Međuispit iz Uvoda u umjetnu inteligenciju (2021./2022.)

– NEKORIGIRANA VERZIJA –

*Ispit se sastoji od **20 pitanja** i ukupno nosi **20 bodova**. Točan odgovor nosi 1 bod, a netočan -1/3 boda. Trajanje ispita je **150 minuta**. Primjerak ispita trebate predati zajedno sa svojim rješenjima.*

1. Pretraživanje prostora stanja i heurističko pretraživanje (7 pitanja)

1 (P) Oblikujemo računalnu igru u kojoj se računalno upravljani igrači kreću po 2D mapi i skupljaju raznorazne artefakte. U jednom trenutku igrači trebaju u što manje poteza doći do zajedničkog zbornog mjesta. Igrači se mogu kretati u četiri smjera (gore, dolje, lijevo, desno), pri čemu rade po jedan korak. Mapa sadržava prepreke (zidove), kroz koje igrači ne mogu proći. Mapa također sadržava slučajan broj slučajno raspoređenih teleportera, koji igrača mogu trenutačno (u samo jednom koraku) transportirati na slučajno mjesto na mapi. No, jednom upotrijebljeni teleporter više se ne može upotrijebiti. Za ovu igru želimo oblikovati optimističnu heuristiku koja će navoditi računalno upravljane igrače do zbornog mjesta. Za oblikovanje heuristike koristimo metodu relaksacije. **Koja od sljedećih relaksacija igre daje optimističnu heuristiku?**

A Teleporteri se mogu koristiti neograničeno mnogo puta

B Igrači mogu prolaziti kroz zidove i dvaput koristiti isti teleporter

C Teleporteri uvijek transportiraju igrača do teleportera koji je najbliži zbornom mjestu

D Igrači se mogu kretati dijagonalno, ali ne mogu koristiti teleportere

2 (P) Stanje s slagalice 3×3 neka je $[[1, 5, 2], [4, \square, 3], [7, 8, 6]]$, dok je ciljno stanje $[[1, 2, 3], [4, 5, 6], [7, 8, \square]]$. Neka je h optimistična heuristika. **Među ponuđenim vrijednostima, koju vrijednost $h(s)$ može poprimiti, a da je najviše obaviještena?**

A 3 B 8 C 7 D 6

3 (P) Razmatramo problem slagalice dimenzija 3×4 s prosječnom dubinom rješenja $d = 12$. Za nalaženje rješenja algoritam pretraživanja u dubinu u prosjeku koristi 720 B memorije. **Koliko bi memorije u prosjeku koristio algoritam pretraživanja u širinu na ovome problemu?**

A 240 MB B 320 MB C 5.4 MB D 404 KB

4 (R) Algoritmom slijepog pretraživanja rješavamo problem Hanojskih tornjeva s tri klinova i tri diska. Klinovi su označeni sa A, B i C, a diskovi sa 1, 2 i 3, pri čemu broj odgovara veličini diska (disk 1 je najmanji, a disk 3 najveći). Stanje možemo označiti kao AxByCz, gdje x, y i z označavaju diskove koji se nalaze na klinovima A, B, odnosno C, krenuvši od dna prema vrhu (ako je klin prazan, ispuštamo njegovo slovo). Npr., u stanju B21C3 na klinu B su diskovi 2 (ispod) i 1 (gore), a na klinu C je disk 3. Početno stanje je A321. Zadatak je prebaciti diskove s klina A na klin C, tj. pronaći stanje C321, koristeći klin B kao pomoćni klin. Pritom se u svakom koraku prebacuje samo jedan disk, uz uvjet da veći disk ne smije biti položen na manji disk. Koristimo algoritam iterativnog pretraživanja u dubinu (IDS) bez liste posjećenih stanja. Algoritam čvorove proširuje leksikografskim poretkom gdje znamenke dolaze prije slova, npr., stanje A32B1 prethodi stanju A3B2C1. **Koliko će čvorova generirati algoritam IDS prije nego što generira čvor sa stanjem A3C21**?

A 16 B 25 C 29 D 22

5 (T) Svi algoritmi slijepog pretraživanja u dubinu eksponencijalne su vremenske složenosti, međutim konkretni izrazi za asimptotsku složenost ipak se malo razlikuju. Asimptotska vremenska složenost algoritma pretraživanja u širinu (BFS) je $\mathcal{O}(b^{d+1})$, a algoritma iterativnog pretraživanja u dubinu (IDS) $\mathcal{O}(b^d)$. **Zašto asimptotska vremenska složenost algoritma IDS nije ista kao i ona algoritma BFS, $\mathcal{O}(b^{d+1})$?**

A IDS generirane čvorove umeće na početak, a BFS na kraj liste, pa BFS na razini d generira b dodatnih čvorova

B IDS čvorove na razini d proširuje b^d puta, dok BFS dodatno proširuje svaki čvor na razini d osim ciljnog

C IDS više puta proširuje iste čvorove, ali na posljednoj razini proširuje sve čvorove osim ciljnog

D IDS ima dubinsko ograničenje koje povećava do dubine rješenja d, pa nikada ne proširuje čvorove razine d

6 (T) Prednost algoritma A^* nad algoritmom pretraživanja s jednolikom cijenom (UCS) jest ta da algoritam A^* koristi heuristiku, dok ju algoritam UCS ne koristi. **Kako se ta prednost manifestira u praksi?**

A Ako je heuristika optimistična, A^* će pronaći put koji je manje cijene od onoga koji pronalazi UCS

B Ako je heuristika optimistična, A^* će generirati manji ili jednak broj čvorova od algoritma UCS

C Neovisno o svojstvima heuristike, A^* će ispitati jednak ili manji broj čvorova od onoga koji će ispitati UCS

D Ako rješenje postoji, A^* će u manje koraka pronaći rješenje manje ili jednake cijene od onoga koje nalazi UCS

7 (R) Prostor stanja pretražujemo algoritmom A^*. Skup stanja je $S = \{a, b, c, d, e, f\}$, a funkcija sljedbenika je $succ(a) = \{(b, 2), (c, 2)\}$, $succ(b) = \{(c, 5), (d, 2)\}$, $succ(c) = \{(d, 1), (f, 20)\}$, $succ(d) = \{(e, 2)\}$, $succ(e) = \{(f, 14)\}$ te $succ(f) = \emptyset$. Heurističke vrijednosti stanja su $h(a) = 16$, $h(b) = 6$, $h(c) = 14$, $h(d) = 4$, $h(e) = 2$, $h(f) = 0$. Početno stanje je a, a ciljno f. Izvedite korake algoritma A^*, bilježeći u svakom koraku sadržaj liste otvorenih čvorova O i skupa zatvorenih čvorova C. U nultom koraku algoritma vrijedi $O = [(a, 0)]$ i $C = \emptyset$. **Koji je sadržaj liste O i skupa C nakon petog koraka izvođenja algoritma?**

A $O = [(d, 3), (f, 22)]$, $C = \{(a, 0), (b, 2), (c, 7), (e, 6)\}$

B $O = [(d, 3)]$, $C = \{(a, 0), (b, 2), (c, 2), (e, 6)\}$

C Algoritam ne dostiže peti korak

D $O = [(d, 3), (f, 20)]$, $C = \{(a, 0), (b, 2), (c, 2), (e, 6)\}$

2. Igranje igara (3 pitanja)

8 (R) Razmatramo igru dvaju minimax-igrača. Stablo igre definirano je prijelazima $succ(A) = \{B, C, D\}$, $succ(B) = \{E, F\}$, $succ(C) = \{G, H\}$, $succ(D) = \{I, J\}$, $succ(G) = succ(F) = \{K, L\}$, $succ(H) = succ(E) = succ(I) = \{M, N\}$ te $succ(J) = \{O, P\}$. Igrač MAX koristi heuristiku h_1, a igrač MIN heuristiku h_2. Vrijednosti heuristika su sljedeće (obje heuristike definiraju dobit dotičnog igrača):

	E	F	G	H	I	J	K	L	M	N	O	P
h_1	1	−2	3	2	0	5	−1	0	−2	−1	0	2
h_2	0	5	2	4	−1	2	3	2	1	−1	1	−2

Oba igrača pretražuju najviše dva poteza unaprijed (jedan svoj i jedan suparnički). Igru započinje igrač MAX u stanju A. **Kroz koja stanja će se razvijati igra?**

A $A, C, G, \ldots$ B $A, B, F, \ldots$ C $A, C, H, \ldots$ D $A, D, I, \ldots$

9 (T) Igrač MAX primjenjuje algoritam minimax s heuristikom h na korijen stabla igre kako bi odredio svoj optimalan potez. Nakon što povuče taj potez, na redu je protivnik MIN, koji također vuče svoj optimalan potez. Igrač MAX zatim nanovo primjenjuje algoritam MAX na novom stanju igre. Pritom igrač MAX ponovo izračunava minimax-vrijednosti i za one čvorove za koje je ranije (za prethodni potez) već bio izračunao minimax-vrijednosti. **Zbog čega je potrebno ponovno izračunavati minimax-vrijednosti za te čvorove?**

A Algoritam minimax radi pretraživanje u dubinu, pa ne pohranjuje čvorove na stazama izvan poteza koji je predvidio igrač MAX

B Protivnik MIN može koristi heuristiku manje točnosti od heuristike h, pa njegovi potezi mogu biti drugačiji od onih koje je predvidio igrač MAX

C Ako je za prethodni izračun korištena heuristika, novi izračun bit će informiraniji jer će ići dvije razine dublje od prethodnog

D Heuristika h nije savršena, pa igrač MAX ne može očekivati da će ostvariti točno onu isplatu koju predviđa heuristika h

10 (P) Razmatramo dva igraća algoritma: jedan s heuristikom h_1 i dubinom pretraživanja d_1 te drugi s heuristikom h_2 i dubinom pretraživanja d_2. Oba algoritma koriste strategiju minimax. **Kada će prvi algoritam češće pobjeđivati drugi algoritam?**

A Ako $h_1 = h_2$ i $d_1 > d_2$ B Ako $d_1 = d_2$ i $h_1 > h_2$ C Ako $h_1 > h_2$ D Ako $d_1 > d_2$

3. Prikazivanje znanja i automatsko zaključivanje (6 pitanja)

11 (T) Unifikacija je postupak svođenja izraza K_1 i K_2 iz FOL na identičan oblik primjenom odgovarajuće supstitucije varijabli, koju nazivamo "unifikator". Postupak unifikacije ključan je za zaključivanje u FOL. Neka α, β i θ označavaju supstitucije, a "$\circ$" kompoziciju dviju supstitucija. **Što mora vrijediti, a da je supstitucija θ najopćenitiji zajednički unifikator (MGU)?**

A $\exists\alpha\big((K_1\alpha = K_2\alpha) \to \forall\beta(\theta = \alpha \circ \beta)\big)$
C $\forall\alpha\forall\beta\big((K_1\theta = K_2\theta) \to (\theta = \alpha \circ \beta)\big)$
B $\forall\beta\exists\alpha\big((K_1\alpha = K_2\alpha) \to (\beta = \theta \circ \alpha)\big)$
D $\forall\alpha\big((K_1\alpha = K_2\alpha) \to \exists\beta(\alpha = \theta \circ \beta)\big)$

12 (T) Logička posljedica središnji je pojam u logici i razlog zašto uopće koristimo sustav formalne logike za prikazivanje znanja. U propozicijskoj logici, logičku posljedicu moguće je dokazati na nekoliko načina. Jedan način temelji se na teoremu semantičke dedukcije. **Kako riječima glasi teorem semantičke dedukcije?**

A Ciljna formula logička je posljedica premisa ako i samo ako je implikacija s premisa na ciljnu formulu tautologija
B Premise i negacija cilja zajedno su zadovoljivi ako i samo ako je ciljna formula logička posljedica premisa
C Ako su premise nezadovoljive, bilo koja formula logička je posljedica takvih premisa
D Ciljna formula je logička posljedica premisa ako i samo ako je ciljna formula istinita kada su istinite premise

13 (R) Dani su FOL atomi $P(f(x), y, g(w, b))$ i $P(y, f(g(a, z)), x)$, gdje je P predikat, a i b su konstante, a f i g su funkcije. **Što je najopćenitiji zajednički unifikator ovih dvaju atoma?**

A Atome nije moguće unificirati
C $P(f(g(a, b)), f(g(a, b)), g(a, b))$
B $P(f(g(a, b)), f(g(a, z)), g(w, b))$
D $P(f(f(g(a, b))), f(g(a, b)), f(g(a, b)))$

14 (R) Zadane su premise: *"Ivo voli sve vrste hrane. Jabuka i piletina su hrana. Branko jede lješnjake i nije mu zlo od njih. Vesna jede sve što i Branko. Ako netko nešto jede i nije mu od toga zlo, onda je to hrana."* Formalizirajte ove premise u FOL i pretvorite ih u klauzule. Pritom koristite $V(x, y)$ za "x voli y", $J(x, y)$ za "x jede y", "$Z(x, y)$" za "x-u je zlo od y", i "$H(x)$" "x je hrana". Zatim rezolucijom opovrgavanjem uz strategiju skupa potpore dokažite cilj *Ivo voli lješnjake*. **Koliko je minimalno koraka (primjena rezolucijskog pravila) potrebno za dokaz?**

A 3 B 6 C 4 D 5

15 (P) Na postupak dokazivanja rezolucijom opovrgavanjem možemo gledati kao na problem pretraživanje prostora stanja, gdje stanja odgovaraju skupu klauzula (onih zadanih i onih izvedenih), a prijelazi između stanja odgovaraju primjeni rezolucijskog pravila na jedan par klauzula. Neka skup premisa sadrži 8 klauzula, a negirani cilj 2 klauzule. Za dokazivanje koristimo dvije strategije: strategiju zasićenja po razinama (LSS) i strategiju skupa potpore (SOS). Jednom razriješeni par klauzula više se ne razrješava. Izračunajte gornju ogradu na ukupan broj primjena rezolucijskog pravila za LSS i za SOS, ako znamo da je cilj dokaziv u tri koraka (tri primjene rezolucijskog pravila). **Koliko je puta gornja ograda za LSS veća od gornje ograde za SOS?**

A 5.71× B 7.51× C 9.65× D 4.53×

16 (P) Ispravnost je važno svojstvo pravila zaključivanja. **Koje je od sljedećih pravila zaključivanja ispravno?**

A $\neg A, A \to B \vdash \neg B$
C $A \to (B \to C) \vdash (A \vee B) \to C$
B $A \vdash A \wedge B$
D $A \to (B \vee C), \neg C \vdash A \to B$

4. Logičko programiranje i ekspertni sustavi (4 pitanja)

17 (R) Razmatramo sljedeći program u Prologu:

```
roditelj(kronos, zeus).
roditelj(zeus, minos).
roditelj(europa, minos).
potomak(X, Y) :- roditelj(Y, X).
potomak(X, Y) :- roditelj(Y, Z), potomak(X, Z).
bog(kronos).
bog(X) :- potomak(X, Y), bog(Y).
```

```
polubog(X) :- potomak(X, Y), not(bog(Y)).
```

Nad ovako definiranom bazom izvodimo upit `polubog(minos)`. Nacrtajte Prologovo stablo dokaza za ovaj upit do trenutka drugog povratka u pretraživanju. Pritom za dokazivanje negiranog cilja koristite dva čvora (jedan za `not(P(X))` i drugi za `P(X)`)), dok prazni stog ne brojite kao čvor. **Koliko čvorova ima stablo dokaza neposredno prije drugog povratka u pretraživanju?**

A 12 B 8 C 6 D 10

18 (T) Prolog je deklarativan programski jezik, ali sadrži neke nedeklarativne aspekte. Jedan aspekt nedeklarativnosti u Prologu jest taj da u nekim slučajevima ne vrijedi komutativnost operatora "∧", npr., program `P(X,Y) :- Q(X), R(Y)` i program `P(X,Y) :- R(Y), Q(X)` ne evaluiraju se nužno identično. **Zbog čega u Prologu u ovom slučaju može postojati odstupanje proceduralnog značenja od deklarativnog?**

A Prolog pretpostavlja zatvorenost svijeta, no u stvarnosti činjenice koje nisu u bazi znanja mogu biti istinite

B Prolog koristi rezoluciju SLD i pretraživanje u dubinu, pa dokazivanje ovisi o redoslijedu atoma na stogu

C Prolog koristi definitne klauzule, pa do odstupanja dolazi kada neku FOL formulu ne možemo izraziti u Prologu

D Prolog ne provodi standardizaciju klauzula, pa različiti redoslijed klauzula može dati različite unifikatore

19 (T) Zaključivanje u ekspertnim sustavima može biti izvedeno kao ulančavanje unaprijed ili ulančavanje unazad. **U kojim je primjenama zaključivanje unazad bolji izbor od ulančavanja unaprijed?**

A Kada novoizvedeni zaključci mogu invalidirati ranije izvedene zaključke

B Kada ako–onda pravila imaju različite prioritete koji se mogu mijenjati tijekom izvođenja programa

C Kada je broj mogućih hipoteza malen, a broj potpora (dokaza) za te hipoteze potencijalno velik

D Kada je znanje moguće dekomponirati u granule znanja i formalizirati ih ako–onda pravilima

20 (R) Baza znanja ekspertnog sustava sadrži sljedeća pravila:

(1) AKO $(A = a_2) \wedge (D = d_2)$ ONDA $C = c_1$
(2) AKO $(F = f_3) \vee (B = b_3)$ ONDA $C = c_2$
(3) AKO $(E = e_1) \vee (B = b_1)$ ONDA $(A = a_1) \wedge (D = d_2)$
(4) AKO $F = f_1$ ONDA $D = d_2$
(5) AKO $F = f_2$ ONDA $E = e_2$
(6) AKO $(B = b_3) \vee (D = d_1)$ ONDA $(E = e_1) \wedge (A = a_2)$

Sustav koristimo za izvođenje vrijednosti varijable C ulančavanjem unazad. Prednost imaju pravila s manjim rednim brojem. Pravila koja su jednom palila više ne mogu paliti. Jednom izvedeni međuciljevi brišu se sa stoga, neovisno o njihovoj poziciji na stogu. Na možebitne upite od strane sustava, korisnik odgovara sa $B = b_3$ i $F = f_1$. **Što radi ekspertni sustav pri izvođenju vrijednosti varijable C?**

A Pali četiri pravila i izvodi $C = c_1$
B Odbacuje pravilo 2 te kasnije pali pravilo 5
C Pali tri pravila i izvodi $C = c_2$
D Pali pet pravila i izvodi $C = c_1$

Međuispit iz Uvoda u umjetnu inteligenciju (2021./2022.)

– NEKORIGIRANA VERZIJA –

*Ispit se sastoji od **20 pitanja** i ukupno nosi **20 bodova**. Točan odgovor nosi 1 bod, a netočan -1/3 boda. Trajanje ispita je **150 minuta**. Primjerak ispita trebate predati zajedno sa svojim rješenjima.*

1. Pretraživanje prostora stanja i heurističko pretraživanje (7 pitanja)

1 (P) Razmatramo problem slagalice dimenzija 4×4 s prosječnom dubinom rješenja $d = 12$. Za nalaženje rješenja algoritam pretraživanja u dubinu u prosjeku koristi 720 B memorije. **Koliko bi memorije u prosjeku koristio algoritam pretraživanja u širinu na ovome problemu?**

A 25.2 MB B 320 MB C 10.1 MB D 240 MB

2 (R) Algoritmom slijepog pretraživanja rješavamo problem Hanojskih tornjeva s tri klinova i tri diska. Klinovi su označeni sa A, B i C, a diskovi sa 1, 2 i 3, pri čemu broj odgovara veličini diska (disk 1 je najmanji, a disk 3 najveći). Stanje možemo označiti kao AxByCz, gdje x, y i z označavaju diskove koji se nalaze na klinovima A, B, odnosno C, krenuvši od dna prema vrhu (ako je klin prazan, ispuštamo njegovo slovo). Npr., u stanju B21C3 na klinu B su diskovi 2 (ispod) i 1 (gore), a na klinu C je disk 3. Početno stanje je A321. Zadatak je prebaciti diskove s klina A na klin C, tj. pronaći stanje C321, koristeći klin B kao pomoćni klin. Pritom se u svakom koraku prebacuje samo jedan disk, uz uvjet da veći disk ne smije biti položen na manji disk. Koristimo algoritam iterativnog pretraživanja u dubinu (IDS) bez liste posjećenih stanja. Algoritam čvorove proširuje leksikografskim poretkom gdje znamenke dolaze prije slova, npr., stanje A32B1 prethodi stanju A3B2C1. **Koliko će čvorova generirati algoritam IDS prije nego što generira čvor sa stanjem A3C21**?

A 25 B 16 C 29 D 22

3 (T) Svi algoritmi slijepog pretraživanja u dubinu eksponencijalne su vremenske složenosti, međutim konkretni izrazi za asimptotsku složenost ipak se malo razlikuju. Asimptotska vremenska složenost algoritma pretraživanja u širinu (BFS) je $\mathcal{O}(b^{d+1})$, a algoritma iterativnog pretraživanja u dubinu (IDS) $\mathcal{O}(b^d)$. **Zašto asimptotska vremenska složenost algoritma IDS nije ista kao i ona algoritma BFS, $\mathcal{O}(b^{d+1})$?**

A IDS čvorove na razini d proširuje b^d puta, dok BFS dodatno proširuje svaki čvor na razini d osim ciljnog

B IDS ima dubinsko ograničenje koje povećava do dubine rješenja d, pa nikada ne proširuje čvorove razine d

C IDS više puta proširuje iste čvorove, ali na posljednoj razini proširuje sve čvorove osim ciljnog

D IDS generirane čvorove umeće na početak, a BFS na kraj liste, pa BFS na razini d generira b dodatnih čvorova

4 (P) Stanje s slagalice 3×3 neka je $[[1, \square, 2], [4, 5, 3], [7, 8, 6]]$, dok je ciljno stanje $[[1, 2, 3], [4, 5, 6], [7, 8, \square]]$. Neka je h optimistična heuristika. **Među ponuđenim vrijednostima, koju vrijednost $h(s)$ može poprimiti, a da je najviše obaviještena?**

A 2 B 9 C 8 D 5

5 (P) Oblikujemo računalnu igru u kojoj se računalno upravljani igrači kreću po 2D mapi i skupljaju raznorazne artefakte. U jednom trenutku igrači trebaju u što manje poteza doći do zajedničkog zbornog mjesta. Igrači se mogu kretati u četiri smjera (gore, dolje, lijevo, desno), pri čemu rade po jedan korak. Mapa sadržava prepreke (zidove), kroz koje igrači ne mogu proći. Mapa također sadržava slučajan broj slučajno raspoređenih teleportera, koji igrača mogu trenutačno (u samo jednom koraku) transportirati na slučajno mjesto na mapi. No, jednom upotrijebljeni teleporter više se ne može upotrijebiti. Za ovu igru želimo oblikovati optimističnu heuristiku koja će navoditi računalno upravljane igrače do zbornog mjesta. Za oblikovanje heuristike koristimo metodu relaksacije. **Koja od sljedećih relaksacija igre daje optimističnu heuristiku?**

A Igrači mogu prolaziti kroz zidove i dvaput koristiti isti teleporter

B Teleporteri se mogu koristiti neograničeno mnogo puta

C Teleporteri uvijek transportiraju igrača do teleportera koji je najbliži zbornom mjestu

D Igrači se mogu kretati dijagonalno, ali ne mogu koristiti teleportere

6 (R) Prostor stanja pretražujemo algoritmom A^*. Skup stanja je $S = \{a, b, c, d, e, f\}$, a funkcija sljedbenika je $succ(a) = \{(b, 2), (c, 2)\}$, $succ(b) = \{(c, 5), (d, 2)\}$, $succ(c) = \{(d, 1), (f, 20)\}$, $succ(d) = \{(e, 2)\}$, $succ(e) = \{(f, 14)\}$ te $succ(f) = \emptyset$. Heurističke vrijednosti stanja su $h(a) = 16$, $h(b) = 6$, $h(c) = 14$, $h(d) = 4$, $h(e) = 2$, $h(f) = 0$. Početno stanje je a, a ciljno f. Izvedite korake algoritma A^*, bilježeći u svakom koraku sadržaj liste otvorenih čvorova O i skupa zatvorenih čvorova C. U nultom koraku algoritma vrijedi $O = [(a, 0)]$ i $C = \emptyset$. **Koji je sadržaj liste O i skupa C nakon petog koraka izvođenja algoritma?**

A Algoritam ne dostiže peti korak

B $O = [(d, 4), (f, 20)]$, $C = \{(a, 16), (b, 6), (c, 14), (e, 2)\}$

C $O = [(d, 3)]$, $C = \{(a, 0), (b, 2), (c, 2), (e, 6)\}$

D $O = [(d, 3), (f, 20)]$, $C = \{(a, 0), (b, 2), (c, 2), (e, 6)\}$

7 (T) Prednost algoritma A^* nad algoritmom pretraživanja s jednolikom cijenom (UCS) jest ta da algoritam A^* koristi heuristiku, dok ju algoritam UCS ne koristi. **Kako se ta prednost manifestira u praksi?**

A Ako rješenje postoji, A^* će u manje koraka pronaći rješenje manje ili jednake cijene od onoga koje nalazi UCS

B Neovisno o svojstvima heuristike, A^* će ispitati jednak ili manji broj čvorova od onoga koji će ispitati UCS

C Ako je heuristika optimistična, A^* će pronaći put koji je manje cijene od onoga koji pronalazi UCS

D Ako je heuristika optimistična, A^* će generirati manji ili jednak broj čvorova od algoritma UCS

2. Igranje igara (3 pitanja)

8 (R) Razmatramo igru dvaju minimax-igrača. Stablo igre definirano je prijelazima $succ(A) = \{B, C, D\}$, $succ(B) = \{E, F\}$, $succ(C) = \{G, H\}$, $succ(D) = \{I, J\}$, $succ(G) = succ(F) = \{K, L\}$, $succ(H) = succ(E) = succ(I) = \{M, N\}$ te $succ(J) = \{O, P\}$. Igrač MAX koristi heuristiku h_1, a igrač MIN heuristiku h_2. Vrijednosti heuristika su sljedeće (obje heuristike definiraju dobit dotičnog igrača):

	E	F	G	H	I	J	K	L	M	N	O	P
h_1	1	−2	3	2	0	5	−1	0	−2	−1	0	2
h_2	0	5	2	4	−1	2	3	2	1	−1	1	−2

Oba igrača pretražuju najviše dva poteza unaprijed (jedan svoj i jedan suparnički). Igru započinje igrač MAX u stanju A. **Kroz koja stanja će se razvijati igra?**

A $A, C, H, \ldots$ B $A, C, G, \ldots$ C $A, D, I, \ldots$ D $A, D, J, \ldots$

9 (T) Igrač MAX primjenjuje algoritam minimax s heuristikom h na korijen stabla igre kako bi odredio svoj optimalan potez. Nakon što povuče taj potez, na redu je protivnik MIN, koji također vuče svoj optimalan potez. Igrač MAX zatim nanovo primjenjuje algoritam MAX na novom stanju igre. Pritom igrač MAX ponovo izračunava minimax-vrijednosti i za one čvorove za koje je ranije (za prethodni potez) već bio izračunao minimax-vrijednosti. **Zbog čega je potrebno ponovno izračunavati minimax-vrijednosti za te čvorove?**

A Ako je za prethodni izračun korištena heuristika, novi izračun bit će informiraniji jer će ići dvije razine dublje od prethodnog

B Heuristika h nije savršena, pa igrač MAX ne može očekivati da će ostvariti točno onu isplatu koju predviđa heuristika h

C Algoritam minimax radi pretraživanje u dubinu, pa ne pohranjuje čvorove na stazama izvan poteza koji je predvidio igrač MAX

D Protivnik MIN može koristi heuristiku manje točnosti od heuristike h, pa njegovi potezi mogu biti drugačiji od onih koje je predvidio igrač MAX

10 (P) Razmatramo dva igraća algoritma: jedan s heuristikom h_1 i dubinom pretraživanja d_1 te drugi s heuristikom h_2 i dubinom pretraživanja d_2. Oba algoritma koriste strategiju minimax. **Kada će prvi algoritam češće pobjeđivati drugi algoritam?**

A Ako $h_1 = h_2$ i $d_1 > d_2$ B Ako $d_1 = d_2$ i $h_1 > h_2$ C Ako $h_2 > h_1$ D Ako $d_1 > d_2$

3. Prikazivanje znanja i automatsko zaključivanje (6 pitanja)

11 (T) Logička posljedica središnji je pojam u logici i razlog zašto uopće koristimo sustav formalne logike za prikazivanje znanja. U propozicijskoj logici, logičku posljedicu moguće je dokazati na nekoliko načina. Jedan način temelji se na teoremu semantičke dedukcije. **Kako riječima glasi teorem semantičke dedukcije?**

A Ako su premise nezadovoljive, bilo koja formula logička je posljedica takvih premisa

B Ciljna formula je logička posljedica premisa ako i samo ako je ciljna formula istinita kada su istinite premise

C Premise i negacija cilja zajedno su zadovoljivi ako i samo ako je ciljna formula logička posljedica premisa

D Ciljna formula logička je posljedica premisa ako i samo ako je implikacija s premisa na ciljnu formulu tautologija

12 (P) Ispravnost je važno svojstvo pravila zaključivanja. **Koje je od sljedećih pravila zaključivanja ispravno?**

A $A \to B, B \to C \vdash C \to \neg A$ C $A \to B, B \to C \vdash C \to A$

B $A \to (B \vee C), \neg C \vdash A \to B$ D $\neg A, A \to B \vdash \neg B$

13 (T) Unifikacija je postupak svođenja izraza K_1 i K_2 iz FOL na identičan oblik primjenom odgovarajuće supstitucije varijabli, koju nazivamo "unifikator". Postupak unifikacije ključan je za zaključivanje u FOL. Neka α, β i θ označavaju supstitucije, a "$\circ$" kompoziciju dviju supstitucija. **Što mora vrijediti, a da je supstitucija θ najopćenitiji zajednički unifikator (MGU)?**

A $\forall\alpha\big((K_1\alpha = K_2\alpha) \to \exists\beta(\alpha = \theta \circ \beta)\big)$ C $\forall\beta\exists\alpha\big((K_1\alpha = K_2\alpha) \to (\beta = \theta \circ \alpha)\big)$

B $\exists\alpha\big((K_1\alpha = K_2\alpha) \to \forall\beta(\theta = \alpha \circ \beta)\big)$ D $\forall\alpha\forall\beta\big((K_1\theta = K_2\theta) \to (\theta = \alpha \circ \beta)\big)$

14 (R) Zadane su premise: *"Ivo voli sve vrste hrane. Jabuka i piletina su hrana. Branko jede lješnjake i nije mu zlo od njih. Vesna jede sve što i Branko. Ako netko nešto jede i nije mu od toga zlo, onda je to hrana."* Formalizirajte ove premise u FOL i pretvorite ih u klauzule. Pritom koristite $V(x, y)$ za "x voli y", $J(x, y)$ za "x jede y", "$Z(x, y)$" za "x-u je zlo od y", i "$H(x)$" "x je hrana". Zatim rezolucijom opovrgavanjem uz strategiju skupa potpore dokažite cilj *Ivo voli lješnjake*. **Koliko je minimalno koraka (primjena rezolucijskog pravila) potrebno za dokaz?**

A 3 B 5 C 6 D 4

15 (P) Na postupak dokazivanja rezolucijom opovrgavanjem možemo gledati kao na problem pretraživanje prostora stanja, gdje stanja odgovaraju skupu klauzula (onih zadanih i onih izvedenih), a prijelazi između stanja odgovaraju primjeni rezolucijskog pravila na jedan par klauzula. Neka skup premisa sadrži 8 klauzula, a negirani cilj 2 klauzule. Za dokazivanje koristimo dvije strategije: strategiju zasićenja po razinama (LSS) i strategiju skupa potpore (SOS). Jednom razriješeni par klauzula više se ne razrješava. Izračunajte gornju ogradu na ukupan broj primjena rezolucijskog pravila za LSS i za SOS, ako znamo da je cilj dokaziv u tri koraka (tri primjene rezolucijskog pravila). **Koliko je puta gornja ograda za LSS veća od gornje ograde za SOS?**

A 5.71× B 4.53× C 9.65× D 7.51×

16 (R) Dani su FOL atomi $P(f(x), y, g(w, b))$ i $P(y, f(g(a, z)), x)$, gdje je P predikat, a i b su konstante, a f i g su funkcije. **Što je najopćenitiji zajednički unifikator ovih dvaju atoma?**

A Atome nije moguće unificirati C $P(f(f(g(a, b))), f(g(a, b)), f(g(a, b)))$

B $P(f(g(a, b)), f(g(a, b)), g(a, b))$ D $P(f(x), f(g(x, y)), g(a, b))$

4. Logičko programiranje i ekspertni sustavi (4 pitanja)

17 (R) Baza znanja ekspertnog sustava sadrži sljedeća pravila:

(1) AKO $(A = a_2) \wedge (D = d_2)$ ONDA $C = c_1$
(2) AKO $(F = f_3) \vee (B = b_3)$ ONDA $C = c_2$
(3) AKO $(E = e_1) \vee (B = b_1)$ ONDA $(A = a_1) \wedge (D = d_2)$
(4) AKO $F = f_1$ ONDA $D = d_2$
(5) AKO $F = f_2$ ONDA $E = e_2$
(6) AKO $(B = b_3) \vee (D = d_1)$ ONDA $(E = e_1) \wedge (A = a_2)$

Sustav koristimo za izvođenje vrijednosti varijable C ulančavanjem unazad. Prednost imaju pravila s manjim rednim brojem. Pravila koja su jednom palila više ne mogu paliti. Jednom izvedeni međuciljevi brišu se sa stoga,

neovisno o njihovoj poziciji na stogu. Na možebitne upite od strane sustava, korisnik odgovara sa $B = b_3$ i $F = f_1$. **Što radi ekspertni sustav pri izvođenju vrijednosti varijable C?**

A Odbacuje pravilo 3 te kasnije pali pravilo 4

B Izvodi $A = a_2$ te kasnije $A = a_1$

C Završava sa 4 činjenica u radnoj memoriji

D Izvodi $E = e_1$ te kasnije $E = e_2$

18 (T) Prolog je deklarativan programski jezik, ali sadrži neke nedeklarativne aspekte. Jedan aspekt nedeklarativnosti u Prologu jest taj da u nekim slučajevima ne vrijedi komutativnost operatora "$\wedge$", npr., program `P(X,Y) :- Q(X), R(Y)` i program `P(X,Y) :- R(Y), Q(X)` ne evaluiraju se nužno identično. **Zbog čega u Prologu u ovom slučaju može postojati odstupanje proceduralnog značenja od deklarativnog?**

A Prolog pretpostavlja zatvorenost svijeta, no u stvarnosti činjenice koje nisu u bazi znanja mogu biti istinite

B Prolog koristi definitne klauzule, pa do odstupanja dolazi kada neku FOL formulu ne možemo izraziti u Prologu

C Prolog ne provodi standardizaciju klauzula, pa različiti redoslijed klauzula može dati različite unifikatore

D Prolog koristi rezoluciju SLD i pretraživanje u dubinu, pa dokazivanje ovisi o redoslijedu atoma na stogu

19 (T) Zaključivanje u ekspertnim sustavima može biti izvedeno kao ulančavanje unaprijed ili ulančavanje unazad. **U kojim je primjenama zaključivanje unazad bolji izbor od ulančavanja unaprijed?**

A Kada je znanje moguće dekomponirati u granule znanja i formalizirati ih ako–onda pravilima

B Kada novoizvedeni zaključci mogu invalidirati ranije izvedene zaključke

C Kada ako–onda pravila imaju različite prioritete koji se mogu mijenjati tijekom izvođenja programa

D Kada je broj mogućih hipoteza malen, a broj potpora (dokaza) za te hipoteze potencijalno velik

20 (R) Razmatramo sljedeći program u Prologu:

```
roditelj(kronos, zeus).
roditelj(zeus, minos).
roditelj(europa, minos).
potomak(X, Y) :- roditelj(Y, X).
potomak(X, Y) :- roditelj(Y, Z), potomak(X, Z).
bog(kronos).
bog(X) :- potomak(X, Y), bog(Y).
polubog(X) :- potomak(X, Y), not(bog(Y)).
```

Nad ovako definiranom bazom izvodimo upit `polubog(minos)`. Nacrtajte Prologovo stablo dokaza za ovaj upit do trenutka drugog povratka u pretraživanju. Pritom za dokazivanje negiranog cilja koristite dva čvora (jedan za `not(P(X))` i drugi za `P(X)`), dok prazni stog ne brojite kao čvor. **Koliko čvorova ima stablo dokaza neposredno prije drugog povratka u pretraživanju?**

A 10 B 6 C 12 D 8

Međuispit iz Uvoda u umjetnu inteligenciju (2021./2022.)

– NEKORIGIRANA VERZIJA –

*Ispit se sastoji od **20 pitanja** i ukupno nosi **20 bodova**. Točan odgovor nosi 1 bod, a netočan -1/3 boda. Trajanje ispita je **150 minuta**. Primjerak ispita trebate predati zajedno sa svojim rješenjima.*

1. Pretraživanje prostora stanja i heurističko pretraživanje (7 pitanja)

1 (P) Oblikujemo računalnu igru u kojoj se računalno upravljani igrači kreću po 2D mapi i skupljaju raznorazne artefakte. U jednom trenutku igrači trebaju u što manje poteza doći do zajedničkog zbornog mjesta. Igrači se mogu kretati u četiri smjera (gore, dolje, lijevo, desno), pri čemu rade po jedan korak. Mapa sadržava prepreke (zidove), kroz koje igrači ne mogu proći. Mapa također sadržava slučajan broj slučajno raspoređenih teleportera, koji igrača mogu trenutačno (u samo jednom koraku) transportirati na slučajno mjesto na mapi. No, jednom upotrijebljeni teleporter više se ne može upotrijebiti. Za ovu igru želimo oblikovati optimističnu heuristiku koja će navoditi računalno upravljane igrače do zbornog mjesta. Za oblikovanje heuristike koristimo metodu relaksacije. **Koja od sljedećih relaksacija igre daje optimističnu heuristiku?**

- A Igrači mogu prolaziti kroz zidove i dvaput koristiti isti teleporter
- B Teleporteri uvijek transportiraju igrača do teleportera koji je najbliži zbornom mjestu
- C Teleporteri se mogu koristiti neograničeno mnogo puta
- D Igrači se mogu kretati dijagonalno, ali ne mogu koristiti teleportere

2 (T) Svi algoritmi slijepog pretraživanja u dubinu eksponencijalne su vremenske složenosti, međutim konkretni izrazi za asimptotsku složenost ipak se malo razlikuju. Asimptotska vremenska složenost algoritma pretraživanja u širinu (BFS) je $\mathcal{O}(b^{d+1})$, a algoritma iterativnog pretraživanja u dubinu (IDS) $\mathcal{O}(b^d)$. **Zašto asimptotska vremenska složenost algoritma IDS nije ista kao i ona algoritma BFS,** $\mathcal{O}(b^{d+1})$?

- A IDS generirane čvorove umeće na početak, a BFS na kraj liste, pa BFS na razini d generira b dodatnih čvorova
- B IDS čvorove na razini d proširuje b^d puta, dok BFS dodatno proširuje svaki čvor na razini d osim ciljnog
- C IDS više puta proširuje iste čvorove, ali na posljednoj razini proširuje sve čvorove osim ciljnog
- D IDS ima dubinsko ograničenje koje povećava do dubine rješenja d, pa nikada ne proširuje čvorove razine d

3 (T) Prednost algoritma A^* nad algoritmom pretraživanja s jednolikom cijenom (UCS) jest ta da algoritam A^* koristi heuristiku, dok ju algoritam UCS ne koristi. **Kako se ta prednost manifestira u praksi?**

- A Ako je heuristika optimistična, A^* će generirati manji ili jednak broj čvorova od algoritma UCS
- B Ako rješenje postoji, A^* će u manje koraka pronaći rješenje manje ili jednake cijene od onoga koje nalazi UCS
- C Neovisno o svojstvima heuristike, A^* će ispitati jednak ili manji broj čvorova od onoga koji će ispitati UCS
- D Ako je heuristika optimistična, A^* će pronaći put koji je manje cijene od onoga koji pronalazi UCS

4 (R) Algoritmom slijepog pretraživanja rješavamo problem Hanojskih tornjeva s tri klinova i tri diska. Klinovi su označeni sa A, B i C, a diskovi sa 1, 2 i 3, pri čemu broj odgovara veličini diska (disk 1 je najmanji, a disk 3 najveći). Stanje možemo označiti kao AxByCz, gdje x, y i z označavaju diskove koji se nalaze na klinovima A, B, odnosno C, krenuvši od dna prema vrhu (ako je klin prazan, ispuštamo njegovo slovo). Npr., u stanju B21C3 na klinu B su diskovi 2 (ispod) i 1 (gore), a na klinu C je disk 3. Početno stanje je A321. Zadatak je prebaciti diskove s klina A na klin C, tj. pronaći stanje C321, koristeći klin B kao pomoćni klin. Pritom se u svakom koraku prebacuje samo jedan disk, uz uvjet da veći disk ne smije biti položen na manji disk. Koristimo algoritam iterativnog pretraživanja u dubinu (IDS) bez liste posjećenih stanja. Algoritam čvorove proširuje leksikografskim poretkom gdje znamenke dolaze prije slova, npr., stanje A32B1 prethodi stanju A3B2C1. **Koliko će čvorova generirati algoritam IDS prije nego što generira čvor sa stanjem A3C21**?

A 25 B 16 C 22 D 29

5 (R) Prostor stanja pretražujemo algoritmom A^*. Skup stanja je $S = \{a, b, c, d, e, f\}$, a funkcija sljedbenika je $succ(a) = \{(b, 2), (c, 2)\}$, $succ(b) = \{(c, 5), (d, 2)\}$, $succ(c) = \{(d, 1), (f, 20)\}$, $succ(d) = \{(e, 2)\}$, $succ(e) = \{(f, 14)\}$ te $succ(f) = \emptyset$. Heurističke vrijednosti stanja su $h(a) = 16$, $h(b) = 6$, $h(c) = 14$, $h(d) = 4$, $h(e) = 2$, $h(f) = 0$. Početno stanje je a, a ciljno f. Izvedite korake algoritma A^*, bilježeći u svakom koraku sadržaj liste otvorenih čvorova O i skupa zatvorenih čvorova C. U nultom koraku algoritma vrijedi $O = [(a, 0)]$ i $C = \emptyset$. **Koji je sadržaj liste O i skupa C nakon šestog koraka izvođenja algoritma?**

A Algoritam ne dostiže šesti korak

B $O = [(e, 5), (f, 20)]$, $C = \{(a, 0), (b, 2), (c, 7)\}$

C $O = []$, $C = \{(a, 0), (b, 2), (c, 2), (d, 3), (e, 5)\}$

D $O = [(e, 5), (f, 20)]$, $C = \{(a, 0), (b, 2), (c, 2), (d, 3)\}$

6 (P) Razmatramo problem slagalice dimenzija 4×4 s prosječnom dubinom rješenja $d = 12$. Za nalaženje rješenja algoritam pretraživanja u dubinu u prosjeku koristi 720 B memorije. **Koliko bi memorije u prosjeku koristio algoritam pretraživanja u širinu na ovome problemu?**

A 778.5 KB B 240 MB C 10.1 MB D 320 MB

7 (P) Stanje s slagalice 3×3 neka je $[[1, 5, 2], [4, \square, 3], [7, 8, 6]]$, dok je ciljno stanje $[[1, 2, 3], [4, 5, 6], [7, 8, \square]]$. Neka je h optimistična heuristika. **Među ponuđenim vrijednostima, koju vrijednost $h(s)$ može poprimiti, a da je najviše obaviještena?**

A 0 B 1 C 5 D 3

2. Igranje igara (3 pitanja)

8 (R) Razmatramo igru dvaju minimax-igrača. Stablo igre definirano je prijelazima $succ(A) = \{B, C, D\}$, $succ(B) = \{E, F\}$, $succ(C) = \{G, H\}$, $succ(D) = \{I, J\}$, $succ(G) = succ(F) = \{K, L\}$, $succ(H) = succ(E) = succ(I) = \{M, N\}$ te $succ(J) = \{O, P\}$. Igrač MAX koristi heuristiku h_1, a igrač MIN heuristiku h_2. Vrijednosti heuristika su sljedeće (obje heuristike definiraju dobit dotičnog igrača):

	E	F	G	H	I	J	K	L	M	N	O	P
h_1	1	−2	3	2	0	5	−1	0	−2	−1	0	2
h_2	0	5	2	4	−1	2	3	2	1	−1	1	−2

Oba igrača pretražuju najviše dva poteza unaprijed (jedan svoj i jedan suparnički). Igru započinje igrač MAX u stanju A. **Kroz koja stanja će se razvijati igra?**

A $A, D, J, \ldots$ B $A, C, H, \ldots$ C $A, C, G, \ldots$ D $A, B, E, \ldots$

9 (T) Igrač MAX primjenjuje algoritam minimax s heuristikom h na korijen stabla igre kako bi odredio svoj optimalan potez. Nakon što povuče taj potez, na redu je protivnik MIN, koji također vuče svoj optimalan potez. Igrač MAX zatim nanovo primjenjuje algoritam MAX na novom stanju igre. Pritom igrač MAX ponovo izračunava minimax-vrijednosti i za one čvorove za koje je ranije (za prethodni potez) već bio izračunao minimax-vrijednosti. **Zbog čega je potrebno ponovno izračunavati minimax-vrijednosti za te čvorove?**

A Algoritam minimax radi pretraživanje u dubinu, pa ne pohranjuje čvorove na stazama izvan poteza koji je predvidio igrač MAX

B Ako je za prethodni izračun korištena heuristika, novi izračun bit će informiraniji jer će ići dvije razine dublje od prethodnog

C Protivnik MIN može koristi heuristiku manje točnosti od heuristike h, pa njegovi potezi mogu biti drugačiji od onih koje je predvidio igrač MAX

D Heuristika h nije savršena, pa igrač MAX ne može očekivati da će ostvariti točno onu isplatu koju predviđa heuristika h

10 (P) Razmatramo dva igraća algoritma: jedan s heuristikom h_1 i dubinom pretraživanja d_1 te drugi s heuristikom h_2 i dubinom pretraživanja d_2. Oba algoritma koriste strategiju minimax. **Kada će prvi algoritam češće pobjeđivati drugi algoritam?**

A Ako $h_1 = h_2$ i $d_1 > d_2$ B Ako $d_1 > d_2$ C Ako $d_2 < d_1$ D Ako $d_1 = d_2$ i $h_1 < h_2$

3. Prikazivanje znanja i automatsko zaključivanje (6 pitanja)

11 (R) Dani su FOL atomi $P(f(x), y, g(w, b))$ i $P(y, f(g(a, z)), x)$, gdje je P predikat, a i b su konstante, a f i g su funkcije. **Što je najopćenitiji zajednički unifikator ovih dvaju atoma?**

A $P(f(f(g(a,b))), f(g(a,b)), f(g(a,b)))$
B Atome nije moguće unificirati
C $P(f(x), f(g(x,y)), g(a,b))$
D $P(f(g(a,b)), f(g(a,b)), g(a,b))$

12 (T) Logička posljedica središnji je pojam u logici i razlog zašto uopće koristimo sustav formalne logike za prikazivanje znanja. U propozicijskoj logici, logičku posljedicu moguće je dokazati na nekoliko načina. Jedan način temelji se na teoremu semantičke dedukcije. **Kako riječima glasi teorem semantičke dedukcije?**

A Ako su premise nezadovoljive, bilo koja formula logička je posljedica takvih premisa
B Ciljna formula logička je posljedica premisa ako i samo ako je implikacija s premisa na ciljnu formulu tautologija
C Premise i negacija cilja zajedno su zadovoljivi ako i samo ako je ciljna formula logička posljedica premisa
D Ciljna formula je logička posljedica premisa ako i samo ako je ciljna formula istinita kada su istinite premise

13 (P) Ispravnost je važno svojstvo pravila zaključivanja. **Koje je od sljedećih pravila zaključivanja ispravno?**

A $A \to B, B \to C \vdash C \to A$
B $A \to (B \to C) \vdash (A \vee B) \to C$
C $A \to B, B \to C \vdash C \to \neg A$
D $A \to (B \vee C), \neg C \vdash A \to B$

14 (T) Unifikacija je postupak svođenja izraza K_1 i K_2 iz FOL na identičan oblik primjenom odgovarajuće supstitucije varijabli, koju nazivamo "unifikator". Postupak unifikacije ključan je za zaključivanje u FOL. Neka α, β i θ označavaju supstitucije, a "$\circ$" kompoziciju dviju supstitucija. **Što mora vrijediti, a da je supstitucija θ najopćenitiji zajednički unifikator (MGU)?**

A $\forall\beta\exists\alpha\big((K_1\alpha = K_2\alpha) \to (\beta = \theta \circ \alpha)\big)$
B $\forall\alpha\big((K_1\alpha = K_2\alpha) \to \exists\beta(\alpha = \theta \circ \beta)\big)$
C $\forall\alpha\forall\beta\big((K_1\theta = K_2\theta) \to (\theta = \alpha \circ \beta)\big)$
D $\exists\alpha\big((K_1\alpha = K_2\alpha) \to \forall\beta(\theta = \alpha \circ \beta)\big)$

15 (P) Na postupak dokazivanja rezolucijom opovrgavanjem možemo gledati kao na problem pretraživanje prostora stanja, gdje stanja odgovaraju skupu klauzula (onih zadanih i onih izvedenih), a prijelazi između stanja odgovaraju primjeni rezolucijskog pravila na jedan par klauzula. Neka skup premisa sadrži 7 klauzula, a negirani cilj 2 klauzule. Za dokazivanje koristimo dvije strategije: strategiju zasićenja po razinama (LSS) i strategiju skupa potpore (SOS). Jednom razriješeni par klauzula više se ne razrješava. Izračunajte gornju ograda na ukupan broj primjena rezolucijskog pravila za LSS i za SOS, ako znamo da je cilj dokaziv u tri koraka (tri primjene rezolucijskog pravila). **Koliko je puta gornja ograda za LSS veća od gornje ograde za SOS?**

A $9.65\times$ B $7.51\times$ C $5.71\times$ D $4.53\times$

16 (R) Zadane su premise: *"Ivo voli sve vrste hrane. Jabuka i piletina su hrana. Branko jede lješnjake i nije mu zlo od njih. Vesna jede sve što i Branko. Ako netko nešto jede i nije mu od toga zlo, onda je to hrana."* Formalizirajte ove premise u FOL i pretvorite ih u klauzule. Pritom koristite $V(x, y)$ za "x voli y", $J(x, y)$ za "x jede y", "$Z(x, y)$" za "x-u je zlo od y", i "$H(x)$" "x je hrana". Zatim rezolucijom opovrgavanjem uz strategiju skupa potpore dokažite cilj *Ivo voli lješnjake*. **Koliko je minimalno koraka (primjena rezolucijskog pravila) potrebno za dokaz?**

A 5 B 3 C 6 D 4

4. Logičko programiranje i ekspertni sustavi (4 pitanja)

17 (R) Baza znanja ekspertnog sustava sadrži sljedeća pravila:

(1) AKO $(A = a_2) \wedge (D = d_2)$ ONDA $C = c_1$
(2) AKO $(F = f_3) \vee (B = b_3)$ ONDA $C = c_2$
(3) AKO $(E = e_1) \vee (B = b_1)$ ONDA $(A = a_1) \wedge (D = d_2)$
(4) AKO $F = f_1$ ONDA $D = d_2$
(5) AKO $F = f_2$ ONDA $E = e_2$
(6) AKO $(B = b_3) \vee (D = d_1)$ ONDA $(E = e_1) \wedge (A = a_2)$

Sustav koristimo za izvođenje vrijednosti varijable C ulančavanjem unazad. Prednost imaju pravila s manjim rednim brojem. Pravila koja su jednom palila više ne mogu paliti. Jednom izvedeni međuciljevi brišu se sa stoga,

neovisno o njihovoj poziciji na stogu. Na možebitne upite od strane sustava, korisnik odgovara sa $B = b_3$ i $F = f_1$. **Što radi ekspertni sustav pri izvođenju vrijednosti varijable C?**

A Pali tri pravila i izvodi $C = c_2$

B Završava sa 3 činjenice u radnoj memoriji

C Odbacuje pravilo 6 te kasnije pali pravilo 3

D Odbacuje pravilo 2 te kasnije pali pravilo 5

18 (R) Razmatramo sljedeći program u Prologu:

```
roditelj(kronos, zeus).
roditelj(zeus, minos).
roditelj(europa, minos).
potomak(X, Y) :- roditelj(Y, X).
potomak(X, Y) :- roditelj(Y, Z), potomak(X, Z).
bog(kronos).
bog(X) :- potomak(X, Y), bog(Y).
polubog(X) :- potomak(X, Y), not(bog(Y)).
```

Nad ovako definiranom bazom izvodimo upit `polubog(minos)`. Nacrtajte Prologovo stablo dokaza za ovaj upit do trenutka drugog povratka u pretraživanju. Pritom za dokazivanje negiranog cilja koristite dva čvora (jedan za `not(P(X))` i drugi za `P(X)`), dok prazni stog ne brojite kao čvor. **Koliko čvorova ima stablo dokaza neposredno prije drugog povratka u pretraživanju?**

A 6 B 8 C 10 D 12

19 (T) Prolog je deklarativan programski jezik, ali sadrži neke nedeklarativne aspekte. Jedan aspekt nedeklarativnosti u Prologu jest taj da u nekim slučajevima ne vrijedi komutativnost operatora "$\wedge$", npr., program `P(X,Y) :- Q(X), R(Y)` i program `P(X,Y) :- R(Y), Q(X)` ne evaluiraju se nužno identično. **Zbog čega u Prologu u ovom slučaju može postojati odstupanje proceduralnog značenja od deklarativnog?**

A Prolog pretpostavlja zatvorenost svijeta, no u stvarnosti činjenice koje nisu u bazi znanja mogu biti istinite

B Prolog koristi rezoluciju SLD i pretraživanje u dubinu, pa dokazivanje ovisi o redoslijedu atoma na stogu

C Prolog ne provodi standardizaciju klauzula, pa različiti redoslijed klauzula može dati različite unifikatore

D Prolog koristi definitne klauzule, pa do odstupanja dolazi kada neku FOL formulu ne možemo izraziti u Prologu

20 (T) Zaključivanje u ekspertnim sustavima može biti izvedeno kao ulančavanje unaprijed ili ulančavanje unazad. **U kojim je primjenama zaključivanje unazad bolji izbor od ulančavanja unaprijed?**

A Kada je broj mogućih hipoteza malen, a broj potpora (dokaza) za te hipoteze potencijalno velik

B Kada ako–onda pravila imaju različite prioritete koji se mogu mijenjati tijekom izvođenja programa

C Kada novoizvedeni zaključci mogu invalidirati ranije izvedene zaključke

D Kada je znanje moguće dekomponirati u granule znanja i formalizirati ih ako–onda pravilima

Međuispit iz Uvoda u umjetnu inteligenciju (2021./2022.)

– NEKORIGIRANA VERZIJA –

*Ispit se sastoji od **20 pitanja** i ukupno nosi **20 bodova**. Točan odgovor nosi 1 bod, a netočan -1/3 boda. Trajanje ispita je **150 minuta**. Primjerak ispita trebate predati zajedno sa svojim rješenjima.*

1. Pretraživanje prostora stanja i heurističko pretraživanje (7 pitanja)

1 (R) Algoritmom slijepog pretraživanja rješavamo problem Hanojskih tornjeva s tri klinova i tri diska. Klinovi su označeni sa A, B i C, a diskovi sa 1, 2 i 3, pri čemu broj odgovara veličini diska (disk 1 je najmanji, a disk 3 najveći). Stanje možemo označiti kao AxByCz, gdje x, y i z označavaju diskove koji se nalaze na klinovima A, B, odnosno C, krenuvši od dna prema vrhu (ako je klin prazan, ispuštamo njegovo slovo). Npr., u stanju B21C3 na klinu B su diskovi 2 (ispod) i 1 (gore), a na klinu C je disk 3. Početno stanje je A321. Zadatak je prebaciti diskove s klina A na klin C, tj. pronaći stanje C321, koristeći klin B kao pomoćni klin. Pritom se u svakom koraku prebacuje samo jedan disk, uz uvjet da veći disk ne smije biti položen na manji disk. Koristimo algoritam iterativnog pretraživanja u dubinu (IDS) bez liste posjećenih stanja. Algoritam čvorove proširuje leksikografskim poretkom gdje znamenke dolaze prije slova, npr., stanje A32B1 prethodi stanju A3B2C1. **Koliko će čvorova generirati algoritam IDS prije nego što generira čvor sa stanjem A3C21**?

A 16 B 29 C 25 D 22

2 (T) Svi algoritmi slijepog pretraživanja u dubinu eksponencijalne su vremenske složenosti, međutim konkretni izrazi za asimptotsku složenost ipak se malo razlikuju. Asimptotska vremenska složenost algoritma pretraživanja u širinu (BFS) je $\mathcal{O}(b^{d+1})$, a algoritma iterativnog pretraživanja u dubinu (IDS) $\mathcal{O}(b^d)$. **Zašto asimptotska vremenska složenost algoritma IDS nije ista kao i ona algoritma BFS,** $\mathcal{O}(b^{d+1})$?

A IDS ima dubinsko ograničenje koje povećava do dubine rješenja d, pa nikada ne proširuje čvorove razine d

B IDS više puta proširuje iste čvorove, ali na posljednoj razini proširuje sve čvorove osim ciljnog

C IDS čvorove na razini d proširuje b^d puta, dok BFS dodatno proširuje svaki čvor na razini d osim ciljnog

D IDS generirane čvorove umeće na početak, a BFS na kraj liste, pa BFS na razini d generira b dodatnih čvorova

3 (R) Prostor stanja pretražujemo algoritmom A^*. Skup stanja je $S = \{a, b, c, d, e, f\}$, a funkcija sljedbenika je $succ(a) = \{(b, 2), (c, 2)\}$, $succ(b) = \{(c, 5), (d, 2)\}$, $succ(c) = \{(d, 1), (f, 20)\}$, $succ(d) = \{(e, 2)\}$, $succ(e) = \{(f, 14)\}$ te $succ(f) = \emptyset$. Heurističke vrijednosti stanja su $h(a) = 16$, $h(b) = 6$, $h(c) = 14$, $h(d) = 4$, $h(e) = 2$, $h(f) = 0$. Početno stanje je a, a ciljno f. Izvedite korake algoritma A^*, bilježeći u svakom koraku sadržaj liste otvorenih čvorova O i skupa zatvorenih čvorova C. U nultom koraku algoritma vrijedi $O = [(a, 0)]$ i $C = \emptyset$. **Koji je sadržaj liste O i skupa C nakon šestog koraka izvođenja algoritma?**

A $O = [(e, 5), (f, 20)]$, $C = \{(a, 0), (b, 2), (c, 2), (d, 3)\}$

B $O = [(e, 5), (f, 22)]$, $C = \{(a, 0), (b, 2), (c, 7), (d, 3)\}$

C $O = [(e, 5), (f, 20)]$, $C = \{(a, 0), (b, 2), (c, 7)\}$

D Algoritam ne dostiže šesti korak

4 (P) Oblikujemo računalnu igru u kojoj se računalno upravljani igrači kreću po 2D mapi i skupljaju raznorazne artefakte. U jednom trenutku igrači trebaju u što manje poteza doći do zajedničkog zbornog mjesta. Igrači se mogu kretati u četiri smjera (gore, dolje, lijevo, desno), pri čemu rade po jedan korak. Mapa sadržava prepreke (zidove), kroz koje igrači ne mogu proći. Mapa također sadržava slučajan broj slučajno raspoređenih teleportera, koji igrača mogu trenutačno (u samo jednom koraku) transportirati na slučajno mjesto na mapi. No, jednom upotrijebljeni teleporter više se ne može upotrijebiti. Za ovu igru želimo oblikovati optimističnu heuristiku koja će

navoditi računalno upravljane igrače do zbornog mjesta. Za oblikovanje heuristike koristimo metodu relaksacije. **Koja od sljedećih relaksacija igre daje optimističnu heuristiku?**

A Igrači se mogu kretati dijagonalno, ali ne mogu koristiti teleportere

B Teleporteri se mogu koristiti neograničeno mnogo puta

C Teleporteri uvijek transportiraju igrača do teleportera koji je najbliži zbornom mjestu

D Igrači mogu prolaziti kroz zidove i dvaput koristiti isti teleporter

5 (P) Razmatramo problem slagalice dimenzija 3×3 s prosječnom dubinom rješenja $d = 11$. Za nalaženje rješenja algoritam pretraživanja u dubinu u prosjeku koristi 720 B memorije. **Koliko bi memorije u prosjeku koristio algoritam pretraživanja u širinu na ovome problemu?**

A 1.1 MB B 1.8 MB C 1.6 KB D 65 MB

6 (T) Prednost algoritma A^* nad algoritmom pretraživanja s jednolikom cijenom (UCS) jest ta da algoritam A^* koristi heuristiku, dok ju algoritam UCS ne koristi. **Kako se ta prednost manifestira u praksi?**

A Ako je heuristika optimistična, A^* će generirati manji ili jednak broj čvorova od algoritma UCS

B Neovisno o svojstvima heuristike, A^* će ispitati jednak ili manji broj čvorova od onoga koji će ispitati UCS

C Ako rješenje postoji, A^* će u manje koraka pronaći rješenje manje ili jednake cijene od onoga koje nalazi UCS

D Ako je heuristika optimistična, A^* će pronaći put koji je manje cijene od onoga koji pronalazi UCS

7 (P) Stanje s slagalice 3×3 neka je $[[1, 2, \square], [4, 5, 3], [7, 8, 6]]$, dok je ciljno stanje $[[1, 2, 3], [4, 5, 6], [7, 8, \square]]$. Neka je h optimistična heuristika. **Među ponuđenim vrijednostima, koju vrijednost $h(s)$ može poprimiti, a da je najviše obaviještena?**

A 5 B 1 C 4 D 6

2. Igranje igara (3 pitanja)

8 (T) Igrač MAX primjenjuje algoritam minimax s heuristikom h na korijen stabla igre kako bi odredio svoj optimalan potez. Nakon što povuče taj potez, na redu je protivnik MIN, koji također vuče svoj optimalan potez. Igrač MAX zatim nanovo primjenjuje algoritam MAX na novom stanju igre. Pritom igrač MAX ponovo izračunava minimax-vrijednosti i za one čvorove za koje je ranije (za prethodni potez) već bio izračunao minimax-vrijednosti. **Zbog čega je potrebno ponovno izračunavati minimax-vrijednosti za te čvorove?**

A Algoritam minimax radi pretraživanje u dubinu, pa ne pohranjuje čvorove na stazama izvan poteza koji je predvidio igrač MAX

B Heuristika h nije savršena, pa igrač MAX ne može očekivati da će ostvariti točno onu isplatu koju predviđa heuristika h

C Protivnik MIN može koristi heuristiku manje točnosti od heuristike h, pa njegovi potezi mogu biti drugačiji od onih koje je predvidio igrač MAX

D Ako je za prethodni izračun korištena heuristika, novi izračun bit će informiraniji jer će ići dvije razine dublje od prethodnog

9 (R) Razmatramo igru dvaju minimax-igrača. Stablo igre definirano je prijelazima $succ(A) = \{B, C, D\}$, $succ(B) = \{E, F\}$, $succ(C) = \{G, H\}$, $succ(D) = \{I, J\}$, $succ(G) = succ(F) = \{K, L\}$, $succ(H) = succ(E) = succ(I) = \{M, N\}$ te $succ(J) = \{O, P\}$. Igrač MAX koristi heuristiku h_1, a igrač MIN heuristiku h_2. Vrijednosti heuristika su sljedeće (obje heuristike definiraju dobit dotičnog igrača):

	E	F	G	H	I	J	K	L	M	N	O	P
h_1	1	−2	3	2	0	5	−1	0	−2	−1	0	2
h_2	0	5	2	4	−1	2	3	2	1	−1	1	−2

Oba igrača pretražuju najviše dva poteza unaprijed (jedan svoj i jedan suparnički). Igru započinje igrač MAX u stanju A. **Kroz koja stanja će se razvijati igra?**

A $A, C, G, \ldots$ B $A, B, E, \ldots$ C $A, D, I, \ldots$ D $A, C, H, \ldots$

10 (P) Razmatramo dva igraća algoritma: jedan s heuristikom h_1 i dubinom pretraživanja d_1 te drugi s heuristikom h_2 i dubinom pretraživanja d_2. Oba algoritma koriste strategiju minimax. **Kada će prvi algoritam češće pobjeđivati drugi algoritam?**

A Ako $d_1 = d_2$ i $h_1 < h_2$ B Ako $h_1 = h_2$ i $d_1 > d_2$ C Ako $h_1 = h_2$ i $d_1 < d_2$ D Ako $d_2 < d_1$

3. Prikazivanje znanja i automatsko zaključivanje (6 pitanja)

11 (P) Na postupak dokazivanja rezolucijom opovrgavanjem možemo gledati kao na problem pretraživanje prostora stanja, gdje stanja odgovaraju skupu klauzula (onih zadanih i onih izvedenih), a prijelazi između stanja odgovaraju primjeni rezolucijskog pravila na jedan par klauzula. Neka skup premisa sadrži 8 klauzula, a negirani cilj 2 klauzule. Za dokazivanje koristimo dvije strategije: strategiju zasićenja po razinama (LSS) i strategiju skupa potpore (SOS). Jednom razriješeni par klauzula više se ne razrješava. Izračunajte gornju ogradu na ukupan broj primjena rezolucijskog pravila za LSS i za SOS, ako znamo da je cilj dokaziv u tri koraka (tri primjene rezolucijskog pravila). **Koliko je puta gornja ograda za LSS veća od gornje ograde za SOS?**

A $4.53\times$ B $9.65\times$ C $7.51\times$ D $5.71\times$

12 (T) Logička posljedica središnji je pojam u logici i razlog zašto uopće koristimo sustav formalne logike za prikazivanje znanja. U propozicijskoj logici, logičku posljedicu moguće je dokazati na nekoliko načina. Jedan način temelji se na teoremu semantičke dedukcije. **Kako riječima glasi teorem semantičke dedukcije?**

A Ciljna formula je logička posljedica premisa ako i samo ako je ciljna formula istinita kada su istinite premise

B Premise i negacija cilja zajedno su zadovoljivi ako i samo ako je ciljna formula logička posljedica premisa

C Ako su premise nezadovoljive, bilo koja formula logička je posljedica takvih premisa

D Ciljna formula logička je posljedica premisa ako i samo ako je implikacija s premisa na ciljnu formulu tautologija

13 (P) Ispravnost je važno svojstvo pravila zaključivanja. **Koje je od sljedećih pravila zaključivanja ispravno?**

A $A \to (B \to C) \vdash (A \vee B) \to C$ C $B, A \to B \vdash A$

B $A \to (B \vee C), \neg C \vdash A \to B$ D $A \vee B \vdash B$

14 (R) Dani su FOL atomi $P(f(x), y, g(w, f(z)))$ i $P(y, f(g(a, z)), x)$, gdje je P predikat, a je konstanta, a f i g su funkcije. **Što je najopćenitji zajednički unifikator ovih dvaju atoma?**

A Atome nije moguće unificirati C $P(f(f(g(a,b))), f(g(a,b)), f(g(a,b)))$

B $P(f(y), f(g(a,z)), g(w,b))$ D $P(f(g(a,b)), f(g(a,b)), g(a,b))$

15 (T) Unifikacija je postupak svođenja izraza K_1 i K_2 iz FOL na identičan oblik primjenom odgovarajuće supstitucije varijabli, koju nazivamo "unifikator". Postupak unifikacije ključan je za zaključivanje u FOL. Neka α, β i θ označavaju supstitucije, a "$\circ$" kompoziciju dviju supstitucija. **Što mora vrijediti, a da je supstitucija θ najopćenitiji zajednički unifikator (MGU)?**

A $\forall\alpha\forall\beta\big((K_1\theta = K_2\theta) \to (\theta = \alpha \circ \beta)\big)$ C $\forall\alpha\big((K_1\alpha = K_2\alpha) \to \exists\beta(\alpha = \theta \circ \beta)\big)$

B $\forall\beta\exists\alpha\big((K_1\alpha = K_2\alpha) \to (\beta = \theta \circ \alpha)\big)$ D $\exists\alpha\big((K_1\alpha = K_2\alpha) \to \forall\beta(\theta = \alpha \circ \beta)\big)$

16 (R) Zadane su premise: *"Ivo voli sve vrste hrane. Jabuka i piletina su hrana. Branko jede lješnjake i nije mu zlo od njih. Vesna jede sve što i Branko. Ako netko nešto jede i nije mu od toga zlo, onda je to hrana."* Formalizirajte ove premise u FOL i pretvorite ih u klauzule. Pritom koristite $V(x, y)$ za "x voli y", $J(x, y)$ za "x jede y", "$Z(x, y)$" za "x-u je zlo od y", i "$H(x)$" "x je hrana". Zatim rezolucijom opovrgavanjem uz strategiju skupa potpore dokažite cilj *Ivo voli lješnjake.* **Koliko je minimalno koraka (primjena rezolucijskog pravila) potrebno za dokaz?**

A 4 B 5 C 6 D 3

4. Logičko programiranje i ekspertni sustavi (4 pitanja)

17 (T) Zaključivanje u ekspertnim sustavima može biti izvedeno kao ulančavanje unaprijed ili ulančavanje unazad. **U kojim je primjenama zaključivanje unazad bolji izbor od ulančavanja unaprijed?**

A Kada novoizvedeni zaključci mogu invalidirati ranije izvedene zaključke

B Kada je znanje moguće dekomponirati u granule znanja i formalizirati ih ako–onda pravilima

C Kada ako–onda pravila imaju različite prioritete koji se mogu mijenjati tijekom izvođenja programa

D Kada je broj mogućih hipoteza malen, a broj potpora (dokaza) za te hipoteze potencijalno velik

18 (R) Razmatramo sljedeći program u Prologu:

```
roditelj(kronos, zeus).
roditelj(zeus, minos).
roditelj(europa, minos).
potomak(X, Y) :- roditelj(Y, X).
potomak(X, Y) :- roditelj(Y, Z), potomak(X, Z).
bog(kronos).
bog(X) :- potomak(X, Y), bog(Y).
polubog(X) :- potomak(X, Y), not(bog(Y)).
```

Nad ovako definiranom bazom izvodimo upit `polubog(minos)`. Nacrtajte Prologovo stablo dokaza za ovaj upit do trenutka drugog povratka u pretraživanju. Pritom za dokazivanje negiranog cilja koristite dva čvora (jedan za `not(P(X))` i drugi za `P(X)`), dok prazni stog ne brojite kao čvor. **Koliko čvorova ima stablo dokaza neposredno prije drugog povratka u pretraživanju?**

A 8 B 12 C 6 D 10

19 (T) Prolog je deklarativan programski jezik, ali sadrži neke nedeklarativne aspekte. Jedan aspekt nedeklarativnosti u Prologu jest taj da u nekim slučajevima ne vrijedi komutativnost operatora "∧", npr., program `P(X,Y) :- Q(X), R(Y)` i program `P(X,Y) :- R(Y), Q(X)` ne evaluiraju se nužno identično. **Zbog čega u Prologu u ovom slučaju može postojati odstupanje proceduralnog značenja od deklarativnog?**

A Prolog ne provodi standardizaciju klauzula, pa različiti redoslijed klauzula može dati različite unifikatore

B Prolog koristi definitne klauzule, pa do odstupanja dolazi kada neku FOL formulu ne možemo izraziti u Prologu

C Prolog koristi rezoluciju SLD i pretraživanje u dubinu, pa dokazivanje ovisi o redoslijedu atoma na stogu

D Prolog pretpostavlja zatvorenost svijeta, no u stvarnosti činjenice koje nisu u bazi znanja mogu biti istinite

20 (R) Baza znanja ekspertnog sustava sadrži sljedeća pravila:

(1) AKO $(A = a_2) \wedge (D = d_2)$ ONDA $C = c_1$
(2) AKO $(F = f_3) \vee (B = b_3)$ ONDA $C = c_2$
(3) AKO $(E = e_1) \vee (B = b_1)$ ONDA $(A = a_1) \wedge (D = d_2)$
(4) AKO $F = f_1$ ONDA $D = d_2$
(5) AKO $F = f_2$ ONDA $E = e_2$
(6) AKO $(B = b_3) \vee (D = d_1)$ ONDA $(E = e_1) \wedge (A = a_2)$

Sustav koristimo za izvođenje vrijednosti varijable C ulančavanjem unazad. Prednost imaju pravila s manjim rednim brojem. Pravila koja su jednom palila više ne mogu paliti. Jednom izvedeni međuciljevi brišu se sa stoga, neovisno o njihovoj poziciji na stogu. Na možebitne upite od strane sustava, korisnik odgovara sa $B = b_3$ i $F = f_1$. **Što radi ekspertni sustav pri izvođenju vrijednosti varijable C?**

A Pali šest pravila i izvodi $C = C_2$

B Odbacuje pravilo 3 te kasnije pali pravilo 4

C Odbacuje pravilo 5 te kasnije pali pravilo 6

D Odbacuje pravilo 2 te kasnije pali pravilo 5

Međuispit iz Uvoda u umjetnu inteligenciju (2021./2022.)

– NEKORIGIRANA VERZIJA –

*Ispit se sastoji od **20 pitanja** i ukupno nosi **20 bodova**. Točan odgovor nosi 1 bod, a netočan -1/3 boda. Trajanje ispita je **150 minuta**. Primjerak ispita trebate predati zajedno sa svojim rješenjima.*

1. Pretraživanje prostora stanja i heurističko pretraživanje (7 pitanja)

1 (T) Prednost algoritma A^* nad algoritmom pretraživanja s jednolikom cijenom (UCS) jest ta da algoritam A^* koristi heuristiku, dok ju algoritam UCS ne koristi. **Kako se ta prednost manifestira u praksi?**

A Neovisno o svojstvima heuristike, A^* će ispitati jednak ili manji broj čvorova od onoga koji će ispitati UCS

B Ako rješenje postoji, A^* će u manje koraka pronaći rješenje manje ili jednake cijene od onoga koje nalazi UCS

C Ako je heuristika optimistična, A^* će generirati manji ili jednak broj čvorova od algoritma UCS

D Ako je heuristika optimistična, A^* će pronaći put koji je manje cijene od onoga koji pronalazi UCS

2 (P) Oblikujemo računalnu igru u kojoj se računalno upravljani igrači kreću po 2D mapi i skupljaju raznorazne artefakte. U jednom trenutku igrači trebaju u što manje poteza doći do zajedničkog zbornog mjesta. Igrači se mogu kretati u četiri smjera (gore, dolje, lijevo, desno), pri čemu rade po jedan korak. Mapa sadržava prepreke (zidove), kroz koje igrači ne mogu proći. Mapa također sadržava slučajan broj slučajno raspoređenih teleportera, koji igrača mogu trenutačno (u samo jednom koraku) transportirati na slučajno mjesto na mapi. No, jednom upotrijebljeni teleporter više se ne može upotrijebiti. Za ovu igru želimo oblikovati optimističnu heuristiku koja će navoditi računalno upravljane igrače do zbornog mjesta. Za oblikovanje heuristike koristimo metodu relaksacije. **Koja od sljedećih relaksacija igre daje optimističnu heuristiku?**

A Igrači se mogu kretati dijagonalno, ali ne mogu koristiti teleportere

B Igrači mogu prolaziti kroz zidove i dvaput koristiti isti teleporter

C Teleporteri se mogu koristiti neograničeno mnogo puta

D Teleporteri uvijek transportiraju igrača do teleportera koji je najbliži zbornom mjestu

3 (R) Algoritmom slijepog pretraživanja rješavamo problem Hanojskih tornjeva s tri klinova i tri diska. Klinovi su označeni sa A, B i C, a diskovi sa 1, 2 i 3, pri čemu broj odgovara veličini diska (disk 1 je najmanji, a disk 3 najveći). Stanje možemo označiti kao AxByCz, gdje x, y i z označavaju diskove koji se nalaze na klinovima A, B, odnosno C, krenuvši od dna prema vrhu (ako je klin prazan, ispuštamo njegovo slovo). Npr., u stanju B21C3 na klinu B su diskovi 2 (ispod) i 1 (gore), a na klinu C je disk 3. Početno stanje je A321. Zadatak je prebaciti diskove s klina A na klin C, tj. pronaći stanje C321, koristeći klin B kao pomoćni klin. Pritom se u svakom koraku prebacuje samo jedan disk, uz uvjet da veći disk ne smije biti položen na manji disk. Koristimo algoritam iterativnog pretraživanja u dubinu (IDS) bez liste posjećenih stanja. Algoritam čvorove proširuje leksikografskim poretkom gdje znamenke dolaze prije slova, npr., stanje A32B1 prethodi stanju A3B2C1. **Koliko će čvorova generirati algoritam IDS prije nego što generira čvor sa stanjem A3C21**?

A 29 B 25 C 22 D 16

4 (P) Stanje s slagalice 3×3 neka je $[[1,5,2],[4,\square,3],[7,8,6]]$, dok je ciljno stanje $[[1,2,3],[4,5,6],[7,8,\square]]$. Neka je h optimistična heuristika. **Među ponuđenim vrijednostima, koju vrijednost $h(s)$ može poprimiti, a da je najviše obaviještena?**

A 9 B 6 C 0 D 3

5 (T) Svi algoritmi slijepog pretraživanja u dubinu eksponencijalne su vremenske složenosti, međutim konkretni izrazi za asimptotsku složenost ipak se malo razlikuju. Asimptotska vremenska složenost algoritma pretraživanja

u širinu (BFS) je $\mathcal{O}(b^{d+1})$, a algoritma iterativnog pretraživanja u dubinu (IDS) $\mathcal{O}(b^d)$. **Zašto asimptotska vremenska složenost algoritma IDS nije ista kao i ona algoritma BFS, $\mathcal{O}(b^{d+1})$?**

A IDS generirane čvorove umeće na početak, a BFS na kraj liste, pa BFS na razini d generira b dodatnih čvorova

B IDS više puta proširuje iste čvorove, ali na posljednoj razini proširuje sve čvorove osim ciljnog

C IDS ima dubinsko ograničenje koje povećava do dubine rješenja d, pa nikada ne proširuje čvorove razine d

D IDS čvorove na razini d proširuje b^d puta, dok BFS dodatno proširuje svaki čvor na razini d osim ciljnog

6 (P) Razmatramo problem slagalice dimenzija 3×3 s prosječnom dubinom rješenja $d = 12$. Za nalaženje rješenja algoritam pretraživanja u dubinu u prosjeku koristi 610 B memorije. **Koliko bi memorije u prosjeku koristio algoritam pretraživanja u širinu na ovome problemu?**

A 2.4 MB B 203 MB C 202 KB D 3.7 MB

7 (R) Prostor stanja pretražujemo algoritmom A^*. Skup stanja je $S = \{a, b, c, d, e, f\}$, a funkcija sljedbenika je $succ(a) = \{(b, 2), (c, 2)\}$, $succ(b) = \{(c, 5), (d, 2)\}$, $succ(c) = \{(d, 1), (f, 20)\}$, $succ(d) = \{(e, 2)\}$, $succ(e) = \{(f, 14)\}$ te $succ(f) = \emptyset$. Heurističke vrijednosti stanja su $h(a) = 16$, $h(b) = 6$, $h(c) = 14$, $h(d) = 4$, $h(e) = 2$, $h(f) = 0$. Početno stanje je a, a ciljno f. Izvedite korake algoritma A^*, bilježeći u svakom koraku sadržaj liste otvorenih čvorova O i skupa zatvorenih čvorova C. U nultom koraku algoritma vrijedi $O = [(a, 0)]$ i $C = \emptyset$. **Koji je sadržaj listi O i skupa C nakon petog koraka izvođenja algoritma?**

A $O = [],\ C = \{(a, 0), (b, 2), (c, 2), (d, 3), (e, 5)\}$

B Algoritam ne dostiže peti korak

C $O = [(d, 3), (f, 20)],\ C = \{(a, 0), (b, 2), (c, 2), (e, 6)\}$

D $O = [(d, 3), (f, 20)],\ C = \{(a, 0), (b, 2), (c, 2)\}$

2. Igranje igara (3 pitanja)

8 (R) Razmatramo igru dvaju minimax-igrača. Stablo igre definirano je prijelazima $succ(A) = \{B, C, D\}$, $succ(B) = \{E, F\}$, $succ(C) = \{G, H\}$, $succ(D) = \{I, J\}$, $succ(G) = succ(F) = \{K, L\}$, $succ(H) = succ(E) = succ(I) = \{M, N\}$ te $succ(J) = \{O, P\}$. Igrač MAX koristi heuristiku h_1, a igrač MIN heuristiku h_2. Vrijednosti heuristika su sljedeće (obje heuristike definiraju dobit dotičnog igrača):

	E	F	G	H	I	J	K	L	M	N	O	P
h_1	1	−2	3	2	0	5	−1	0	−2	−1	0	2
h_2	0	5	2	4	−1	2	3	2	1	−1	1	−2

Oba igrača pretražuju najviše dva poteza unaprijed (jedan svoj i jedan suparnički). Igru započinje igrač MAX u stanju A. **Kroz koja stanja će se razvijati igra?**

A $A, C, H, \ldots$ B $A, C, G, \ldots$ C $A, B, F, \ldots$ D $A, B, E, \ldots$

9 (T) Igrač MAX primjenjuje algoritam minimax s heuristikom h na korijen stabla igre kako bi odredio svoj optimalan potez. Nakon što povuče taj potez, na redu je protivnik MIN, koji također vuče svoj optimalan potez. Igrač MAX zatim nanovo primjenjuje algoritam MAX na novom stanju igre. Pritom igrač MAX ponovo izračunava minimax-vrijednosti i za one čvorove za koje je ranije (za prethodni potez) već bio izračunao minimax-vrijednosti. **Zbog čega je potrebno ponovno izračunavati minimax-vrijednosti za te čvorove?**

A Heuristika h nije savršena, pa igrač MAX ne može očekivati da će ostvariti točno onu isplatu koju predviđa heuristika h

B Algoritam minimax radi pretraživanje u dubinu, pa ne pohranjuje čvorove na stazama izvan poteza koji je predvidio igrač MAX

C Ako je za prethodni izračun korištena heuristika, novi izračun bit će informiraniji jer će ići dvije razine dublje od prethodnog

D Protivnik MIN može koristi heuristiku manje točnosti od heuristike h, pa njegovi potezi mogu biti drugačiji od onih koje je predvidio igrač MAX

10 (P) Razmatramo dva igraća algoritma: jedan s heuristikom h_1 i dubinom pretraživanja d_1 te drugi s heuristikom h_2 i dubinom pretraživanja d_2. Oba algoritma koriste strategiju minimax. **Kada će prvi algoritam češće pobjeđivati drugi algoritam?**

A Ako $d_2 < d_1$ B Ako $h_1 = h_2$ i $d_1 < d_2$ C Ako $d_1 > d_2$ D Ako $h_1 = h_2$ i $d_1 > d_2$

3. Prikazivanje znanja i automatsko zaključivanje (6 pitanja)

11 (R) Zadane su premise: *"Ivo voli sve vrste hrane. Jabuka i piletina su hrana. Branko jede lješnjake i nije mu zlo od njih. Vesna jede sve što i Branko. Ako netko nešto jede i nije mu od toga zlo, onda je to hrana."* Formalizirajte ove premise u FOL i pretvorite ih u klauzule. Pritom koristite $V(x, y)$ za "x voli y", $J(x, y)$ za "x jede y", "$Z(x, y)$" za "x-u je zlo od y", i "$H(x)$" "x je hrana". Zatim rezolucijom opovrgavanjem uz strategiju skupa potpore dokažite cilj *Ivo voli lješnjake*. **Koliko je minimalno koraka (primjena rezolucijskog pravila) potrebno za dokaz?**

A 4 B 6 C 3 D 5

12 (P) Na postupak dokazivanja rezolucijom opovrgavanjem možemo gledati kao na problem pretraživanje prostora stanja, gdje stanja odgovaraju skupu klauzula (onih zadanih i onih izvedenih), a prijelazi između stanja odgovaraju primjeni rezolucijskog pravila na jedan par klauzula. Neka skup premisa sadrži 8 klauzula, a negirani cilj 2 klauzule. Za dokazivanje koristimo dvije strategije: strategiju zasićenja po razinama (LSS) i strategiju skupa potpore (SOS). Jednom razriješeni par klauzula više se ne razrješava. Izračunajte gornju ogradu na ukupan broj primjena rezolucijskog pravila za LSS i za SOS, ako znamo da je cilj dokaziv u tri koraka (tri primjene rezolucijskog pravila). **Koliko je puta gornja ograda za LSS veća od gornje ograde za SOS?**

A $7.51\times$ B $5.71\times$ C $9.65\times$ D $4.53\times$

13 (T) Logička posljedica središnji je pojam u logici i razlog zašto uopće koristimo sustav formalne logike za prikazivanje znanja. U propozicijskoj logici, logičku posljedicu moguće je dokazati na nekoliko načina. Jedan način temelji se na teoremu semantičke dedukcije. **Kako riječima glasi teorem semantičke dedukcije?**

A Ciljna formula logička je posljedica premisa ako i samo ako je implikacija s premisa na ciljnu formulu tautologija

B Ciljna formula je logička posljedica premisa ako i samo ako je ciljna formula istinita kada su istinite premise

C Premise i negacija cilja zajedno su zadovoljivi ako i samo ako je ciljna formula logička posljedica premisa

D Ako su premise nezadovoljive, bilo koja formula logička je posljedica takvih premisa

14 (T) Unifikacija je postupak svođenja izraza K_1 i K_2 iz FOL na identičan oblik primjenom odgovarajuće supstitucije varijabli, koju nazivamo "unifikator". Postupak unifikacije ključan je za zaključivanje u FOL. Neka α, β i θ označavaju supstitucije, a "$\circ$" kompoziciju dviju supstitucija. **Što mora vrijediti, a da je supstitucija θ najopćenitiji zajednički unifikator (MGU)?**

A $\forall\beta\exists\alpha\big((K_1\alpha = K_2\alpha) \to (\beta = \theta \circ \alpha)\big)$ C $\forall\alpha\big((K_1\alpha = K_2\alpha) \to \exists\beta(\alpha = \theta \circ \beta)\big)$

B $\exists\alpha\big((K_1\alpha = K_2\alpha) \to \forall\beta(\theta = \alpha \circ \beta)\big)$ D $\forall\alpha\forall\beta\big((K_1\theta = K_2\theta) \to (\theta = \alpha \circ \beta)\big)$

15 (R) Dani su FOL atomi $P(f(x), y, g(w, b))$ i $P(y, f(g(a, z)), x)$, gdje je P predikat, a i b su konstante, a f i g su funkcije. **Što je najopćenitiji zajednički unifikator ovih dvaju atoma?**

A Atome nije moguće unificirati C $P(f(g(a, b)), f(g(a, b)), g(a, b))$

B $P(f(y), f(g(a, z)), g(w, b))$ D $P(f(x), f(g(x, y)), g(a, b))$

16 (P) Ispravnost je važno svojstvo pravila zaključivanja. **Koje je od sljedećih pravila zaključivanja ispravno?**

A $A \to (B \to C) \vdash (A \vee B) \to C$ C $A \to B, B \to C \vdash C \to A$

B $A \vdash A \wedge B$ D $A \to (B \to C) \vdash (A \wedge B) \to C$

4. Logičko programiranje i ekspertni sustavi (4 pitanja)

17 (T) Zaključivanje u ekspertnim sustavima može biti izvedeno kao ulančavanje unaprijed ili ulančavanje unazad. **U kojim je primjenama zaključivanje unazad bolji izbor od ulančavanja unaprijed?**

A Kada je broj mogućih hipoteza malen, a broj potpora (dokaza) za te hipoteze potencijalno velik

B Kada je znanje moguće dekomponirati u granule znanja i formalizirati ih ako–onda pravilima

C Kada novoizvedeni zaključci mogu invalidirati ranije izvedene zaključke

D Kada ako–onda pravila imaju različite prioritete koji se mogu mijenjati tijekom izvođenja programa

18 (R) Razmatramo sljedeći program u Prologu:

```
roditelj(kronos, zeus).
roditelj(zeus, minos).
roditelj(europa, minos).
potomak(X, Y) :- roditelj(Y, X).
potomak(X, Y) :- roditelj(Y, Z), potomak(X, Z).
bog(kronos).
bog(X) :- potomak(X, Y), bog(Y).
polubog(X) :- potomak(X, Y), not(bog(Y)).
```

Nad ovako definiranom bazom izvodimo upit `polubog(minos)`. Nacrtajte Prologovo stablo dokaza za ovaj upit do trenutka drugog povratka u pretraživanju. Pritom za dokazivanje negiranog cilja koristite dva čvora (jedan za `not(P(X))` i drugi za `P(X)`), dok prazni stog ne brojite kao čvor. **Koliko čvorova ima stablo dokaza neposredno prije drugog povratka u pretraživanju?**

A 10 B 12 C 8 D 6

19 (R) Baza znanja ekspertnog sustava sadrži sljedeća pravila:

(1) AKO $(A = a_2) \wedge (D = d_2)$ ONDA $C = c_1$
(2) AKO $(F = f_3) \vee (B = b_3)$ ONDA $C = c_2$
(3) AKO $(E = e_1) \vee (B = b_1)$ ONDA $(A = a_1) \wedge (D = d_2)$
(4) AKO $F = f_1$ ONDA $D = d_2$
(5) AKO $F = f_2$ ONDA $E = e_2$
(6) AKO $(B = b_3) \vee (D = d_1)$ ONDA $(E = e_1) \wedge (A = a_2)$

Sustav koristimo za izvođenje vrijednosti varijable C ulančavanjem unazad. Prednost imaju pravila s manjim rednim brojem. Pravila koja su jednom palila više ne mogu paliti. Jednom izvedeni međuciljevi brišu se sa stoga, neovisno o njihovoj poziciji na stogu. Na možebitne upite od strane sustava, korisnik odgovara sa $B = b_3$ i $F = f_1$. **Što radi ekspertni sustav pri izvođenju vrijednosti varijable C?**

A Završava sa 4 činjenica u radnoj memoriji

B Odbacuje pravilo 5 te kasnije pali pravilo 6

C Odbacuje pravilo 6 te kasnije pali pravilo 3

D Odbacuje pravilo 2 te kasnije pali pravilo 5

20 (T) Prolog je deklarativan programski jezik, ali sadrži neke nedeklarativne aspekte. Jedan aspekt nedeklarativnosti u Prologu jest taj da u nekim slučajevima ne vrijedi komutativnost operatora "$\wedge$", npr., program `P(X,Y) :- Q(X), R(Y)` i program `P(X,Y) :- R(Y), Q(X)` ne evaluiraju se nužno identično. **Zbog čega u Prologu u ovom slučaju može postojati odstupanje proceduralnog značenja od deklarativnog?**

A Prolog ne provodi standardizaciju klauzula, pa različiti redoslijed klauzula može dati različite unifikatore

B Prolog pretpostavlja zatvorenost svijeta, no u stvarnosti činjenice koje nisu u bazi znanja mogu biti istinite

C Prolog koristi definitne klauzule, pa do odstupanja dolazi kada neku FOL formulu ne možemo izraziti u Prologu

D Prolog koristi rezoluciju SLD i pretraživanje u dubinu, pa dokazivanje ovisi o redoslijedu atoma na stogu

Međuispit iz Uvoda u umjetnu inteligenciju (2021./2022.)

– NEKORIGIRANA VERZIJA –

*Ispit se sastoji od **20 pitanja** i ukupno nosi **20 bodova**. Točan odgovor nosi 1 bod, a netočan -1/3 boda. Trajanje ispita je **150 minuta**. Primjerak ispita trebate predati zajedno sa svojim rješenjima.*

1. Pretraživanje prostora stanja i heurističko pretraživanje (7 pitanja)

1 (R) Prostor stanja pretražujemo algoritmom A^*. Skup stanja je $S = \{a, b, c, d, e, f\}$, a funkcija sljedbenika je $succ(a) = \{(b, 2), (c, 2)\}$, $succ(b) = \{(c, 5), (d, 2)\}$, $succ(c) = \{(d, 1), (f, 20)\}$, $succ(d) = \{(e, 2)\}$, $succ(e) = \{(f, 14)\}$ te $succ(f) = \emptyset$. Heurističke vrijednosti stanja su $h(a) = 16$, $h(b) = 6$, $h(c) = 14$, $h(d) = 4$, $h(e) = 2$, $h(f) = 0$. Početno stanje je a, a ciljno f. Izvedite korake algoritma A^*, bilježeći u svakom koraku sadržaj liste otvorenih čvorova O i skupa zatvorenih čvorova C. U nultom koraku algoritma vrijedi $O = [(a, 0)]$ i $C = \emptyset$. **Koji je sadržaj liste O i skupa C nakon šestog koraka izvođenja algoritma?**

A Algoritam ne dostiže šesti korak

B $O = [(e, 2), (f, 0)]$, $C = \{(a, 16), (b, 6), (c, 14), (d, 3)\}$

C $O = [(e, 5), (f, 20)]$, $C = \{(a, 0), (b, 2), (c, 7)\}$

D $O = [(e, 5), (f, 20)]$, $C = \{(a, 0), (b, 2), (c, 2), (d, 3)\}$

2 (P) Razmatramo problem slagalice dimenzija 3×3 s prosječnom dubinom rješenja $d = 11$. Za nalaženje rješenja algoritam pretraživanja u dubinu u prosjeku koristi 720 B memorije. **Koliko bi memorije u prosjeku koristio algoritam pretraživanja u širinu na ovome problemu?**

A 65 MB B 720 MB C 76 KB D 1.1 MB

3 (P) Oblikujemo računalnu igru u kojoj se računalno upravljani igrači kreću po 2D mapi i skupljaju raznorazne artefakte. U jednom trenutku igrači trebaju u što manje poteza doći do zajedničkog zbornog mjesta. Igrači se mogu kretati u četiri smjera (gore, dolje, lijevo, desno), pri čemu rade po jedan korak. Mapa sadržava prepreke (zidove), kroz koje igrači ne mogu proći. Mapa također sadržava slučajan broj slučajno raspoređenih teleportera, koji igrača mogu trenutačno (u samo jednom koraku) transportirati na slučajno mjesto na mapi. No, jednom upotrijebljeni teleporter više se ne može upotrijebiti. Za ovu igru želimo oblikovati optimističnu heuristiku koja će navoditi računalno upravljane igrače do zbornog mjesta. Za oblikovanje heuristike koristimo metodu relaksacije. **Koja od sljedećih relaksacija igre daje optimističnu heuristiku?**

A Teleporteri uvijek transportiraju igrača do teleportera koji je najbliži zbornom mjestu

B Teleporteri se mogu koristiti neograničeno mnogo puta

C Igrači mogu prolaziti kroz zidove i dvaput koristiti isti teleporter

D Igrači se mogu kretati dijagonalno, ali ne mogu koristiti teleportere

4 (T) Svi algoritmi slijepog pretraživanja u dubinu eksponencijalne su vremenske složenosti, međutim konkretni izrazi za asimptotsku složenost ipak se malo razlikuju. Asimptotska vremenska složenost algoritma pretraživanja u širinu (BFS) je $\mathcal{O}(b^{d+1})$, a algoritma iterativnog pretraživanja u dubinu (IDS) $\mathcal{O}(b^d)$. **Zašto asimptotska vremenska složenost algoritma IDS nije ista kao i ona algoritma BFS, $\mathcal{O}(b^{d+1})$?**

A IDS više puta proširuje iste čvorove, ali na posljednoj razini proširuje sve čvorove osim ciljnog

B IDS generirane čvorove umeće na početak, a BFS na kraj liste, pa BFS na razini d generira b dodatnih čvorova

C IDS ima dubinsko ograničenje koje povećava do dubine rješenja d, pa nikada ne proširuje čvorove razine d

D IDS čvorove na razini d proširuje b^d puta, dok BFS dodatno proširuje svaki čvor na razini d osim ciljnog

5 (P) Stanje s slagalice 3×3 neka je $[[1, 2, \square], [4, 5, 3], [7, 8, 6]]$, dok je ciljno stanje $[[1, 2, 3], [4, 5, 6], [7, 8, \square]]$. Neka je h optimistična heuristika. **Među ponuđenim vrijednostima, koju vrijednost $h(s)$ može poprimiti, a da je najviše obaviještena?**

A 8 B 1 C 4 D 3

6 (T) Prednost algoritma A^* nad algoritmom pretraživanja s jednolikom cijenom (UCS) jest ta da algoritam A^* koristi heuristiku, dok ju algoritam UCS ne koristi. **Kako se ta prednost manifestira u praksi?**

A Ako je heuristika optimistična, A^* će generirati manji ili jednak broj čvorova od algoritma UCS

B Ako rješenje postoji, A^* će u manje koraka pronaći rješenje manje ili jednake cijene od onoga koje nalazi UCS

C Ako je heuristika optimistična, A^* će pronaći put koji je manje cijene od onoga koji pronalazi UCS

D Neovisno o svojstvima heuristike, A^* će ispitati jednak ili manji broj čvorova od onoga koji će ispitati UCS

7 (R) Algoritmom slijepog pretraživanja rješavamo problem Hanojskih tornjeva s tri klinova i tri diska. Klinovi su označeni sa A, B i C, a diskovi sa 1, 2 i 3, pri čemu broj odgovara veličini diska (disk 1 je najmanji, a disk 3 najveći). Stanje možemo označiti kao AxByCz, gdje x, y i z označavaju diskove koji se nalaze na klinovima A, B, odnosno C, krenuvši od dna prema vrhu (ako je klin prazan, ispuštamo njegovo slovo). Npr., u stanju B21C3 na klinu B su diskovi 2 (ispod) i 1 (gore), a na klinu C je disk 3. Početno stanje je A321. Zadatak je prebaciti diskove s klina A na klin C, tj. pronaći stanje C321, koristeći klin B kao pomoćni klin. Pritom se u svakom koraku prebacuje samo jedan disk, uz uvjet da veći disk ne smije biti položen na manji disk. Koristimo algoritam iterativnog pretraživanja u dubinu (IDS) bez liste posjećenih stanja. Algoritam čvorove proširuje leksikografskim poretkom gdje znamenke dolaze prije slova, npr., stanje A32B1 prethodi stanju A3B2C1. **Koliko će čvorova generirati algoritam IDS prije nego što generira čvor sa stanjem A3C21**?

A 22 B 25 C 16 D 29

2. Igranje igara (3 pitanja)

8 (P) Razmatramo dva igraća algoritma: jedan s heuristikom h_1 i dubinom pretraživanja d_1 te drugi s heuristikom h_2 i dubinom pretraživanja d_2. Oba algoritma koriste strategiju minimax. **Kada će prvi algoritam češće pobjeđivati drugi algoritam?**

A Ako $h_1 = h_2$ i $d_1 < d_2$ B Ako $h_1 = h_2$ i $d_1 > d_2$ C Ako $h_1 > h_2$ D Ako $h_2 > h_1$

9 (R) Razmatramo igru dvaju minimax-igrača. Stablo igre definirano je prijelazima $succ(A) = \{B, C, D\}$, $succ(B) = \{E, F\}$, $succ(C) = \{G, H\}$, $succ(D) = \{I, J\}$, $succ(G) = succ(F) = \{K, L\}$, $succ(H) = succ(E) = succ(I) = \{M, N\}$ te $succ(J) = \{O, P\}$. Igrač MAX koristi heuristiku h_1, a igrač MIN heuristiku h_2. Vrijednosti heuristika su sljedeće (obje heuristike definiraju dobit dotičnog igrača):

	E	F	G	H	I	J	K	L	M	N	O	P
h_1	1	−2	3	2	0	5	−1	0	−2	−1	0	2
h_2	0	5	2	4	−1	2	3	2	1	−1	1	−2

Oba igrača pretražuju najviše dva poteza unaprijed (jedan svoj i jedan suparnički). Igru započinje igrač MAX u stanju A. **Kroz koja stanja će se razvijati igra?**

A $A, C, G, \ldots$ B $A, B, E, \ldots$ C $A, C, H, \ldots$ D $A, D, J, \ldots$

10 (T) Igrač MAX primjenjuje algoritam minimax s heuristikom h na korijen stabla igre kako bi odredio svoj optimalan potez. Nakon što povuče taj potez, na redu je protivnik MIN, koji također vuče svoj optimalan potez. Igrač MAX zatim nanovo primjenjuje algoritam MAX na novom stanju igre. Pritom igrač MAX ponovo izračunava minimax-vrijednosti i za one čvorove za koje je ranije (za prethodni potez) već bio izračunao minimax-vrijednosti. **Zbog**

čega je potrebno ponovno izračunavati minimax-vrijednosti za te čvorove?

A Algoritam minimax radi pretraživanje u dubinu, pa ne pohranjuje čvorove na stazama izvan poteza koji je predvidio igrač MAX

B Protivnik MIN može koristi heuristiku manje točnosti od heuristike h, pa njegovi potezi mogu biti drugačiji od onih koje je predvidio igrač MAX

C Ako je za prethodni izračun korištena heuristika, novi izračun bit će informiraniji jer će ići dvije razine dublje od prethodnog

D Heuristika h nije savršena, pa igrač MAX ne može očekivati da će ostvariti točno onu isplatu koju predviđa heuristika h

3. Prikazivanje znanja i automatsko zaključivanje (6 pitanja)

11 (R) Zadane su premise: *"Ivo voli sve vrste hrane. Jabuka i piletina su hrana. Branko jede lješnjake i nije mu zlo od njih. Vesna jede sve što i Branko. Ako netko nešto jede i nije mu od toga zlo, onda je to hrana."* Formalizirajte ove premise u FOL i pretvorite ih u klauzule. Pritom koristite $V(x, y)$ za "x voli y", $J(x, y)$ za "x jede y", "$Z(x, y)$" za "x-u je zlo od y", i "$H(x)$" "x je hrana". Zatim rezolucijom opovrgavanjem uz strategiju skupa potpore dokažite cilj *Ivo voli lješnjake*. **Koliko je minimalno koraka (primjena rezolucijskog pravila) potrebno za dokaz?**

A 5 B 4 C 3 D 6

12 (P) Na postupak dokazivanja rezolucijom opovrgavanjem možemo gledati kao na problem pretraživanje prostora stanja, gdje stanja odgovaraju skupu klauzula (onih zadanih i onih izvedenih), a prijelazi između stanja odgovaraju primjeni rezolucijskog pravila na jedan par klauzula. Neka skup premisa sadrži 7 klauzula, a negirani cilj 2 klauzule. Za dokazivanje koristimo dvije strategije: strategiju zasićenja po razinama (LSS) i strategiju skupa potpore (SOS). Jednom razriješeni par klauzula više se ne razrješava. Izračunajte gornju ograda na ukupan broj primjena rezolucijskog pravila za LSS i za SOS, ako znamo da je cilj dokaziv u tri koraka (tri primjene rezolucijskog pravila). **Koliko je puta gornja ograda za LSS veća od gornje ograde za SOS?**

A $4.53\times$ B $7.51\times$ C $5.71\times$ D $9.65\times$

13 (R) Dani su FOL atomi $P(f(x), y, g(w, f(z)))$ i $P(y, f(g(a, z)), x)$, gdje je P predikat, a je konstanta, a f i g su funkcije. **Što je najopćenitji zajednički unifikator ovih dvaju atoma?**

A $P(f(g(a,b)), f(g(a,b)), g(a,b))$
B $P(f(f(g(a,b))), f(g(a,b)), f(g(a,b)))$
C $P(f(g(a,b)), f(g(a,z)), g(w,b))$
D Atome nije moguće unificirati

14 (P) Ispravnost je važno svojstvo pravila zaključivanja. **Koje je od sljedećih pravila zaključivanja ispravno?**

A $A \to (B \to C) \vdash (A \vee B) \to C$
B $B, A \to B \vdash A$
C $A \to (B \vee C), \neg C \vdash A \to B$
D $A \to B, B \to C \vdash C \to A$

15 (T) Unifikacija je postupak svođenja izraza K_1 i K_2 iz FOL na identičan oblik primjenom odgovarajuće supstitucije varijabli, koju nazivamo "unifikator". Postupak unifikacije ključan je za zaključivanje u FOL. Neka α, β i θ označavaju supstitucije, a "$\circ$" kompoziciju dviju supstitucija. **Što mora vrijediti, a da je supstitucija θ najopćenitiji zajednički unifikator (MGU)?**

A $\forall\alpha\forall\beta\big((K_1\theta = K_2\theta) \to (\theta = \alpha \circ \beta)\big)$
B $\forall\beta\exists\alpha\big((K_1\alpha = K_2\alpha) \to (\beta = \theta \circ \alpha)\big)$
C $\exists\alpha\big((K_1\alpha = K_2\alpha) \to \forall\beta(\theta = \alpha \circ \beta)\big)$
D $\forall\alpha\big((K_1\alpha = K_2\alpha) \to \exists\beta(\alpha = \theta \circ \beta)\big)$

16 (T) Logička posljedica središnji je pojam u logici i razlog zašto uopće koristimo sustav formalne logike za prikazivanje znanja. U propozicijskoj logici, logičku posljedicu moguće je dokazati na nekoliko načina. Jedan način temelji se na teoremu semantičke dedukcije. **Kako riječima glasi teorem semantičke dedukcije?**

A Ciljna formula logička je posljedica premisa ako i samo ako je implikacija s premisa na ciljnu formulu tautologija

B Ako su premise nezadovoljive, bilo koja formula logička je posljedica takvih premisa

C Premise i negacija cilja zajedno su zadovoljivi ako i samo ako je ciljna formula logička posljedica premisa

D Ciljna formula je logička posljedica premisa ako i samo ako je ciljna formula istinita kada su istinite premise

4. Logičko programiranje i ekspertni sustavi (4 pitanja)

17 (R) Baza znanja ekspertnog sustava sadrži sljedeća pravila:

(1) AKO $(A = a_2) \wedge (D = d_2)$ ONDA $C = c_1$
(2) AKO $(F = f_3) \vee (B = b_3)$ ONDA $C = c_2$
(3) AKO $(E = e_1) \vee (B = b_1)$ ONDA $(A = a_1) \wedge (D = d_2)$
(4) AKO $F = f_1$ ONDA $D = d_2$
(5) AKO $F = f_2$ ONDA $E = e_2$
(6) AKO $(B = b_3) \vee (D = d_1)$ ONDA $(E = e_1) \wedge (A = a_2)$

Sustav koristimo za izvođenje vrijednosti varijable C ulančavanjem unazad. Prednost imaju pravila s manjim rednim brojem. Pravila koja su jednom palila više ne mogu paliti. Jednom izvedeni međuciljevi brišu se sa stoga, neovisno o njihovoj poziciji na stogu. Na možebitne upite od strane sustava, korisnik odgovara sa $B = b_3$ i $F = f_1$. **Što radi ekspertni sustav pri izvođenju vrijednosti varijable C?**

A Pali pet pravila i izvodi $C = c_1$

B Završava sa 3 činjenice u radnoj memoriji

C Odbacuje pravilo 5 te kasnije pali pravilo 6

D Odbacuje pravilo 2 te kasnije pali pravilo 5

18 (T) Zaključivanje u ekspertnim sustavima može biti izvedeno kao ulančavanje unaprijed ili ulančavanje unazad. **U kojim je primjenama zaključivanje unazad bolji izbor od ulančavanja unaprijed?**

A Kada je broj mogućih hipoteza malen, a broj potpora (dokaza) za te hipoteze potencijalno velik

B Kada novoizvedeni zaključci mogu invalidirati ranije izvedene zaključke

C Kada je znanje moguće dekomponirati u granule znanja i formalizirati ih ako–onda pravilima

D Kada ako–onda pravila imaju različite prioritete koji se mogu mijenjati tijekom izvođenja programa

19 (R) Razmatramo sljedeći program u Prologu:

```
roditelj(kronos, zeus).
roditelj(zeus, minos).
roditelj(europa, minos).
potomak(X, Y) :- roditelj(Y, X).
potomak(X, Y) :- roditelj(Y, Z), potomak(X, Z).
bog(kronos).
bog(X) :- potomak(X, Y), bog(Y).
polubog(X) :- potomak(X, Y), not(bog(Y)).
```

Nad ovako definiranom bazom izvodimo upit `polubog(minos)`. Nacrtajte Prologovo stablo dokaza za ovaj upit do trenutka drugog povratka u pretraživanju. Pritom za dokazivanje negiranog cilja koristite dva čvora (jedan za `not(P(X))` i drugi za `P(X)`)), dok prazni stog ne brojite kao čvor. **Koliko čvorova ima stablo dokaza neposredno prije drugog povratka u pretraživanju?**

A 8 B 6 C 12 D 10

20 (T) Prolog je deklarativan programski jezik, ali sadrži neke nedeklarativne aspekte. Jedan aspekt nedeklarativnosti u Prologu jest taj da u nekim slučajevima ne vrijedi komutativnost operatora "$\wedge$", npr., program `P(X,Y) :- Q(X), R(Y)` i program `P(X,Y) :- R(Y), Q(X)` ne evaluiraju se nužno identično. **Zbog čega u Prologu u ovom slučaju može postojati odstupanje proceduralnog značenja od deklarativnog?**

A Prolog koristi definitne klauzule, pa do odstupanja dolazi kada neku FOL formulu ne možemo izraziti u Prologu

B Prolog pretpostavlja zatvorenost svijeta, no u stvarnosti činjenice koje nisu u bazi znanja mogu biti istinite

C Prolog ne provodi standardizaciju klauzula, pa različiti redoslijed klauzula može dati različite unifikatore

D Prolog koristi rezoluciju SLD i pretraživanje u dubinu, pa dokazivanje ovisi o redoslijedu atoma na stogu

Međuispit iz Uvoda u umjetnu inteligenciju (2021./2022.)

– NEKORIGIRANA VERZIJA –

*Ispit se sastoji od **20 pitanja** i ukupno nosi **20 bodova**. Točan odgovor nosi 1 bod, a netočan -1/3 boda. Trajanje ispita je **150 minuta**. Primjerak ispita trebate predati zajedno sa svojim rješenjima.*

1. Pretraživanje prostora stanja i heurističko pretraživanje (7 pitanja)

1 (R) Algoritmom slijepog pretraživanja rješavamo problem Hanojskih tornjeva s tri klinova i tri diska. Klinovi su označeni sa A, B i C, a diskovi sa 1, 2 i 3, pri čemu broj odgovara veličini diska (disk 1 je najmanji, a disk 3 najveći). Stanje možemo označiti kao AxByCz, gdje x, y i z označavaju diskove koji se nalaze na klinovima A, B, odnosno C, krenuvši od dna prema vrhu (ako je klin prazan, ispuštamo njegovo slovo). Npr., u stanju B21C3 na klinu B su diskovi 2 (ispod) i 1 (gore), a na klinu C je disk 3. Početno stanje je A321. Zadatak je prebaciti diskove s klina A na klin C, tj. pronaći stanje C321, koristeći klin B kao pomoćni klin. Pritom se u svakom koraku prebacuje samo jedan disk, uz uvjet da veći disk ne smije biti položen na manji disk. Koristimo algoritam iterativnog pretraživanja u dubinu (IDS) bez liste posjećenih stanja. Algoritam čvorove proširuje leksikografskim poretkom gdje znamenke dolaze prije slova, npr., stanje A32B1 prethodi stanju A3B2C1. **Koliko će čvorova generirati algoritam IDS prije nego što generira čvor sa stanjem A3C21**?

A 22 B 25 C 16 D 29

2 (P) Oblikujemo računalnu igru u kojoj se računalno upravljani igrači kreću po 2D mapi i skupljaju raznorazne artefakte. U jednom trenutku igrači trebaju u što manje poteza doći do zajedničkog zbornog mjesta. Igrači se mogu kretati u četiri smjera (gore, dolje, lijevo, desno), pri čemu rade po jedan korak. Mapa sadržava prepreke (zidove), kroz koje igrači ne mogu proći. Mapa također sadržava slučajan broj slučajno raspoređenih teleportera, koji igrača mogu trenutačno (u samo jednom koraku) transportirati na slučajno mjesto na mapi. No, jednom upotrijebljeni teleporter više se ne može upotrijebiti. Za ovu igru želimo oblikovati optimističnu heuristiku koja će navoditi računalno upravljane igrače do zbornog mjesta. Za oblikovanje heuristike koristimo metodu relaksacije. **Koja od sljedećih relaksacija igre daje optimističnu heuristiku?**

A Igrači se mogu kretati dijagonalno, ali ne mogu koristiti teleportere

B Teleporteri se mogu koristiti neograničeno mnogo puta

C Teleporteri uvijek transportiraju igrača do teleportera koji je najbliži zbornom mjestu

D Igrači mogu prolaziti kroz zidove i dvaput koristiti isti teleporter

3 (T) Svi algoritmi slijepog pretraživanja u dubinu eksponencijalne su vremenske složenosti, međutim konkretni izrazi za asimptotsku složenost ipak se malo razlikuju. Asimptotska vremenska složenost algoritma pretraživanja u širinu (BFS) je $\mathcal{O}(b^{d+1})$, a algoritma iterativnog pretraživanja u dubinu (IDS) $\mathcal{O}(b^d)$. **Zašto asimptotska vremenska složenost algoritma IDS nije ista kao i ona algoritma BFS, $\mathcal{O}(b^{d+1})$?**

A IDS više puta proširuje iste čvorove, ali na posljednoj razini proširuje sve čvorove osim ciljnog

B IDS čvorove na razini d proširuje b^d puta, dok BFS dodatno proširuje svaki čvor na razini d osim ciljnog

C IDS ima dubinsko ograničenje koje povećava do dubine rješenja d, pa nikada ne proširuje čvorove razine d

D IDS generirane čvorove umeće na početak, a BFS na kraj liste, pa BFS na razini d generira b dodatnih čvorova

4 (P) Stanje s slagalice 3×3 neka je $[[1, \square, 2], [4, 5, 3], [7, 8, 6]]$, dok je ciljno stanje $[[1, 2, 3], [4, 5, 6], [7, 8, \square]]$. Neka je h optimistična heuristika. **Među ponuđenim vrijednostima, koju vrijednost $h(s)$ može poprimiti, a da je najviše obaviještena?**

A 6 B 8 C 0 D 2

5 (R) Prostor stanja pretražujemo algoritmom A^*. Skup stanja je $S = \{a, b, c, d, e, f\}$, a funkcija sljedbenika je $succ(a) = \{(b, 2), (c, 2)\}$, $succ(b) = \{(c, 5), (d, 2)\}$, $succ(c) = \{(d, 1), (f, 20)\}$, $succ(d) = \{(e, 2)\}$, $succ(e) = \{(f, 14)\}$ te $succ(f) = \emptyset$. Heurističke vrijednosti stanja su $h(a) = 16$, $h(b) = 6$, $h(c) = 14$, $h(d) = 4$, $h(e) = 2$, $h(f) = 0$. Početno stanje je a, a ciljno f. Izvedite korake algoritma A^*, bilježeći u svakom koraku sadržaj liste otvorenih čvorova O i skupa zatvorenih čvorova C. U nultom koraku algoritma vrijedi $O = [(a, 0)]$ i $C = \emptyset$. **Koji je sadržaj liste O i skupa C nakon petog koraka izvođenja algoritma?**

A $O = [(d, 3), (f, 20)]$, $C = \{(a, 0), (b, 2), (c, 2), (e, 6)\}$

B Algoritam ne dostiže peti korak

C $O = [(d, 3)]$, $C = \{(a, 0), (b, 2), (c, 2), (e, 6)\}$

D $O = []$, $C = \{(a, 0), (b, 2), (c, 2), (d, 3), (e, 5)\}$

6 (T) Prednost algoritma A^* nad algoritmom pretraživanja s jednolikom cijenom (UCS) jest ta da algoritam A^* koristi heuristiku, dok ju algoritam UCS ne koristi. **Kako se ta prednost manifestira u praksi?**

A Ako je heuristika optimistična, A^* će generirati manji ili jednak broj čvorova od algoritma UCS

B Neovisno o svojstvima heuristike, A^* će ispitati jednak ili manji broj čvorova od onoga koji će ispitati UCS

C Ako je heuristika optimistična, A^* će pronaći put koji je manje cijene od onoga koji pronalazi UCS

D Ako rješenje postoji, A^* će u manje koraka pronaći rješenje manje ili jednake cijene od onoga koje nalazi UCS

7 (P) Razmatramo problem slagalice dimenzija 3×4 s prosječnom dubinom rješenja $d = 12$. Za nalaženje rješenja algoritam pretraživanja u dubinu u prosjeku koristi 720 B memorije. **Koliko bi memorije u prosjeku koristio algoritam pretraživanja u širinu na ovome problemu?**

A 5.4 MB B 404 KB C 240 MB D 202 KB

2. Igranje igara (3 pitanja)

8 (R) Razmatramo igru dvaju minimax-igrača. Stablo igre definirano je prijelazima $succ(A) = \{B, C, D\}$, $succ(B) = \{E, F\}$, $succ(C) = \{G, H\}$, $succ(D) = \{I, J\}$, $succ(G) = succ(F) = \{K, L\}$, $succ(H) = succ(E) = succ(I) = \{M, N\}$ te $succ(J) = \{O, P\}$. Igrač MAX koristi heuristiku h_1, a igrač MIN heuristiku h_2. Vrijednosti heuristika su sljedeće (obje heuristike definiraju dobit dotičnog igrača):

	E	F	G	H	I	J	K	L	M	N	O	P
h_1	1	−2	3	2	0	5	−1	0	−2	−1	0	2
h_2	0	5	2	4	−1	2	3	2	1	−1	1	−2

Oba igrača pretražuju najviše dva poteza unaprijed (jedan svoj i jedan suparnički). Igru započinje igrač MAX u stanju A. **Kroz koja stanja će se razvijati igra?**

A $A, D, I, \ldots$ B $A, C, H, \ldots$ C $A, C, G, \ldots$ D $A, B, E, \ldots$

9 (T) Igrač MAX primjenjuje algoritam minimax s heuristikom h na korijen stabla igre kako bi odredio svoj optimalan potez. Nakon što povuče taj potez, na redu je protivnik MIN, koji također vuče svoj optimalan potez. Igrač MAX zatim nanovo primjenjuje algoritam MAX na novom stanju igre. Pritom igrač MAX ponovo izračunava minimax-vrijednosti i za one čvorove za koje je ranije (za prethodni potez) već bio izračunao minimax-vrijednosti. **Zbog čega je potrebno ponovno izračunavati minimax-vrijednosti za te čvorove?**

A Protivnik MIN može koristi heuristiku manje točnosti od heuristike h, pa njegovi potezi mogu biti drugačiji od onih koje je predvidio igrač MAX

B Ako je za prethodni izračun korištena heuristika, novi izračun bit će informiraniji jer će ići dvije razine dublje od prethodnog

C Heuristika h nije savršena, pa igrač MAX ne može očekivati da će ostvariti točno onu isplatu koju predviđa heuristika h

D Algoritam minimax radi pretraživanje u dubinu, pa ne pohranjuje čvorove na stazama izvan poteza koji je predvidio igrač MAX

10 (P) Razmatramo dva igraća algoritma: jedan s heuristikom h_1 i dubinom pretraživanja d_1 te drugi s heuristikom h_2 i dubinom pretraživanja d_2. Oba algoritma koriste strategiju minimax. **Kada će prvi algoritam češće pobjeđivati drugi algoritam?**

A Ako $d_1 = d_2$ i $h_1 > h_2$ B Ako $h_1 = h_2$ i $d_1 > d_2$ C Ako $d_2 < d_1$ D Ako $h_1 = h_2$ i $d_1 < d_2$

3. Prikazivanje znanja i automatsko zaključivanje (6 pitanja)

11 (R) Zadane su premise: *"Ivo voli sve vrste hrane. Jabuka i piletina su hrana. Branko jede lješnjake i nije mu zlo od njih. Vesna jede sve što i Branko. Ako netko nešto jede i nije mu od toga zlo, onda je to hrana."* Formalizirajte ove premise u FOL i pretvorite ih u klauzule. Pritom koristite $V(x, y)$ za "x voli y", $J(x, y)$ za "x jede y", "$Z(x, y)$" za "x-u je zlo od y", i "$H(x)$" "x je hrana". Zatim rezolucijom opovrgavanjem uz strategiju skupa potpore dokažite cilj *Ivo voli lješnjake*. **Koliko je minimalno koraka (primjena rezolucijskog pravila) potrebno za dokaz?**

A 3 B 4 C 5 D 6

12 (R) Dani su FOL atomi $P(f(x), y, g(w, f(z)))$ i $P(y, f(g(a, z)), x)$, gdje je P predikat, a je konstanta, a f i g su funkcije. **Što je najopćenitji zajednički unifikator ovih dvaju atoma?**

A Atome nije moguće unificirati
B $P(f(g(a,b)), f(g(a,b)), g(a,b))$
C $P(f(g(a,b)), f(g(a,z)), g(w,b))$
D $P(f(f(g(a,b))), f(g(a,b)), f(g(a,b)))$

13 (T) Logička posljedica središnji je pojam u logici i razlog zašto uopće koristimo sustav formalne logike za prikazivanje znanja. U propozicijskoj logici, logičku posljedicu moguće je dokazati na nekoliko načina. Jedan način temelji se na teoremu semantičke dedukcije. **Kako riječima glasi teorem semantičke dedukcije?**

A Ciljna formula je logička posljedica premisa ako i samo ako je ciljna formula istinita kada su istinite premise

B Premise i negacija cilja zajedno su zadovoljivi ako i samo ako je ciljna formula logička posljedica premisa

C Ako su premise nezadovoljive, bilo koja formula logička je posljedica takvih premisa

D Ciljna formula logička je posljedica premisa ako i samo ako je implikacija s premisa na ciljnu formulu tautologija

14 (P) Na postupak dokazivanja rezolucijom opovrgavanjem možemo gledati kao na problem pretraživanje prostora stanja, gdje stanja odgovaraju skupu klauzula (onih zadanih i onih izvedenih), a prijelazi između stanja odgovaraju primjeni rezolucijskog pravila na jedan par klauzula. Neka skup premisa sadrži 8 klauzula, a negirani cilj 2 klauzule. Za dokazivanje koristimo dvije strategije: strategiju zasićenja po razinama (LSS) i strategiju skupa potpore (SOS). Jednom razriješeni par klauzula više se ne razrješava. Izračunajte gornju ogradu na ukupan broj primjena rezolucijskog pravila za LSS i za SOS, ako znamo da je cilj dokaziv u tri koraka (tri primjene rezolucijskog pravila). **Koliko je puta gornja ograda za LSS veća od gornje ograde za SOS?**

A $7.51\times$ B $4.53\times$ C $5.71\times$ D $9.65\times$

15 (P) Ispravnost je važno svojstvo pravila zaključivanja. **Koje je od sljedećih pravila zaključivanja ispravno?**

A $A \vee B \vdash B$ B $A \to (B \to C) \vdash (A \wedge B) \to C$ C $B, A \to B \vdash A$ D $A \to B, B \to C \vdash C \to A$

16 (T) Unifikacija je postupak svođenja izraza K_1 i K_2 iz FOL na identičan oblik primjenom odgovarajuće supstitucije varijabli, koju nazivamo "unifikator". Postupak unifikacije ključan je za zaključivanje u FOL. Neka α, β i θ označavaju supstitucije, a "$\circ$" kompoziciju dviju supstitucija. **Što mora vrijediti, a da je supstitucija θ najopćenitiji zajednički unifikator (MGU)?**

A $\forall\alpha\big((K_1\alpha = K_2\alpha) \to \exists\beta(\alpha = \theta \circ \beta)\big)$
B $\exists\alpha\big((K_1\alpha = K_2\alpha) \to \forall\beta(\theta = \alpha \circ \beta)\big)$
C $\forall\alpha\forall\beta\big((K_1\theta = K_2\theta) \to (\theta = \alpha \circ \beta)\big)$
D $\forall\beta\exists\alpha\big((K_1\alpha = K_2\alpha) \to (\beta = \theta \circ \alpha)\big)$

4. Logičko programiranje i ekspertni sustavi (4 pitanja)

17 (T) Zaključivanje u ekspertnim sustavima može biti izvedeno kao ulančavanje unaprijed ili ulančavanje unazad. **U kojim je primjenama zaključivanje unazad bolji izbor od ulančavanja unaprijed?**

A Kada ako–onda pravila imaju različite prioritete koji se mogu mijenjati tijekom izvođenja programa

B Kada novoizvedeni zaključci mogu invalidirati ranije izvedene zaključke

C Kada je broj mogućih hipoteza malen, a broj potpora (dokaza) za te hipoteze potencijalno velik

D Kada je znanje moguće dekomponirati u granule znanja i formalizirati ih ako–onda pravilima

18 (R) Razmatramo sljedeći program u Prologu:

```
roditelj(kronos, zeus).
roditelj(zeus, minos).
roditelj(europa, minos).
potomak(X, Y) :- roditelj(Y, X).
potomak(X, Y) :- roditelj(Y, Z), potomak(X, Z).
bog(kronos).
bog(X) :- potomak(X, Y), bog(Y).
polubog(X) :- potomak(X, Y), not(bog(Y)).
```

Nad ovako definiranom bazom izvodimo upit `polubog(minos)`. Nacrtajte Prologovo stablo dokaza za ovaj upit do trenutka drugog povratka u pretraživanju. Pritom za dokazivanje negiranog cilja koristite dva čvora (jedan za `not(P(X))` i drugi za `P(X)`), dok prazni stog ne brojite kao čvor. **Koliko čvorova ima stablo dokaza neposredno prije drugog povratka u pretraživanju?**

A 12 B 10 C 8 D 6

19 (T) Prolog je deklarativan programski jezik, ali sadrži neke nedeklarativne aspekte. Jedan aspekt nedeklarativnosti u Prologu jest taj da u nekim slučajevima ne vrijedi komutativnost operatora "∧", npr., program `P(X,Y) :- Q(X), R(Y)` i program `P(X,Y) :- R(Y), Q(X)` ne evaluiraju se nužno identično. **Zbog čega u Prologu u ovom slučaju može postojati odstupanje proceduralnog značenja od deklarativnog?**

A Prolog ne provodi standardizaciju klauzula, pa različiti redoslijed klauzula može dati različite unifikatore

B Prolog koristi rezoluciju SLD i pretraživanje u dubinu, pa dokazivanje ovisi o redoslijedu atoma na stogu

C Prolog koristi definitne klauzule, pa do odstupanja dolazi kada neku FOL formulu ne možemo izraziti u Prologu

D Prolog pretpostavlja zatvorenost svijeta, no u stvarnosti činjenice koje nisu u bazi znanja mogu biti istinite

20 (R) Baza znanja ekspertnog sustava sadrži sljedeća pravila:

(1) AKO $(A = a_2) \wedge (D = d_2)$ ONDA $C = c_1$
(2) AKO $(F = f_3) \vee (B = b_3)$ ONDA $C = c_2$
(3) AKO $(E = e_1) \vee (B = b_1)$ ONDA $(A = a_1) \wedge (D = d_2)$
(4) AKO $F = f_1$ ONDA $D = d_2$
(5) AKO $F = f_2$ ONDA $E = e_2$
(6) AKO $(B = b_3) \vee (D = d_1)$ ONDA $(E = e_1) \wedge (A = a_2)$

Sustav koristimo za izvođenje vrijednosti varijable C ulančavanjem unazad. Prednost imaju pravila s manjim rednim brojem. Pravila koja su jednom palila više ne mogu paliti. Jednom izvedeni međuciljevi brišu se sa stoga, neovisno o njihovoj poziciji na stogu. Na možebitne upite od strane sustava, korisnik odgovara sa $B = b_3$ i $F = f_1$. **Što radi ekspertni sustav pri izvođenju vrijednosti varijable C?**

A Pali četiri pravila i izvodi $C = c_1$
B Pali tri pravila i izvodi $C = c_2$
C Pali pet pravila i izvodi $C = c_1$
D Odbacuje pravilo 6 te kasnije pali pravilo 3

Međuispit iz Uvoda u umjetnu inteligenciju (2021./2022.)

– NEKORIGIRANA VERZIJA –

*Ispit se sastoji od **20 pitanja** i ukupno nosi **20 bodova**. Točan odgovor nosi 1 bod, a netočan -1/3 boda. Trajanje ispita je **150 minuta**. Primjerak ispita trebate predati zajedno sa svojim rješenjima.*

1. Pretraživanje prostora stanja i heurističko pretraživanje (7 pitanja)

1 (R) Prostor stanja pretražujemo algoritmom A^*. Skup stanja je $S = \{a, b, c, d, e, f\}$, a funkcija sljedbenika je $succ(a) = \{(b, 2), (c, 2)\}$, $succ(b) = \{(c, 5), (d, 2)\}$, $succ(c) = \{(d, 1), (f, 20)\}$, $succ(d) = \{(e, 2)\}$, $succ(e) = \{(f, 14)\}$ te $succ(f) = \emptyset$. Heurističke vrijednosti stanja su $h(a) = 16$, $h(b) = 6$, $h(c) = 14$, $h(d) = 4$, $h(e) = 2$, $h(f) = 0$. Početno stanje je a, a ciljno f. Izvedite korake algoritma A^*, bilježeći u svakom koraku sadržaj liste otvorenih čvorova O i skupa zatvorenih čvorova C. U nultom koraku algoritma vrijedi $O = [(a, 0)]$ i $C = \emptyset$. **Koji je sadržaj liste O i skupa C nakon petog koraka izvođenja algoritma?**

A $O = [(d, 3), (f, 22)]$, $C = \{(a, 0), (b, 2), (c, 7), (e, 6)\}$

B $O = [(d, 3), (f, 20)]$, $C = \{(a, 0), (b, 2), (c, 2), (e, 6)\}$

C Algoritam ne dostiže peti korak

D $O = [(d, 3), (f, 20)]$, $C = \{(a, 0), (b, 2), (c, 2)\}$

2 (P) Razmatramo problem slagalice dimenzija 4×4 s prosječnom dubinom rješenja $d = 12$. Za nalaženje rješenja algoritam pretraživanja u dubinu u prosjeku koristi 720 B memorije. **Koliko bi memorije u prosjeku koristio algoritam pretraživanja u širinu na ovome problemu?**

A 10.1 MB B 25.2 MB C 240 MB D 14.4 KB

3 (P) Stanje s slagalice 3×3 neka je $[[1, 2, \square], [4, 5, 3], [7, 8, 6]]$, dok je ciljno stanje $[[1, 2, 3], [4, 5, 6], [7, 8, \square]]$. Neka je h optimistična heuristika. **Među ponuđenim vrijednostima, koju vrijednost $h(s)$ može poprimiti, a da je najviše obaviještena?**

A 3 B 8 C 1 D 6

4 (T) Prednost algoritma A^* nad algoritmom pretraživanja s jednolikom cijenom (UCS) jest ta da algoritam A^* koristi heuristiku, dok ju algoritam UCS ne koristi. **Kako se ta prednost manifestira u praksi?**

A Ako je heuristika optimistična, A^* će generirati manji ili jednak broj čvorova od algoritma UCS

B Neovisno o svojstvima heuristike, A^* će ispitati jednak ili manji broj čvorova od onoga koji će ispitati UCS

C Ako rješenje postoji, A^* će u manje koraka pronaći rješenje manje ili jednake cijene od onoga koje nalazi UCS

D Ako je heuristika optimistična, A^* će pronaći put koji je manje cijene od onoga koji pronalazi UCS

5 (P) Oblikujemo računalnu igru u kojoj se računalno upravljani igrači kreću po 2D mapi i skupljaju raznorazne artefakte. U jednom trenutku igrači trebaju u što manje poteza doći do zajedničkog zbornog mjesta. Igrači se mogu kretati u četiri smjera (gore, dolje, lijevo, desno), pri čemu rade po jedan korak. Mapa sadržava prepreke (zidove), kroz koje igrači ne mogu proći. Mapa također sadržava slučajan broj slučajno raspoređenih teleportera, koji igrača mogu trenutačno (u samo jednom koraku) transportirati na slučajno mjesto na mapi. No, jednom upotrijebljeni teleporter više se ne može upotrijebiti. Za ovu igru želimo oblikovati optimističnu heuristiku koja će navoditi računalno upravljane igrače do zbornog mjesta. Za oblikovanje heuristike koristimo metodu relaksacije. **Koja od sljedećih relaksacija igre daje optimističnu heuristiku?**

A Teleporteri se mogu koristiti neograničeno mnogo puta

B Teleporteri uvijek transportiraju igrača do teleportera koji je najbliži zbornom mjestu

C Igrači se mogu kretati dijagonalno, ali ne mogu koristiti teleportere

D Igrači mogu prolaziti kroz zidove i dvaput koristiti isti teleporter

6 (R) Algoritmom slijepog pretraživanja rješavamo problem Hanojskih tornjeva s tri klinova i tri diska. Klinovi su označeni sa A, B i C, a diskovi sa 1, 2 i 3, pri čemu broj odgovara veličini diska (disk 1 je najmanji, a disk 3 najveći). Stanje možemo označiti kao AxByCz, gdje x, y i z označavaju diskove koji se nalaze na klinovima A, B, odnosno C, krenuvši od dna prema vrhu (ako je klin prazan, ispuštamo njegovo slovo). Npr., u stanju B21C3 na klinu B su diskovi 2 (ispod) i 1 (gore), a na klinu C je disk 3. Početno stanje je A321. Zadatak je prebaciti diskove s klina A na klin C, tj. pronaći stanje C321, koristeći klin B kao pomoćni klin. Pritom se u svakom koraku prebacuje samo jedan disk, uz uvjet da veći disk ne smije biti položen na manji disk. Koristimo algoritam iterativnog pretraživanja u dubinu (IDS) bez liste posjećenih stanja. Algoritam čvorove proširuje leksikografskim poretkom gdje znamenke dolaze prije slova, npr., stanje A32B1 prethodi stanju A3B2C1. **Koliko će čvorova generirati algoritam IDS prije nego što generira čvor sa stanjem A3C21**?

A 22 B 29 C 16 D 25

7 (T) Svi algoritmi slijepog pretraživanja u dubinu eksponencijalne su vremenske složenosti, međutim konkretni izrazi za asimptotsku složenost ipak se malo razlikuju. Asimptotska vremenska složenost algoritma pretraživanja u širinu (BFS) je $\mathcal{O}(b^{d+1})$, a algoritma iterativnog pretraživanja u dubinu (IDS) $\mathcal{O}(b^d)$. **Zašto asimptotska vremenska složenost algoritma IDS nije ista kao i ona algoritma BFS,** $\mathcal{O}(b^{d+1})$?

A IDS ima dubinsko ograničenje koje povećava do dubine rješenja d, pa nikada ne proširuje čvorove razine d

B IDS čvorove na razini d proširuje b^d puta, dok BFS dodatno proširuje svaki čvor na razini d osim ciljnog

C IDS generirane čvorove umeće na početak, a BFS na kraj liste, pa BFS na razini d generira b dodatnih čvorova

D IDS više puta proširuje iste čvorove, ali na posljednoj razini proširuje sve čvorove osim ciljnog

2. Igranje igara (3 pitanja)

8 (R) Razmatramo igru dvaju minimax-igrača. Stablo igre definirano je prijelazima $succ(A) = \{B, C, D\}$, $succ(B) = \{E, F\}$, $succ(C) = \{G, H\}$, $succ(D) = \{I, J\}$, $succ(G) = succ(F) = \{K, L\}$, $succ(H) = succ(E) = succ(I) = \{M, N\}$ te $succ(J) = \{O, P\}$. Igrač MAX koristi heuristiku h_1, a igrač MIN heuristiku h_2. Vrijednosti heuristika su sljedeće (obje heuristike definiraju dobit dotičnog igrača):

	E	F	G	H	I	J	K	L	M	N	O	P
h_1	1	−2	3	2	0	5	−1	0	−2	−1	0	2
h_2	0	5	2	4	−1	2	3	2	1	−1	1	−2

Oba igrača pretražuju najviše dva poteza unaprijed (jedan svoj i jedan suparnički). Igru započinje igrač MAX u stanju A. **Kroz koja stanja će se razvijati igra?**

A $A, D, I, \ldots$ B $A, D, J, \ldots$ C $A, C, G, \ldots$ D $A, C, H, \ldots$

9 (T) Igrač MAX primjenjuje algoritam minimax s heuristikom h na korijen stabla igre kako bi odredio svoj optimalan potez. Nakon što povuče taj potez, na redu je protivnik MIN, koji također vuče svoj optimalan potez. Igrač MAX zatim nanovo primjenjuje algoritam MAX na novom stanju igre. Pritom igrač MAX ponovo izračunava minimax-vrijednosti i za one čvorove za koje je ranije (za prethodni potez) već bio izračunao minimax-vrijednosti. **Zbog čega je potrebno ponovno izračunavati minimax-vrijednosti za te čvorove?**

A Heuristika h nije savršena, pa igrač MAX ne može očekivati da će ostvariti točno onu isplatu koju predviđa heuristika h

B Protivnik MIN može koristi heuristiku manje točnosti od heuristike h, pa njegovi potezi mogu biti drugačiji od onih koje je predvidio igrač MAX

C Algoritam minimax radi pretraživanje u dubinu, pa ne pohranjuje čvorove na stazama izvan poteza koji je predvidio igrač MAX

D Ako je za prethodni izračun korištena heuristika, novi izračun bit će informiraniji jer će ići dvije razine dublje od prethodnog

10 (P) Razmatramo dva igraća algoritma: jedan s heuristikom h_1 i dubinom pretraživanja d_1 te drugi s heuristikom h_2 i dubinom pretraživanja d_2. Oba algoritma koriste strategiju minimax. **Kada će prvi algoritam češće pobjeđivati drugi algoritam?**

A Ako $h_1 > h_2$ B Ako $h_2 > h_1$ C Ako $d_1 = d_2$ i $h_1 > h_2$ D Ako $h_1 = h_2$ i $d_1 > d_2$

3. Prikazivanje znanja i automatsko zaključivanje (6 pitanja)

11 (P) Na postupak dokazivanja rezolucijom opovrgavanjem možemo gledati kao na problem pretraživanje prostora stanja, gdje stanja odgovaraju skupu klauzula (onih zadanih i onih izvedenih), a prijelazi između stanja odgovaraju primjeni rezolucijskog pravila na jedan par klauzula. Neka skup premisa sadrži 7 klauzula, a negirani cilj 2 klauzule. Za dokazivanje koristimo dvije strategije: strategiju zasićenja po razinama (LSS) i strategiju skupa potpore (SOS). Jednom razriješeni par klauzula više se ne razrješava. Izračunajte gornju ograda na ukupan broj primjena rezolucijskog pravila za LSS i za SOS, ako znamo da je cilj dokaziv u tri koraka (tri primjene rezolucijskog pravila). **Koliko je puta gornja ograda za LSS veća od gornje ograde za SOS?**

A $4.53\times$ B $9.65\times$ C $5.71\times$ D $7.51\times$

12 (T) Logička posljedica središnji je pojam u logici i razlog zašto uopće koristimo sustav formalne logike za prikazivanje znanja. U propozicijskoj logici, logičku posljedicu moguće je dokazati na nekoliko načina. Jedan način temelji se na teoremu semantičke dedukcije. **Kako riječima glasi teorem semantičke dedukcije?**

A Premise i negacija cilja zajedno su zadovoljivi ako i samo ako je ciljna formula logička posljedica premisa

B Ciljna formula je logička posljedica premisa ako i samo ako je ciljna formula istinita kada su istinite premise

C Ako su premise nezadovoljive, bilo koja formula logička je posljedica takvih premisa

D Ciljna formula logička je posljedica premisa ako i samo ako je implikacija s premisa na ciljnu formulu tautologija

13 (R) Zadane su premise: *"Ivo voli sve vrste hrane. Jabuka i piletina su hrana. Branko jede lješnjake i nije mu zlo od njih. Vesna jede sve što i Branko. Ako netko nešto jede i nije mu od toga zlo, onda je to hrana."* Formalizirajte ove premise u FOL i pretvorite ih u klauzule. Pritom koristite $V(x, y)$ za "x voli y", $J(x, y)$ za "x jede y", "$Z(x, y)$" za "x-u je zlo od y", i "$H(x)$" "x je hrana". Zatim rezolucijom opovrgavanjem uz strategiju skupa potpore dokažite cilj *Ivo voli lješnjake*. **Koliko je minimalno koraka (primjena rezolucijskog pravila) potrebno za dokaz?**

A 3 B 4 C 5 D 6

14 (P) Ispravnost je važno svojstvo pravila zaključivanja. **Koje je od sljedećih pravila zaključivanja ispravno?**

A $A \to (B \to C) \vdash (A \vee B) \to C$

B $A \to (B \vee C), \neg C \vdash A \to B$

C $A \to B, B \to C \vdash C \to A$

D $\neg A, A \to B \vdash \neg B$

15 (R) Dani su FOL atomi $P(f(x), y, g(w, f(z)))$ i $P(y, f(g(a, z)), x)$, gdje je P predikat, a je konstanta, a f i g su funkcije. **Što je najopćenitji zajednički unifikator ovih dvaju atoma?**

A $P(f(g(a, b)), f(g(a, b)), g(a, b))$

B Atome nije moguće unificirati

C $P(f(x), f(g(x, y)), g(a, b))$

D $P(f(y), f(g(a, z)), g(w, b))$

16 (T) Unifikacija je postupak svođenja izraza K_1 i K_2 iz FOL na identičan oblik primjenom odgovarajuće supstitucije varijabli, koju nazivamo "unifikator". Postupak unifikacije ključan je za zaključivanje u FOL. Neka α, β i θ označavaju supstitucije, a "$\circ$" kompoziciju dviju supstitucija. **Što mora vrijediti, a da je supstitucija θ najopćenitiji zajednički unifikator (MGU)?**

A $\exists\alpha\big((K_1\alpha = K_2\alpha) \to \forall\beta(\theta = \alpha \circ \beta)\big)$

B $\forall\alpha\forall\beta\big((K_1\theta = K_2\theta) \to (\theta = \alpha \circ \beta)\big)$

C $\forall\beta\exists\alpha\big((K_1\alpha = K_2\alpha) \to (\beta = \theta \circ \alpha)\big)$

D $\forall\alpha\big((K_1\alpha = K_2\alpha) \to \exists\beta(\alpha = \theta \circ \beta)\big)$

4. Logičko programiranje i ekspertni sustavi (4 pitanja)

17 (T) Zaključivanje u ekspertnim sustavima može biti izvedeno kao ulančavanje unaprijed ili ulančavanje unazad. **U kojim je primjenama zaključivanje unazad bolji izbor od ulančavanja unaprijed?**

A Kada novoizvedeni zaključci mogu invalidirati ranije izvedene zaključke

B Kada je znanje moguće dekomponirati u granule znanja i formalizirati ih ako–onda pravilima

C Kada je broj mogućih hipoteza malen, a broj potpora (dokaza) za te hipoteze potencijalno velik

D Kada ako–onda pravila imaju različite prioritete koji se mogu mijenjati tijekom izvođenja programa

18 (R) Baza znanja ekspertnog sustava sadrži sljedeća pravila:

(1) AKO $(A = a_2) \wedge (D = d_2)$ ONDA $C = c_1$
(2) AKO $(F = f_3) \vee (B = b_3)$ ONDA $C = c_2$
(3) AKO $(E = e_1) \vee (B = b_1)$ ONDA $(A = a_1) \wedge (D = d_2)$
(4) AKO $F = f_1$ ONDA $D = d_2$
(5) AKO $F = f_2$ ONDA $E = e_2$
(6) AKO $(B = b_3) \vee (D = d_1)$ ONDA $(E = e_1) \wedge (A = a_2)$

Sustav koristimo za izvođenje vrijednosti varijable C ulančavanjem unazad. Prednost imaju pravila s manjim rednim brojem. Pravila koja su jednom palila više ne mogu paliti. Jednom izvedeni međuciljevi brišu se sa stoga, neovisno o njihovoj poziciji na stogu. Na možebitne upite od strane sustava, korisnik odgovara sa $B = b_3$ i $F = f_1$. **Što radi ekspertni sustav pri izvođenju vrijednosti varijable C?**

A Izvodi $A = a_2$ te kasnije $A = a_1$

B Odbacuje pravilo 6 te kasnije pali pravilo 3

C Završava sa 4 činjenica u radnoj memoriji

D Odbacuje pravilo 3 te kasnije pali pravilo 4

19 (R) Razmatramo sljedeći program u Prologu:

```
roditelj(kronos, zeus).
roditelj(zeus, minos).
roditelj(europa, minos).
potomak(X, Y) :- roditelj(Y, X).
potomak(X, Y) :- roditelj(Y, Z), potomak(X, Z).
bog(kronos).
bog(X) :- potomak(X, Y), bog(Y).
polubog(X) :- potomak(X, Y), not(bog(Y)).
```

Nad ovako definiranom bazom izvodimo upit `polubog(minos)`. Nacrtajte Prologovo stablo dokaza za ovaj upit do trenutka drugog povratka u pretraživanju. Pritom za dokazivanje negiranog cilja koristite dva čvora (jedan za `not(P(X))` i drugi za `P(X)`), dok prazni stog ne brojite kao čvor. **Koliko čvorova ima stablo dokaza neposredno prije drugog povratka u pretraživanju?**

A 8 B 6 C 12 D 10

20 (T) Prolog je deklarativan programski jezik, ali sadrži neke nedeklarativne aspekte. Jedan aspekt nedeklarativnosti u Prologu jest taj da u nekim slučajevima ne vrijedi komutativnost operatora "$\wedge$", npr., program `P(X,Y) :- Q(X), R(Y)` i program `P(X,Y) :- R(Y), Q(X)` ne evaluiraju se nužno identično. **Zbog čega u Prologu u ovom slučaju može postojati odstupanje proceduralnog značenja od deklarativnog?**

A Prolog ne provodi standardizaciju klauzula, pa različiti redoslijed klauzula može dati različite unifikatore

B Prolog koristi rezoluciju SLD i pretraživanje u dubinu, pa dokazivanje ovisi o redoslijedu atoma na stogu

C Prolog koristi definitne klauzule, pa do odstupanja dolazi kada neku FOL formulu ne možemo izraziti u Prologu

D Prolog pretpostavlja zatvorenost svijeta, no u stvarnosti činjenice koje nisu u bazi znanja mogu biti istinite

Grupa	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	B	A	C	B	D	B	D	A	C	A	D	A	C	C	C	D	A	B	C	C
B	C	A	B	A	A	D	D	B	A	A	D	B	A	D	C	B	B	D	D	C
C	A	D	A	A	D	C	D	C	B	A	D	B	D	B	B	D	A	D	B	A
D	C	A	A	D	A	A	B	D	A	B	B	D	B	A	C	A	D	B	C	C
E	C	B	B	D	C	A	C	B	C	D	A	C	A	C	C	D	A	B	B	D
F	D	D	C	C	B	A	B	B	A	C	B	B	D	C	D	A	C	A	C	D
G	B	D	C	D	A	A	A	C	B	B	B	A	D	D	B	A	C	A	B	B
H	B	A	C	A	D	D	A	C	D	D	D	D	B	B	B	D	C	A	C	B