x, y\ Mother(x, y) \equiv (Female(x) \wedge Parent(x, y))$
◯ $\forall x, y\ Mother(x, y) \equiv (Female(x) \wedge Parent(x, y))$
◯ $\forall x, y\ Mother(x, y) \equiv Parent(x, y)$
◯ $\forall x, y\ Mother(x, y) \wedge (Female(x) \wedge Parent(x, y))$

(j) *Oricine iubeşte pe cineva* este formalizată în FOL (20s)

◯ $\forall x \exists y, iubeste(x, y)$
◯ $\exists x \forall y, iubeste(x, y)$
◯ $\forall x \exists y, iubeste(y, x)$
◯ $\exists x \forall y, iubeste(y, x)$

9. **Inferenţă în logica de ordinul întâi**

(a) Un termen *ground* este (20s)

- ◯ un termen cu cel puţin o constantă
- ◯ un termen fără variabile
- ◯ un sol unde un arbore binar poate fi plantat
- ◯ niciuna dintre aceste opţiuni

(b) *Lifting* înseamnă (20s)

- ◯ transformarea din logică propoziţională în logica de ordinul întâi
- ◯ eliminarea constantelor Skolem
- ◯ introducerea constantelor Skolem

(c) Datalog = (10s)

- ◯ Prolog + funcţii
- ◯ baze de date extinse cu reguli
- ◯ clauze în logica de ordinul întâi plus funcţii
- ◯ clauze în logica de ordinul întâi fără funcţii

(d) O bază de date deductivă (20s)

- ◯ utilizează inferenţa pentru a răspunde la interogări
- ◯ combină programarea logică cu baze de date relaţionale
- ◯ mai expresivă decât o bază de date, dar mai puţin decât un limbaj logic

(e) Skolemizarea este procesul de (10s)

- ◯ eliminare a cuantificatorilor universali
- ◯ eliminare a cuantificatorilor existenţiali
- ◯ introducere a cuantificatorilor universali
- ◯ introducere a cuantificatorilor existenţiali

(f) Logica de ordinul întâi a fost inventată de: (10s)

- ◯ Aristotel (384-322 BC)
- ◯ Boole (1815-1864)
- ◯ Ludwig Wittgenstein (1889-1951)
- ◯ Gottlob Frege (1848-1925)

(g) În Prolog nu există nicio modalitate de a aserta un fapt negativ (e.g. $\neg father(a, b)$) (10s)

- ◯ Fals
- ◯ Adevărat

(h) În forma normal conjunctivă, fiecare clauză conţinută este o (20s)

- ◯ disjuncţie de literari
- ◯ conjuncţie de literari

(i) Care nu este o metodă de inferenţă logică? (10s)

- ◯ raţionarea înainte
- ◯ raţionarea înapoi
- ◯ skolemizarea
- ◯ rezoluţia

(j) Utilizări practice ale demonstratoarelor de teoreme includ (20s)

- ◯ verificare hardware
- ◯ verificare software

10. **Planificare clasică**

(a) PDDL provine de la (10s)

- ◯ Planning Domain Description Language
- ◯ Planning Description Domain Language
- ◯ Planning Domain Definition Language
- ◯ nicio variantă din cele de mai sus

(b) O condiţie deschisă este o precondiţie (10s)

- ◯ cu un termen care nu are toate variabilele instanţiate
- ◯ cu prezumţia de lume deschisă
- ◯ a unui pas fără legături cauzale

(c) Un *clobber* (20s)

- ◯ validează o precondiţie obţinută datorită unui efect al unei acţiuni
- ◯ distruge o precondiţie obţinută printr-o legătură cauzală
- ◯ introduce o relaţie de ordine parţială între acţiuni

(d) Anomalia Sussman (20s)

- ◯ ilustrează dezavantajele planificării greedy
- ◯ opreşte algoritmul de planificare, chiar dacă un plan există.
- ◯ poate fi rezolvată cu ajutorul *clobberilor*

(e) O acţiune este relevantă dacă (20s)

- ◯ poate fi următorul pas într-un plan
- ◯ cel puţin unul din efecte se unifică cu un element din starea finală
- ◯ are toate precondiţiile satisfăcute

(f) Care sunt euristici admisibile pentru planificare? (20s)

- ◯ h_1= ignorarea precondiţiilor
- ◯ h_2= ignorarea stării iniţiale
- ◯ h_3 = ignorarea efectelor care elimină fapte din starea curentă
- ◯ h_4= ignorarea stării finale

(g) Ce se elimină pentru h= *Manhattan* în: $on(t, s1)$, $tile(t)$, $blank(s2)$, $adj(s1, s2)$? (30s)

- ◯ $blank(s2)$
- ◯ $adj(s1, s2)$
- ◯ $on(t, s1)$
- ◯ toate precondiţiile

(h) Ce trebuie eliminat pentru h = *numărul de piese plasate greşit* în: $on(t, s1)$, $tile(t)$, $blank(s2)$, $adj(s1, s2)$? (30s)

- ◯ $blank(s2), on(t, s1)$
- ◯ $blank(s2), adj(s1, s2)$
- ◯ $on(t, s1), tile(t)$
- ◯ toate precondiţiile

11. **Planificare în lumea reală**

(a) Complexitatea algoritmului căii critice este (20s)

- ◯ $O(n)$
- ◯ $O(1)$
- ◯ $O(Nb)$
- ◯ $O(N^b)$

(b) $ES(B) =$ (30s)

- ◯ $max_{A \prec B} ES(A) + Duration(A)$
- ◯ $min_{B \succ A} LS(B) + Duration(A)$
- ◯ $max_{A \prec B} ES(A) - Duration(A)$
- ◯ $min_{B \succ A} LS(B) - Duration(A)$

(c) $LS(A) =$ (30s)

- ◯ $max_{A \prec B} ES(A) + Duration(A)$
- ◯ $min_{B \succ A} LS(B) + Duration(A)$
- ◯ $max_{A \prec B} ES(A) - Duration(A)$
- ◯ $min_{B \succ A} LS(B) - Duration(A)$

(d) Euristica *minimum slack* (20s)

- ◯ este similară heuristicii *numărul de valori minime* specifică CSP
- ◯ garantează obţinerea soluţiei optime
- ◯ nicio variantă din cele de mai sus

(e) Este mai bine să avem efecte condiţionate decât o acţiune care nu se poate aplica (20s)

- ◯ întotdeauna
- ◯ niciodată
- ◯ în cazul planificării fără senzori
- ◯ în cazul planificării contingente

(f) Schemele de percepte sunt utilizate în (20s)

- ◯ planificarea fără senzori
- ◯ planificarea contingentă
- ◯ atât planificarea fără senzori, cât şi cea contingentă
- ◯ nicio opţiune din cele anterioare

(g) Pentru identificarea succesului accidental (serendipity) agentul monitorizează din mediu: (20s)

- ◯ acţiunile
- ◯ planul
- ◯ ţelul

(h) La pierderea semnalului GPS, navigatorul e mai bine să execute (20s)

- ◯ replanificare

- ◯ repararea planului
- ◯ depinde de lungimea planului şi perioadă de pierdere a semnalului
- ◯ se opreşte şi asteaptă instrucţiuni de la agentul uman

(i) Acţiunile concurente sunt necesare în cazul (20s)

- ◯ monitorizării planului
- ◯ planificării cu contingenţe
- ◯ planificării multi-agent
- ◯ planificării online

(j) În cazul informaţiilor incomplete se utilizează (20s)

- ◯ planificarea condiţională
- ◯ planificarea cu contingeţe
- ◯ planificarea fără senzori
- ◯ monitorizarea execuţiei şi replanificare

(k) În cazul informaţiilor eronate se utilizează (20s)

- ◯ planificarea conditională
- ◯ planificarea contingentă
- ◯ planificarea fără senzori
- ◯ monitorizarea execuţiei şi replanificare

12. **Calculul evenimentelor**

(a) Care nu este o sarcină de raţionare în calculul evenimentelor? (30s)

- ◯ postdicţia
- ◯ rezoluţia
- ◯ abducţia
- ◯ predicţia

(b) Se poate face planificare în calculul evenimentelor prin (30s)

- ◯ predicţie
- ◯ abducţie
- ◯ postdicţie
- ◯ doar limbajul PDDL poate fi utilizat pentru planificare

(c) $\neg HoldsAt(Awake(Nathan), 0)$ este (30s)

- ◯ o observaţie
- ◯ axiomă cu efect pozitiv
- ◯ axiomă cu efect negativ
- ◯ apariţia unui eveniment

(d) $Happens(Awake(Nathan), 1)$ (30s)

- ◯ o observaţie
- ◯ axiomă cu efect pozitiv
- ◯ axiomă cu efect negativ
- ◯ apariţia unui eveniment

(e) $HoldsAt(Ocupa(p_1, s), t) \wedge HoldsAt(Occupies(p_2, s), t) \rightarrow p_1 = p_2$ (30s)

- ◯ *Ocupa* este o relaţie injectivă
- ◯ *Ocupa* este o relaţie surjectivă

(f) $\neg HoldsAt(On(o, o), t)$ (30s)

- ◯ *On* este o relaţie reflexivă
- ◯ *On* este o relaţie ireflexivă

(g) $\neg HoldsAt(Ringing(p, p), t)$ (30s)

- ◯ un telefon nu poate suna un alt telefon
- ◯ un telefon nu se poate suna pe el însuşi
- ◯ telefonul p nu sună la timpul t

(h) $\neg HoldsAt(Broken(d), t) \rightarrow Initiates(TurnOn(a, d), On(d), t)$ (30s)

- ◯ precondiţie de tip fluent
- ◯ precondiţie de tip acţiune

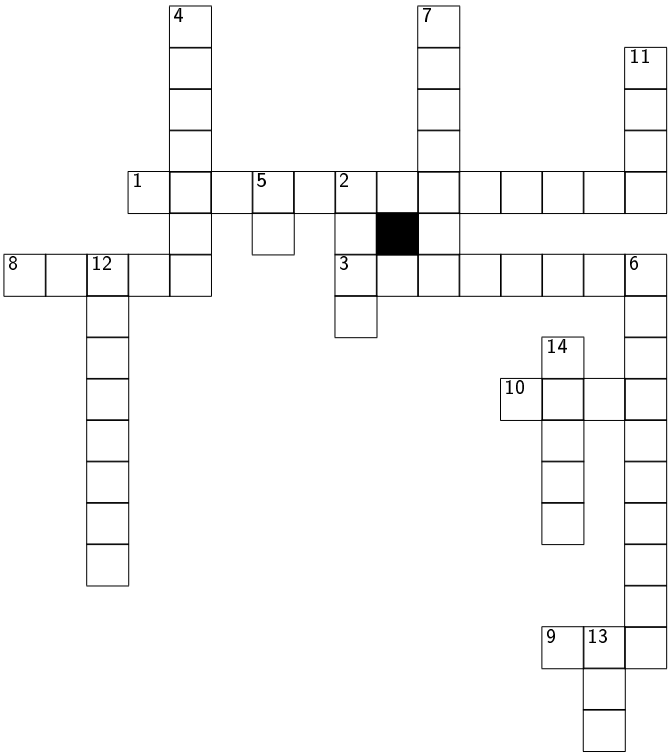
(i) $Happens(WalkThroughDoor(a, d), t) \rightarrow HoldsAt(Near(a, d), t)$ (30s)

- ◯ precondiţie de tip fluent
- ◯ precondiţie de tip acţiune

Capitolul 2

Rebusuri în inteligenţă artificială

Introducere în inteligenţă artificială



Orizontal

1 Când IA depăşeşte inteligenţa umană.

3 Formalizat de Aristotel

8 Vechi sistem expert în medicină

9 Mulţi agenţi

10 Vechi sistem multiagent cognitiv

Vertical

2 Limbaj cu paranteze

4 Veche metodă pentru actualizarea ponderilor între neuroni

5 Algoritmi Genetici! (en)

6 Lumile lui Minsky (en)

7 Vechi sistem expert în chimie

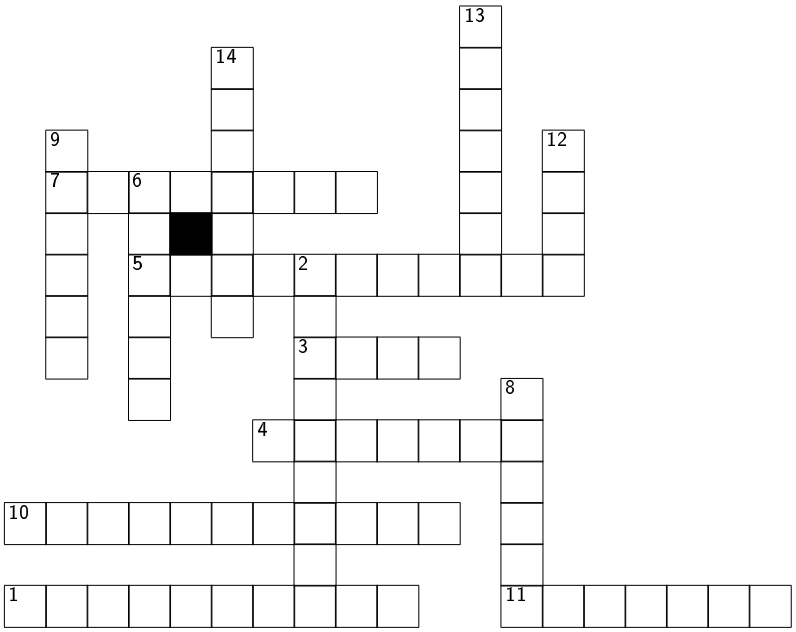
11 Inventatorul primului computer operaţional programabil

12 Turing părea mic în preajma lui

13 Primul calculator electronic

14 Testul lui Turing extins cu viziune computerizată şi robotică

Agenţi inteligenţi



Orizontal

1 Agent care ştie tot ce se va întampla

3 Descrie performanţa, acţiunile, senzorii agentului şi caracterizează mediul (abr)

4 Agent software

5 Opuse jocurilor competitive

7 Mediul în care acţiunile nu depind de stările anterioare

10 Mediul nu se schimbă dar performanţa agentului scade în timpul deliberării

11 Agent conversaţional

Vertical

2 Colectare de informaţie despre mediu

6 Mediu incomplet observabil sau nondeterministic

8 Mediul nu se schimbă în timp ce agentul deliberează

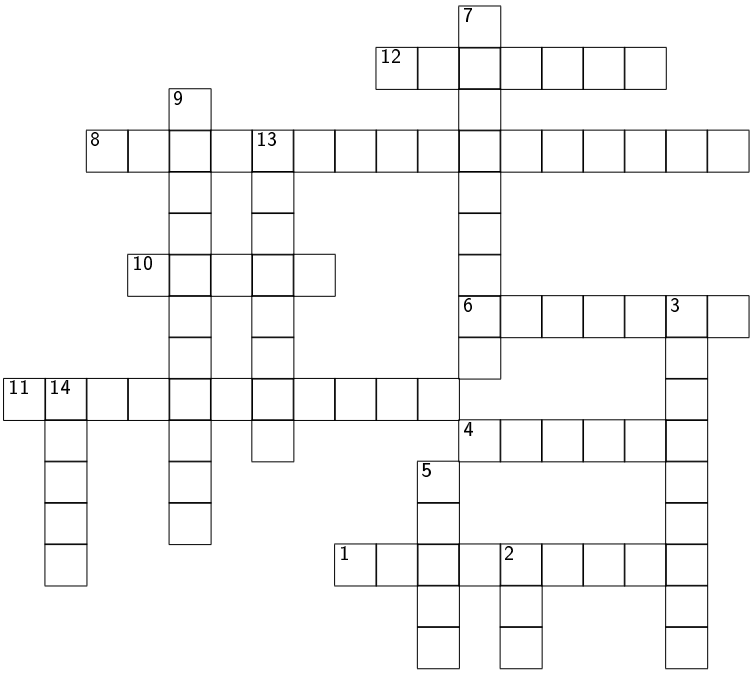
9 Cel mai simplu agent

12 Sistem multi agent în Java

13 Utili în percepţie

14 Agent care se descurcă singur

Rezolvarea problemelor prin căutare



Orizontal

1 Stochează nodurile frunză disponibile pentru expandare

4 Cea mai bună soluţie

6 Algoritm care găseşte soluţia dacă aceasta există

8 Căutare în două sensuri

10 Algoritm informat

11 Euristici monotone

12 Taiere de arbori de căutare (en)

Vertical

2 Problema comis voiajorului (abr, en)

3 Aplicarea acţiunilor legale pe starea curentă

5 Lista cu noduri explorate (en)

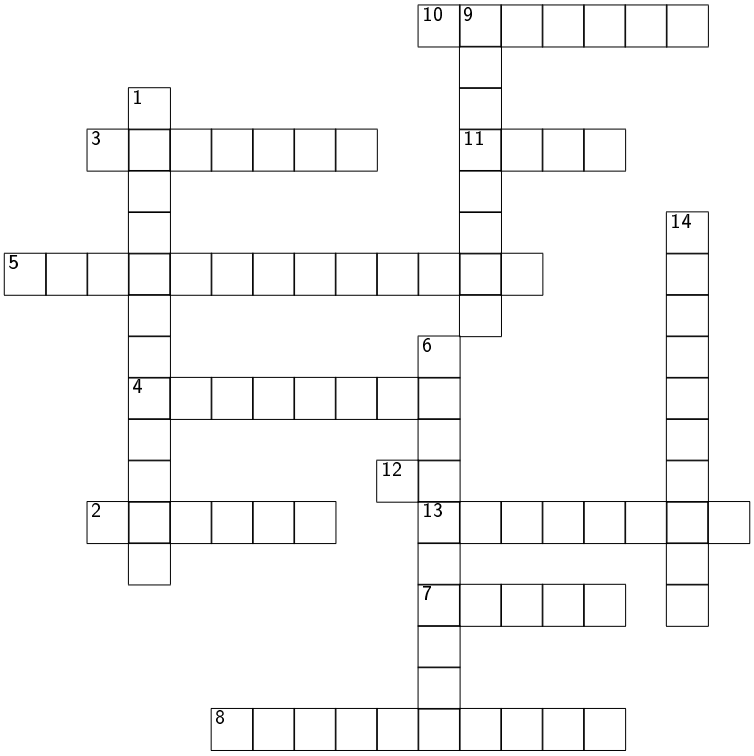
7 Utile în căutarea informată

9 Euristică optimistă

13 Problemă cu mai puţine restricţii

14 Căutare neinformată

Căutare locală



Orizontal

2 Căutare în mediu necunoscut (en)

3 Funcţie care evaluează indivizi (en)

4 Operator genetic

5 Operator genetic

7 Arbore de căutare pentru acţiuni nondeterministe

8 Problemă fără senzori (en)

10 Căutare în care mediul este cunoscut

11 Căutare cu stări interzise

12 Algoritm genetic în care indivizii sunt programe

13 Graf în care toate nodurile au număr egal de muchii de intrare şi ieşire

Vertical

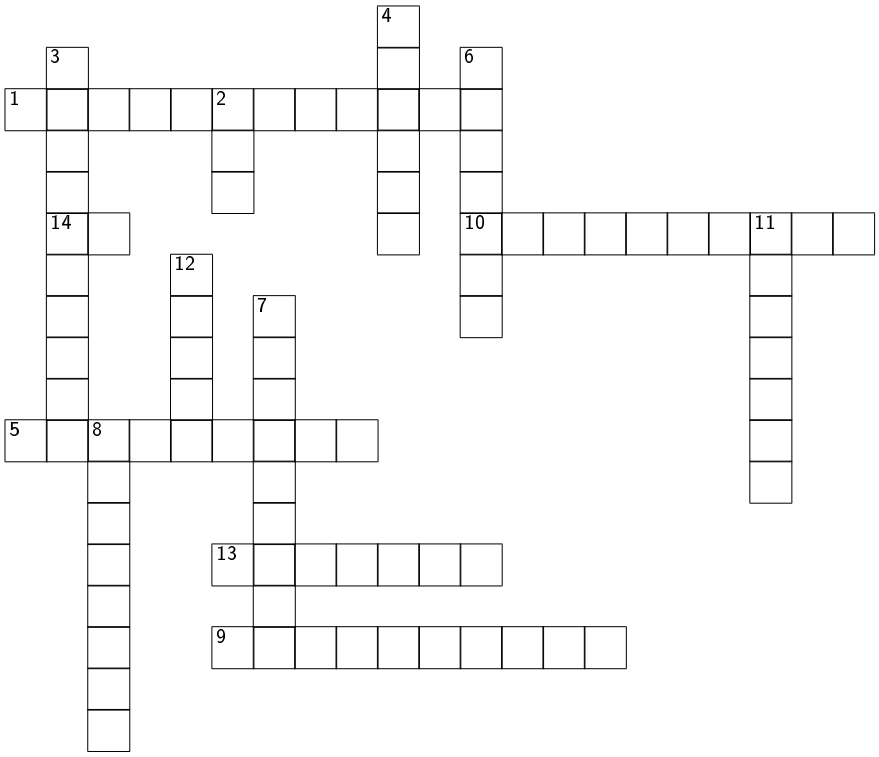
1 Local beam search cu $k = 1$

6 Scade în *simulating annealing*

9 Estimarea stării curente

14 Algoritm genetic cu un singur individ în populaţie

Căutare adversarială



Orizontal

1 Permutaţii ale mutărilor care conduc în aceeaşi stare

5 Tehnică de tăiere a arborilor

9 Jocuri cu elemente aleatoare

10 Simulare pentru evaluarea poziţiei

13 Jocul Othello

14 Joc popular în Asia cu factor mare de ramificaţie

Vertical

2 O jumătate de mutare

3 Şah cu tabla parţial observabilă

4 Test care decide când se aplică funcţia de evaluare a poziţiei (en)

6 Algoritm de căutare adversarială

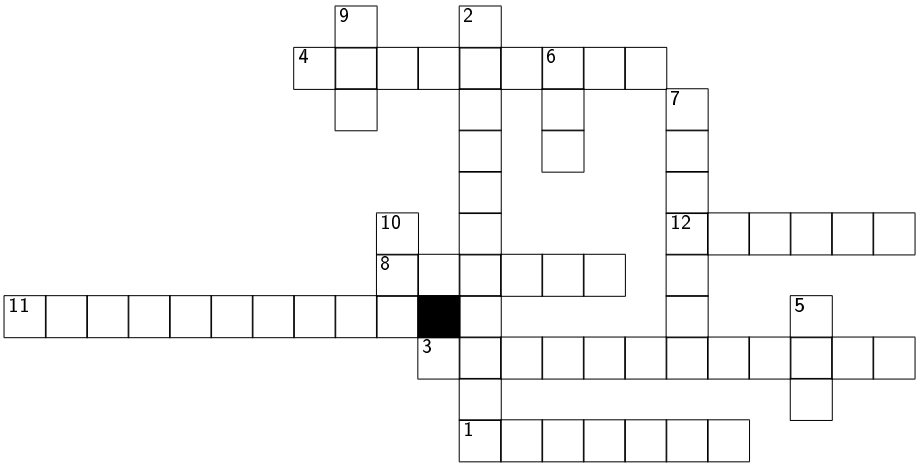
7 Poziţii care nu par să îşi schimbe valoarea în viitorul apropiat

8 Funcţie care primeşte o stare şi întoarce cea mai bună mutare

11 Simulare Monte Carlo pentru jocuri cu zaruri

12 Succesoarea mitologică a lui Deep Blue

Probleme de satisfacere a constrângerilor



Orizontal

1 Constrângere cu multe variabile

3 Euristică la CSP cu căutare locală (en)

4 Inferenţă pe constrângeri

8 Tip de constrângere globală

11 Tip de backtracking în care cea mai recentă decizie este reconsiderată

12 Euristică pe număr de constrângeri (en)

Vertical

2 Schimbarea celei mai recente instanţieri a unei variabile din setul conflict

5 Algoritm pentru consistenţa muchiilor dezvoltat în 1977 de Mackworth

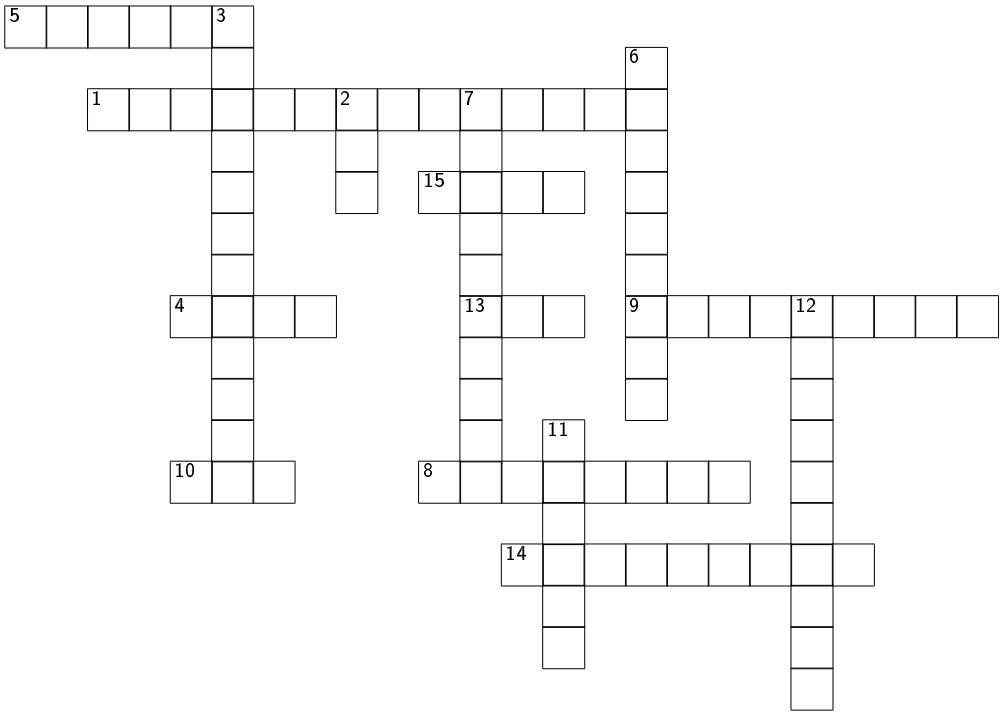
6 Algoritm pentru consistenţa arcului

7 Tip de constrângere globală aplicată şi la Sudoku

9 Euristica celor mai puţine valori rămase

10 Algoritm pentru menţinerea consistenţei arcului

Agenţi logici



Orizontal

1 $a \to b \equiv \neg b \to \neg a$

4 Propoziţie cu un singur literar pozitiv

5 Celebru monstru care duhneşte în labirint

8 $\neg(a \wedge b) \equiv \neg a \vee negb$

9 Deducerea de propoziţii noi pe baza celor vechi

10 Problemă de satisfiabilitate

13 Forma normal conjunctivă (en)

14 Clauză Horn cu exact un literar pozitiv

15 Algoritm pentru satisfiabilitate

Vertical

2 Simbol care nu apare negat sau apare doar negat

3 Propoziţie adevărată în cel puţin o lume posibilă

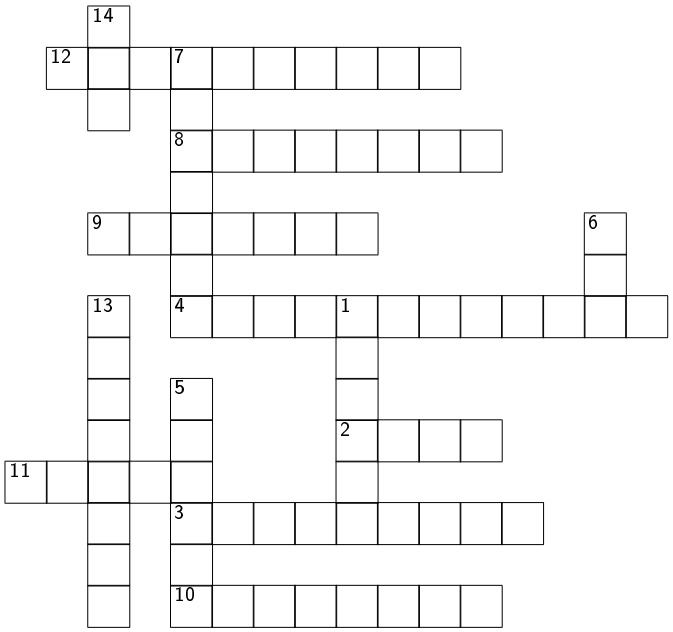
6 Înţelesul propoziţiilor dintr o logică

7 Operator logic pentru $\neg a \vee b$

11 Regulă de înferenţă (lt)

12 Algoritm complet de inferenţă care foloseşte reducerea la absurd

Logica de ordinul întâi



Orizontal

2 Propoziţie atomică pe scurt

3 Cuantificator în logica de ordinul întâi

4 Cuantificator în logica de ordinul întâi

8 Formalizează un domeniu particular

9 Introduce cuantificatorii în 1879

10 Două axiome îi poartă numele

11 Dezvoltă sintaxa logicii de ordinul întâi

12 Propoziţie validă

Verical

1 Funcţii care au valori pentru fiecare intrare din domeniul specificat

5 Termen fără variabile (en)

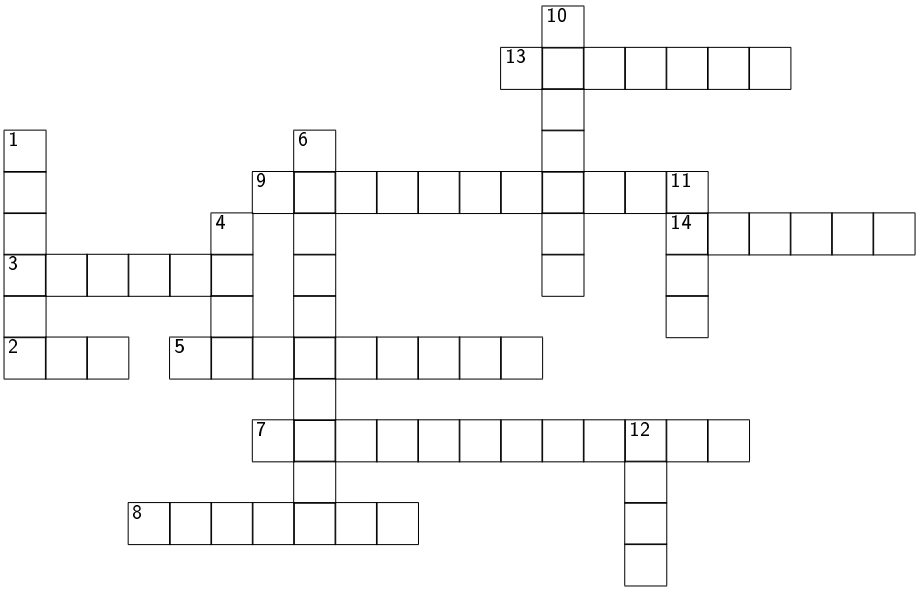
6 Asumpţie în care propoziţiile despre care nu ştim că sunt adevărate sunt false (en)

7 Propozitţii care nu sunt axiome

13 Regulă care leagă o cauză de efectul ei

14 Primul sistem de raţionare în logica de ordinul întâi

Inferenţă în logica de ordinul întâi



Orizontal

2 Cel mai general unificator de clauze (en)

3 Structură cu relaţii de ordine parţială

5 Baze de date care permit inferenţă

7 Regulă de inferenţă în logica de ordiul întâi

8 Demonstrator de teoreme succesor al lui Otter

9 Modus ponens pentru logica de ordinul întâi

13 Baze de cunoştinţe fără funcţii

14 Limbaj de programare logică

Vertical

1 Constante introduse pentru variabilele cuantificate existenţial

4 Algoritm eficient de raţionare cu reguli

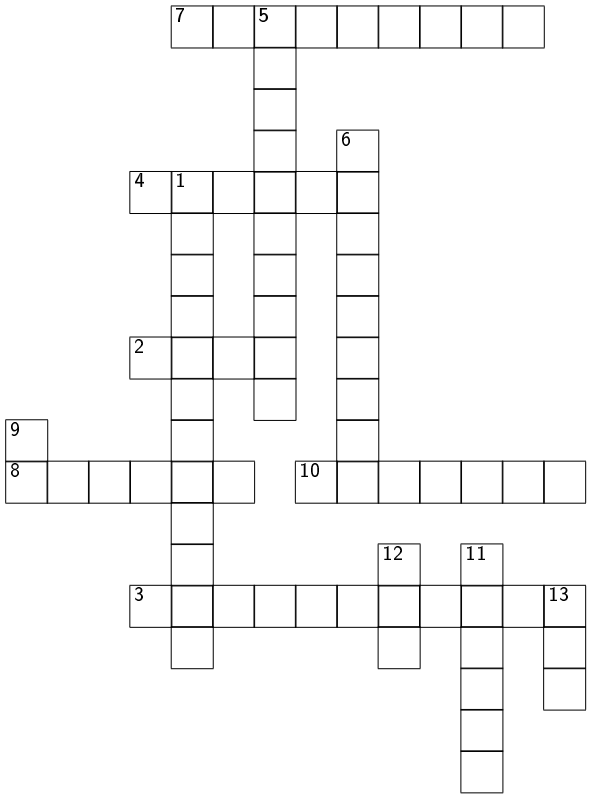
6 Regulă de inferenţă în logica de ordiul întâi

10 Campion între demonstratoarele de teoreme

11 Colecţie de teoreme (en)

12 Demonstrator de teoreme care poate proiecta circuite

Planificare clasică



Orizontal

2 Limbaj specializat pentru probleme de planificare

3 Prezente la acţiuni

4 Prezente la acţiuni

7 Algoritm de planificare

8 Relaţie care variază de la o stare la alta

10 Planificator care foloseşte satisfiabilitate

Vertical

1 Planificator cu euristici în Python

5 Acţiune cu toate precondiţiile satisfăcute

6 Acţiune cu un efect care se regăseşte în starea finală

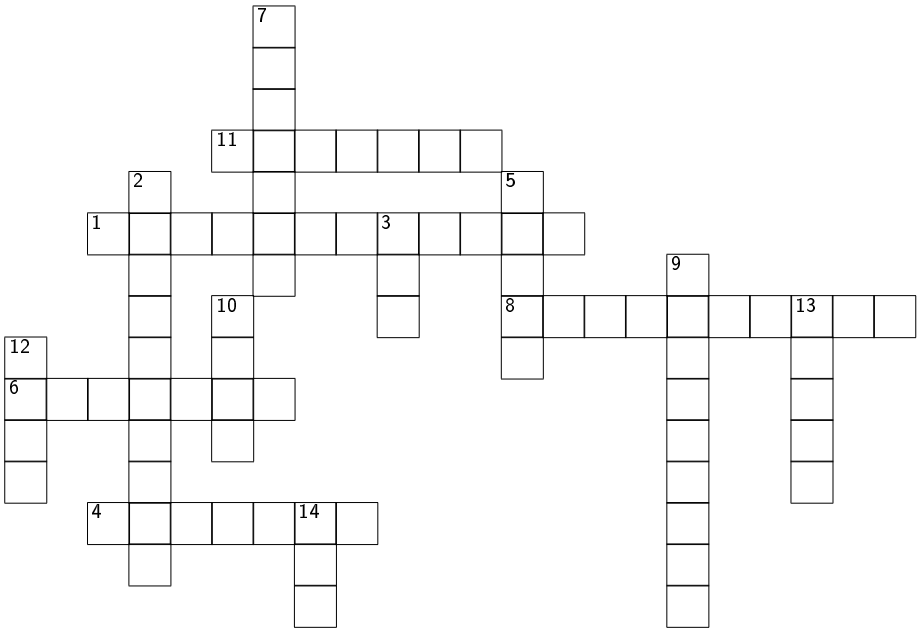
9 Planificator pe repede înainte

11 Limbaj de planificare inclus in PDDL

12 Diagrame de decizie pentru reprezentarea eficientă a planurilor

13 Competiţie de planificare

Planificare în lumea reală



Orizontal

1 Efecte prefixate cu *when*

4 Cea mai lungă cale

6 Nondeterminism generat de un alt agent

8 Planificare fără senzori (en)

11 Consumate de acţiuni

Vertical

2 Necesară în planificarea multi agent

3 Asumpţie în care starea conţine atât fluenţi negativi cât şi pozitivi

5 $LS - ES$ (en)

7 Nondeterminism generat de agentul curent

9 Mod de a selecta planul curent comun pentru mai mulţi agenţi

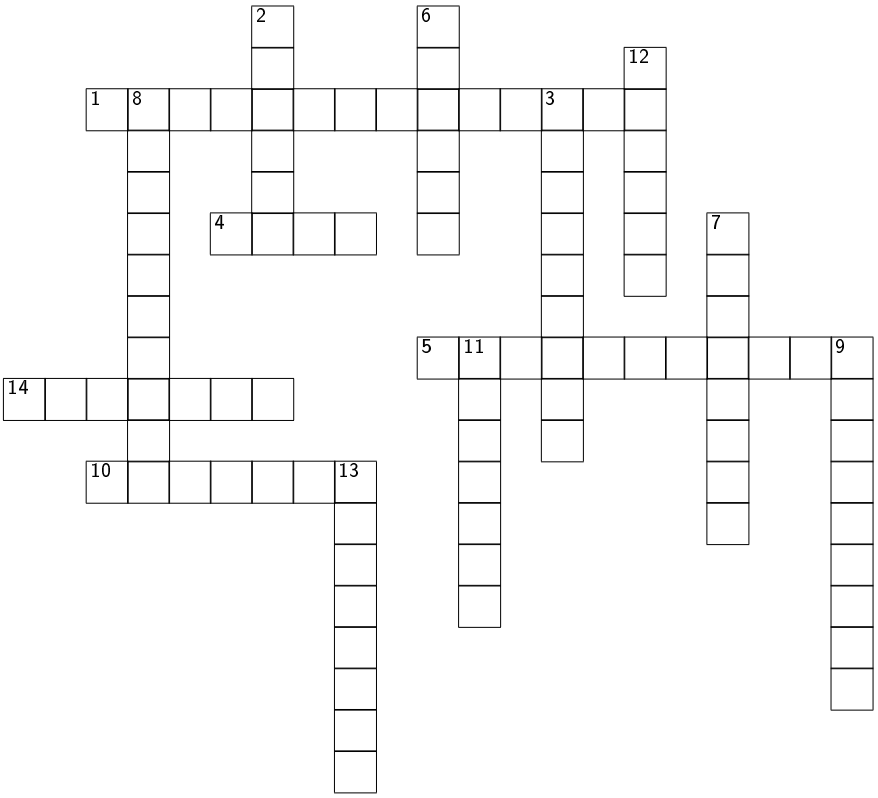
10 Agent pasăre (en)

12 Folosite de către planificatorul conformant HSCP

13 Conferinţă anuală pentru agenţi

14 Determină posibilele momente de început şi sfârşit ale unei acţiuni

Reprezentarea cunoştinţelor. Calculul evenimentelor



Orizontal

1 Mai puternică decât prezumţia de lume închisă

4 Sistem de menţinere a adevărului bazat pe justificări

5 Logici în care o concluzie derivată poate fi retrasă

10 Evenimente care împarţite dau acelaşi eveniment

14 Ontologia wikipediei

Vertical

2 Tipul lui f în $T(f,t)$

3 Ontologie doar cu relaţii de tip *subclasă*

6 Predicat pe intervale temporale (en)

7 Decompoziţie exhaustivă disjunctă

8 Proprietăţi interne ale obiectului

9 Tipul lui e în $Happens(e,i)$

11 Predicat pe intervale temporale (en)

12 Predicat pe intervale temporale (en)

13 Setul maximal de consecinţe dintr o bază de cunoştinţe