1. Rešavanje problema korišćenjem pretrage

1.1 Navesti barem pet opštih elemenata svakog problema pretrage.

- *skup mogućih stanja* - u toku procesa pretrage razmatraju se različita stanja. Za odlučivanje u datom trenutku potrebno je poznavanje skupa svih stanja (ne samo onih dostupnih iz polaznog stanja).
- *polazno stanje* - rešavanje problema kreće od jednog određenog stanja, koje nazivamo polaznim stanjem.
- *test cilja* - problem je rešen ako se dođe do ciljnog stanja, završnog stanja. Potrebno je da postoji raspoloživ efektivan test koji proverava da li se došlo do ciljnog stanja tj. do završetka procesa pretrage.
- *skup mogućih akcija* - u svakom koraku pretrage može se preduzeti neki korak, neka akcija. Niz akcija preduzetih u odgovarajućim trenucima treba da dovede do rešenja problema. Skup mogućih akcija može biti isti u svakom stanju ili može da se razlikuje od stanja do stanja, što zavisi od problema koji se rešava.
- *funkcija prelaska* - ova funkcija preslikava par stanje-akcija u novo stanje, dobijeno izborom neke akcije u nekom stanju.
- *funkcija cene* - ova funkcija preslikava par stanje-akcija u numeričku vrednost — cenu preduzimanja date akcije u datom stanju.

1.2 Kako se, prema dostupnosti informacija koje mogu pomoći u pronalaženju ciljnog stanja u toku pretrage, dele problemi pretrage?

Na probleme informisane i neinformisane pretrage.

1.3 Šta je kombinatorna eksplozija?

Kombinatorna eksplozija je pojava koja se javlja u problemima i čini da njihovo rešavanje zahteva razmatra ogroman broj mogucnosti.

2. Informisana pretraga

2.1 Kako se naziva algoritam pretrage koji uvek bira lokalno optimalne akcije?
Pohlepna pretraga.

2.2 Šta, umesto globalnog ekstremuma, pohlepna pretraga može vratiti kao rezultat?
Lokalni ekstremum.

2.3 Šta je plato u problemima pretrage?
Oblast prostora pretrage u komr funkcija cilja ima konstantnu vrednost.

2.4 Kako se zove oblast prostora pretrage u kojem ciljna funkcija ima konstantnu vrednost?
Plato.

2.5 Čemu je jednaka vrednost *f(n)* koja se u algoritmu A* pridružuje čvoru *n*?
f(n) predstavlja funkciju evaluacije za čvor *n*: *f(n)=g(n)+h(n)*, gde je *g(n)* cena puta od početnog čvora do *n*, a *h(n)* heuristička procena od čvora *n* do cilja.

2.6 Šta, za razliku od Dejkstrinog algoritma, algoritam A* uzima u obzir?
A* uzima heurističku funkciju *h(n)* kojom se može izvršiti usmeravanje pretrage.

2.7 Da li je algoritam A* opštiji od Dejkstrinog algoritma? Da li je Dejkstrin algoritam opštiji od algoritma A*?
Dejkstrin algoritam se može dobiti od A*, kad je *h(n)=0*, stoga je A* opštiji.

2.8 Kada se algoritam A* ponaša isto kao Dejkstrin algoritam?
Kada je *h(n)=0.*

2.9 Da li se tokom primene algoritma A*, može promeniti vrednost g(n) za čvor n? Da li se tokom primene algoritma A*, može promeniti vrednost h(n) za čvor n? Da li se tokom primene algoritma A*, može promeniti vrednost f(n) za čvor n?
g*(n)* može. *h(n)* ne može. *f(n)* može.

2.10 Kako se zove skup iz kojeg se u glavnoj petlji algoritma A* bira tekući čvor?
Otvorena lista.

2.11 Kakva struktura podataka se koristi za čuvanje vrednosti funkcije evaluacije u okviru algoritma A*? Obrazložiti.
Otvorena lista se često implementira kao binarni min-hip, a zatvorena lista kao heš tabela. Operacije za dodavanje i brisanje elemenata iz otvorene liste su složenosti *O(log|V|).* Dodavanje u zatvorenu listu je složenosti *O(1).*

2.12 Da li je, na samom početku primene algoritma A*, lista zatvorenih čvorova prazna?
Da.

2.13 Koji čvor se, prilikom primene algoritma A*, prvi dodaje u listu otvorenih čvorova?
Polazni čvor.

2.14 Kada se, u okviru algoritma A*, u listu zatvorenih čvorova dodaje novi element?
Kada obradimo sve susede tekućeg čvora.

2.15 Tokom primene algoritma A*, ako se ispituje tekući čvor i naiđe na njegov susedni čvor *v* koji nije u zatvorenoj listi, ali jeste u otvorenoj listi, šta treba uraditi?
Treba proveriti da li je put od polaznog čvora do čvora *v* preko tekućeg čvora bolji (kraći/jeftiniji) od poslednjeg puta do *v* (trenutna vrednost *g(v)*). Ako jeste, treba promeniti informaciju o roditelju čvora *v* na tekući čvor i ažurirati vrednost *f(v)*.

2.16 Da li je na kraju primene algoritma A* lista otvorenih čvorova nužno prazna?
Ne.

2.17 Da li je na kraju primene algoritma A* lista zatvorenih čvorova nužno prazna?
Ne.

2.18 Šta je uslov zaustavljanja za algoritam A*?
Uslov je da je broj stanja i broj akcija konačan.

2.19 Za koje grafove je algoritam A* najpogodniji za primenu?
Za retke grafove; kad graf ima manji broj grana, A* će brže izvršti obradu čvorova.

2.20 Kada kažemo da je funkcija heuristike *h* u algoritmu A* dopustiva, a kada kažemo da je konzistentna?
Dopustiva: $0 \leq h(n) \leq h^*(n)$.
Konzistentna: $c(n,m) + h(m) \geq h(n)$.

2.21 Šta znači to da je algoritam A* potpun?
Znači da se završava i da daje neko rešenje (ne mora biti optimalno).

2.22 Pod kojim uslovom je algoritam A* potpun?
Pod uslovima da je broj čvorova i broj akcija konačan, i da postoji neki put između početnog čvora i cilja.

2.23 Pod kojim uslovom je algoritam A* optimalan (nalazi najkraći put)?
Pod uslovom da je heuristika dopustiva.

2.24 Ukoliko je *f** funkcija koja odgovara najkraćem putu između dva čvora u grafu, koje čvorove obrađuje algoritam A*?

U ovom slučaju nije nam ni potreban algoritam A*. Međutim, ako baš želimo da ga primenimo, koristimo *f** kao heuristiku (*f** je dopustiva).

2.25 Kada je u algoritmu A* broj obrađenih čvorova polinomski u odnosu na dužinu najkraćeg puta?

Kada je heuristika kvalitetna, odnosno ako zadovoljava: $|h(x) - h^*(x)| \leq O(\log h^*(x))$

2.26 Kako se zove rastojanje između dva čvora u kojem se broji ukupan broj polja pređenih horizontalno ili vertikalno od prvog do drugog?

Menhetn rastojanje.

2.27 Koliko je Mehnetn rastojanje između donjeg levog i gornjeg desnog polja šahovske table?

14

2.28 Kada se algoritam A* primenjuje na uniformnoj mreži, koja funkcija se obično primenjuje kao heuristika? Da li je ova heuristika dopustiva? Zašto se primenjuje ova heuristika, šta je njena ključna osobina?

Koristi se Menhetn rastojanje, a omogućava se i kretanje dijagonalno. Kretanje dijagonalno penalizujemo sa 1,41, a horizontalno i vertikalno sa 1. Često se navedene vrednosti množe sa, npr. 10^n, kako bi se uprostio račun.
Ukoliko se koristi samo Menhetn rastojanje, heuristika je dopustiva, ali kada se koriste modifikacije, onda nije jer može da preceni stvarno rastojanje. U praksi, pristup sa modifikacijama daje zadovoljavajuće i brže rezultate.

2.29 Kada se algoritam A* primenjuje na uniformnoj mreži, šta se obično koristi kao cena puta do susednog čvora koji je desno, a šta do susednog čvora gore-desno?

1 za desno, 1.41 za gore desno. Često se množi sa 10^n ta vrednost, radi lakšeg računa.

3. Igranje strateških igara

3.1 Opisati ukratko Šenonove strategije.
Dve opšte strategije za izbor poteza:

A: Minimaks procedurom vrši se pretraživanje stabla igre sa određenom funkcijom evaluacije i ocenjivanje legalnih poteza. Bira se potez sa najboljom ocenom.

B: Potez se bira na osnovu trenutne pozicije/situacije u igri i na osnovu odgovarajuće, unapred pripremljene tabele.

3.2 Šta znači da je funkcija evaluacije koja se koristi u strateškim igrama statička?
Funkcija evaluacije se, u skladu sa specifičnim karakteristikama konkretne igre, dodeljuje poziciji pri čemu se ne ispituju ni pozicije iz kojih se došlo u tu poziciju niti mogući nastavci. Gotovo sve znanje o igri koje se koristi u središnjici partije sadržano je u funkciji evaluacije i u najvećoj meri od nje zavisi kvalitet igre programa.

3.3 Koja je najjednostavnija funkcija evaluacije u igrama nulte sume?
Najjednostavnija je *trovrednosna* funkcija - ona se primenjuje samo na završne pozicije igre i ima samo tri različite vrednosti – za pobedu prvog, za pobedu drugog igrača i za nerešen ishod. Trovrednosna funkcija zahteva pretraživanje stabla igre do završnih čvorova, pa je, zbog potencijalno velike dubine pretraživanja, ova funkcija za većinu igara praktično neupotrebljiva.

3.4 Ako je statička ocena neke šahovske pozicije jednaka 0, šta to govori?
Nerešen ishod.

3.5 Ako je statička ocena neke šahovske pozicije jednaka c, koja je ocena pozicije koja je dobijena tako što su sve figure promenile boju?
-c

3.6 Zašto se tako zove *minimaks* algoritam?
Algoritam karakteriše minimizovanje ocene kada je na potezu protivnik i maksimizovanje ocene kada je na potezu sam igrač.

3.7 Ako se pretraga vrši do iste dubine stabla igre, da li algoritam Minimax ispituje isti broj pozicija bez obzira na poredak poteza u jednom čvoru?
Da.

3.8 Po čemu se razlikuju α i β odsecanja?
Alfa-beta algoritam zasnovan je na tzv. alfa i beta odsecanju stabla igre i predstavlja heuristikama ubrzan algoritam minimaks

3.9 Da li algoritam Alfa-beta uvek vraća isti rezultat kao algoritam Minimax?
Da.

3.10 Da li algoritam Alfa-beta uvek obradi manje čvorova nego algoritam Minimax?

Ne.

3.11 U kom slučaju (osnovni) algoritam Alfa-beta obrađuje isti broj čvorova kao i Minimax?

Ako se u svakom čvoru potezi ispituju od najlošijeg ka najboljem (tada nema odsecanja).

3.12 Da li algoritam Minimax može u nekom slučaju, pretražujući do iste dubine da obiđe manji broj čvorova od algoritma Alfa-beta?

Ne.

3.13 Da li Alfa-beta algoritam, u odnosu na Minimax algoritam: (a) daje iste poteze, ali brže; (b) daje nešto lošije poteze, ali znatno brže; (c) daje bolje poteze i to brže; (d) daje bolje poteze ali nešto sporije?

(c) Daje bolje poteze i to brže.

3.14 Kada je broj odsecanja u stablu igre najveći u algoritmu Alfa-beta?

Kada se najpre ispituju potezi najbolji u smislu tekućeg čvora (za korišćenu funkciju evaluacije i za zadatu dubinu).

3.15 Koja heuristika je zasnovana na činjenici da je broj Alfa-beta odsecanja u stablu igre najveći ako se najpre ispituje najbolji?

Killer.

3.16 Opisati ukratko heuristiku kiler.

Heuristika kiler ima za cilj da se u mnogim čvorovima najpre razmatra najbolji (ili makar veoma dobar) potez, kako bi bilo što više alfa i beta odsecanja. Ova heuristika je opšta i ne koristi specifična znanja o igri.

Neka se u pretraživanju stabla Alfa-beta algoritmom prvi put ocenjuje neki čvor na dubini *d* ($d \geq 1$) i neka je *W* najbolji pronađeni potez u smislu tog čvora. Taj potez zvaćemo kiler potezom za dubinu *d*. U svakom sledećem čvoru na dubini *d*, ispitivanje poteza počinjemo sa kiler potezom za tu dubinu. Ukoliko se pokaže da je za taj čvor bolji neki drugi potez (*W′*), onda taj potez postaje kiler potez za dubinu *d*. Ukoliko se pretraživanje stabla igre vrši do dubine d_{max}, opisana heuristika primenjuje se za sve dubine *d* takve da je
$1 \leq d \leq d_{max}-1$.

3.17 Kakav efekat se očekuje od heuristike kiler u iterativnoj primeni Alfa-beta algoritma i zašto?

Efekti iterativnog algoritma su, u svakoj iteraciji, slični efektima Alfa-beta/kiler algoritma, s tim što u iterativnom algoritmu postoji i kiler potez za početni čvor u pretraživanju. Ima izgleda da je u svakoj iteraciji taj kiler potez bolje odabran i da daje bolje rezultate (veći broj alfa i beta odsecanja). Druga važna i dobra osobina iterativnog algoritma je to što za slučaj prekida pretraživanja, praktično u svakom trenutku ima smisleni rezultat kao najbolji pronadeni potez za neku kompletno završenu iteraciju

3.18 Da li iterativni Alfa-beta algoritam sa kiler heuristikom daje uvek isti rezultat kao Alfa-beta algoritam nad istim stablom igre i za istu dubinu pretrage?

Primenom iterativnog Alfa-beta/kiler algoritma dobija se potez sa najboljom ocenom za datu funkciju evaluacije i datu dubinu pretraživanja, isto kao i primenom algoritama minimaks, Alfa-beta i Alfa-beta/kiler.

3.19 Da bi heuristika kiler funkcionisala i na nultom nivou stabla igre koji je algoritam potrebno koristiti?

Sve jedno koji algoritam koristimo, jer do dubine 1 nema alfa i beta odsecanja.

3.20 Koji algoritam je pogodan za igru sa vremenskim prekidima i zašto?

Iterativni Alfa-beta/kiler algoritam - jer praktično u svakom trenutku ima neku kompletno završenu iteraciju, te će biti izbegnuti makar najjednostavniji previdi.

3.21 Šta je to stabilno pretraživanje?

Stabilno pretraživanje - vrši se pretraživanje do neke fiksne dubine, ali se pretraživanje nastavlja i dalje ukoliko je, po nekom kriterijumu, završni čvor „nestabilan".

3.22 Neka je *A* deterministički algoritam za pretraživanje *(d, n, F)*-stabla i neka je $I_A(d, n, F)$ očekivani broj završnih čvorova koje algoritam *A* ispituje. Kako se definiše faktor grananja algoritma *A*?

$R_A(n, F) = \lim_{d \to inf} [I_A(d, n, F)]^{1/d}$

3.23 Koliki je faktor grananja za algoritam Minimax za šahovsku središnjicu?

Faktor grananja je između 35 i 38.

3.24 Koliki je faktor granjanja algoritma minimaks, ako se ispituje uniformno stablo stepena *n* i dubine *d*?

$R_{minimax}(n, F) = n$.

3.25 Ukoliko čvorovi stabla igre imaju stepen *n*, a pretražuje se do dubine *d*, koja je složenost algoritma Minimax a koja algoritma Alfa-beta za taj problem?

Minimax: $O(n^d)$, Alfa-beta: $O(n^{d/2})$.

3.26 U programiranju igara, da li se algoritmi Minimax tipa primenjuju u otvaranju, središnjici ili završnici?

U središnjici.

3.27 Navesti barem dve strategije za završnicu u programima za igre.

Skupovi pozicija kao klase ekvivalencija i Mali saveti.

3.28 Do koje dubine se vrši pretraga u Bramerovom pristupu za završnicu?

Do dubine 1.

4. Genetski algoritmi

4.1 Navesti opšti genetski algoritam.

1. Generisati početnu populaciju (generaciju) jedinki
2. Izračunati funkciju prilagođenosti za svaku
3. Izvršavaj petlju sve dok ne bude zadovoljen uslov zaustavljanja
   * Izaberi skup jedinki za reprodukciju
   * Primenom operatora ukršatanja kreirati nove jedinke i izračunati njihovu prilagođenost
   * Na osnovu starih i novih jedinki, kreiraj novu generaciju
4. Vrati najkvalitetniju jedinku u poslednjoj populaciji

4.2 Da li, u genetskim algoritmima, ciljna funkcija mora da bude:

- definisana za sve moguće jedinke (***da***/ne)
- diskretna (da/***ne***)
- neprekidna (da/***ne***)
- diferencijabilna (da/***ne***)

4.3 Da li, u genetskim algoritmima, funkcija prilagođenosti mora da bude:

- definisana za sve moguće jedinke (***da***/ne)
- diskretna (da/***ne***)
- neprekidna (da/***ne***)
- diferencijabilna (da/***ne***)

4.4 Ukoliko je genetskim algoritmom potrebno odrediti minimum pozitivne funkcije *f* na nekom intervalu, da li je pogodno kao funkciju prilagođenosti koristiti funkciju:

(a) f
(b) −f
(c) inverznu funkciju od f
(d) f ′

4.5 Ukoliko je genetskim algoritmom potrebno odrediti minimum pozitivne funkcije *f* na nekom intervalu, koju je funkciju koristiti kao funkciju prilagođenosti?

-f

4.6 U genetskim algoritmima, koja se reprezentacija jedinki najčešće koristi?

Reprezentacija u vidu nozova binarnih cifara.

4.7 Broj mogućih rešenja datog problema je 1 000 000. Ukoliko se za rešavanje ovog problema koristi genetski algoritam i binarna reprezentacija, kolika je onda dužina hromozoma?

$2^x = 1\,000\,000 \rightarrow x = 20$

4.8 Ako je za potrebe primene genetskog algoritma, domen {3, 4, 5, 6, 7, 8, 9, 10} predstavljen binarnim hromozomima dužine 3 (u istom poretku), kako će biti predstavljena jedinka 9?
110

4.9 Kako se generiše inicijalna populacija u genetskim algoritmima?
Najčešće se generiše slučajno.

4.10 Navesti dva genetska operatora.
Mutacija i ukrštanje.

4.11 Koliko genetski operatori ukrštanja i mutacije imaju ulaznih jedinki?
Ukrštanje ima 2 ulazne jedinke, a mutacija 1.

4.12 Šta je uloga selekcije u genetskim algoritmima?
Selekcija obezbeđuje čuvanje i prenošenje dobrih osobina populacije na sledeću generaciju.

4.13 Koje su vrste selekcije u genetskim algoritmima najpopularnije?
(a) Menhetn i ruletska;
(b) Menhetn i turnirska;
(c) ruletska i turnirska;
(d) ruletska i uniformna.

4.14 Koje vrste selekcija se najčešće koriste u genetskim algoritmima?
Ruletska i turnirska.

4.15 Kako se jedinka bira ruletskom selekcijom?
Veće šanse da učestvuju u reprodukciji imaju prilagođenije jedinke. Ako je *f(i)* vrednost funkcije prilagođenosti za jedinku *i,* a *N* broj jedinki u populaciji, verovatnoća da će jedinka *i* biti izabrana je:

$$p_i = \frac{f(i)}{\sum_j^N f(j)}$$

4.16 Ako je *f(i)* vrednost funkcije kvaliteta (prilagodenosti) za jedinku *i*, a *N* broj jedinki u populaciji, verovatnoća da će jedinka i ruletskom selekcijom biti izabrana da učestvuje u reprodukciji jednaka je $p_i = f(i)/x$, gde je *x* jednako:
(a) 1
(b) $\sum_j^N f(j)$
(c) $\sum_{j, j \neq i}^N f(j)$
(d) $\prod_j^N f(j)$

4.17 Ukoliko su vrednosti prilagođenosti jedinki a, b i c redom 2, 5, 8, koja je verovatnoća da će u ruletskoj selekciji biti izabrana jedinka b?

suma = 2 + 5 + 8 = 15
p(b) = f(b) / suma = 5 / 15 = 1/3

4.18 Genetskim algoritmom se traži maksimum funkcije $20-x^2$. Populacija sadrži (samo) jedinke (1), (-4), (2) i (3). Kolika je, za svaku od jedinki, verovatnoća da će biti izabrana za reprodukciju u ruletskoj selekciji?

$f(x) = 20 - x^2$

f(1) = 20 - 1 = 19
f(-4) = 20 - 16 = 4
f(2) = 20 - 4 = 16
f(3) = 20 - 9 = 11

suma = 19 + 4 + 16 + 11 = 50
p(1) = f(1) / suma = 19/50
p(-4) = f(-4) / suma = 4/50
p(2) = f(2) / suma = 16/50
p(3) = f(3) / suma = 11/50

4.19 U genetskim algoritmima, ako u jednoj generaciji postoje (samo) jedinke A, B i C sa vrednostima prilagodenosti 1, 2 i 3 (redom), koja je verovatnoća da pri ruletskoj selekciji jedinka B uđe u proces reprodukcije?

suma = 1 + 2 + 3 = 6
p(B) = f(B) / suma = 2 / 6 = 1/3

4.20 Opisati algoritam turnirske selekcije.

Za jedan turnir se bira slučajno *k* jedinki populacije. Pobednikom se smatra jedinka sa najvećom prilagođenosti ili se jedinke sortiraju po vrednosti funkcije prilagođenosti i *i*-ta jedinka se bira sa verovatnoćom $p(1-p)^{i-1}$, gde je verovatnoća *p* drugi parametar procesa turnirske selekcije.

4.21 Ako je u turnirskoj selekciji veličina turnira *k* jednaka 1, čemu je ona ekvivalentna?

Ruletskoj selekciji.

4.22 Dve jedinke-roditelja imaju binarne reprezentacije 1010 i 0101. Da li se nekom vrstom ukrštanja može dobiti kao njihov potomak:

(a) 0000
(b) 0011
(c) 1111

Zavisi o kom ukrštanju se radi. Ako se radi o uniformnom ukrštanju sa verovatnoćom *p*, moguće je dobiti sva tri potomka. Ako se radi o višepozicionom ukrštanju, onda ne možemo dobiti ni jednog potomka.

4.23 Dve jedinke-roditelja imaju reprezentacije 0011 i 1010. Da li se u nekom njihovom potomku (dobijenom ukrštanjem) može javiti:

(1) na prvoj poziciji (zdesna nalevo) vrednost 0 (***da***/ne)
(2) na prvoj poziciji (zdesna nalevo) vrednost 1 (***da***/ne)
(3) na drugoj poziciji (zdesna nalevo) vrednost 0 (da/***ne***)
(4) na drugoj poziciji (zdesna nalevo) vrednost 1 (***da***/ne)
(5) na trećoj poziciji (zdesna nalevo) vrednost 0 (***da***/ne)
(6) na trećoj poziciji (zdesna nalevo) vrednost 1 (da/***ne***)
(7) na četvrtoj poziciji (zdesna nalevo) vrednost 0 (***da***/ne)
(8) na četvrtoj poziciji (zdesna nalevo) vrednost 1 (***da***/ne)

4.25 U genetskim algoritmima, kolika je obično verovatnoća da neki bit neke jedinke mutira?

Obično je manja od 1%.

4.26 Da li se od jedinke 1010 mutacijom može dobiti jedinka:

(a) 0000
(b) 0011
(c) 1111
Zavisi kako je definisana mutacija, u jednom slučaju možemo dobiti sve tri date jedinke, dok u drugom ne možemo dobiti ni jednu.

4.27 Ako tokom primene genetskog algoritma ima *N* jedinki, svaka je predstavljena sa *M* bitova, a verovatnoća mutacije je *p*, koliki je očekivani broj mutiranih gena u jednoj generaciji?

*N*p*

4.28 Šta je to elitizam u genetskim algoritmima?

Elitizam - nekoliko najboljih jedinki u generaciji se štite od eliminisanja ili bilo kakvih izmena, i prenose se u sledeću generaciju.

4.29 Navesti bar četiri moguća uslova za zaustavljanje genetskog algoritma.

1. Pronađeno je rešenje koje zadovoljava unapred zadati kriterijum
2. Dostignut je zadati broj generacija
3. Funkcija prilagođenosti je izračunata zadati broj puta
4. Vrednost prilagođenosti najbolje jedinke se tokom određenog broja generacija nije popravila
5. Kombinacija nekoliko uslova

5. Automatsko rasuđivanje u iskaznoj logici

5.1 Da li nad konačnim skupom iskaznih promenljivih ima konačno ili prebrojivo ili neprebrojivo mnogo (sintaksički) različitih iskaznih formula?

Ima konačno mnogo različitih iskaznih formula.

5.2 Šta je literal u iskaznoj logici?

Literal je iskazna formula koja je ili atomička iskazna formula ili negacija atomičke iskazne formule. Atomička iskazna formula je ili promenljiva ili logička konstanta.

5.3 Ako za iskazne formule A i C važi A = C, čemu je jednako A[C→D]?

A = D.

5.4 Ako za iskazne formule A i C važi A ≠ C i A je atomička formula, čemu je jednako A[C→D]?

A = A.

5.5 Čemu je jednako (p ∧ (¬q ∨ r))[¬q ∨ r→ q ⇒ r]?

(p ∧ (¬q ∨ r))[¬q ∨ r → q ⇒ r] = (p ∧ (q ⇒ r)).

5.6 Kako se definiše interpretacija u iskaznoj logici?

Interpretacija je proširenje valuacije - svaka valuacija v određuje jednu funkciju I_v koju zovemo interpretacijom za valuaciju v i koja pridružuje (jedinstvene) istinitosne vrednosti formulama (tj. preslikava skup iskaznih formula u skup {0, 1}).

5.7 Kada je iskazna formula A ⇒ B tačna u interpretaciji I_v?

$I_v(A \Rightarrow B) = 1$, za sve slučajeve osim kada je $I_v(A) = 1$ i $I_v(B) = 0$

5.8 Kada je $I_v(A \Rightarrow B) = 0$?

$I_v(A \Rightarrow B) = 0$, ako je $I_v(A) = 1$ i $I_v(B) = 0$

5.9 Čemu je jednaka vrednost $I_v(A \Leftrightarrow B)$ za valuaciju v?

$I_v(A \Leftrightarrow B) = 1$, ako je $I_v(A) = I_v(B)$
= 0, inače

5.10 Ako iskazna formula ima barem jedan model, kakva je onda ona?

Iskazna formula je zadovoljiva.

5.11 Ako iskazna formula nema nijedan model, kakva je onda ona?

Iskazna formula je kontradikcija.

5.12 Ako iskazna formula nije poreciva, kakva je onda ona?

Iskazna formula je tada valjana.

5.13 Ako iskazna formula nije zadovoljiva, kakva je onda ona?

Iskazna formula je tada kontradikcija.

5.14 Ako iskazna formula nije kontradikcija, kakva je onda ona?

Zadovoljiva.

5.15 Ako je formula ¬F zadovoljiva, kakva je onda formula F?

F je tada poreciva.

5.16 Navesti primer iskazne formule koja je:

- *zadovoljiva*: p ∧ q
- *valjana*: p ∨ ¬p
- *poreciva*: p ∧ q
- *kontradikcija*: p ∧ ¬p
- *zadovoljiva i valjana*: p ∨ ¬p
- *zadovoljiva i nije valjana*: p ∧ q
- *zadovoljiva i poreciva*: p ∧ q
- *zadovoljiva i nije poreciva*: p ∨ ¬p
- *zadovoljiva i nije kontradikcija*: p ∧ q
- *valjana i nije poreciva*: p ∨ ¬p
- *valjana i nije kontradikcija*: p ∨ ¬p
- *poreciva i nije zadovoljiva*: p ∧ ¬p
- *poreciva i nije valjana*: p ∧ q
- *poreciva i kontradikcija*: p ∧ ¬p
- *poreciva i nije kontradikcija*: p ∧ q
- *kontradikcija i nije zadovoljiva*: p ∧ ¬p
- *kontradikcija i nije valjana*: p ∧ ¬p

5.17 Zakružiti tačan odgovor:

- Ako je iskazna formula valjana, da li je ona sigurno zadovoljiva? (***da***/ne)
- Ako je iskazna formula kontradikcija, da li je ona sigurno poreciva? (***da***/ne)
- Ako iskazna formula nije zadovoljiva, da li je ona sigurno kontradikcija? (***da***/ne)
- Ako iskazna formula nije tautologija, da li je ona sigurno poreciva? (***da***/ne)

5.18 Da li je formula (¬p ∧ p ∧ ¬r) ⇒ (¬q ∨ r) tautologija, zadovoljiva, poreciva ili nezadovoljiva?

p	q	r	¬p	¬q	¬r	(¬p ∧ p ∧ ¬r)	(¬q ∨ r)	A
1	1	1	0	0	0	0	1	1
1	1	0	0	0	1	0	0	1
1	0	1	0	1	0	0	1	1
1	0	0	0	1	1	0	1	1
0	1	1	1	0	0	0	1	1
0	1	0	1	0	1	0	0	1
0	0	1	1	1	0	0	1	1
0	0	0	1	1	1	0	1	1

Formula je zadovoljiva i tautologija.

5.19 Da li je formula (¬p ∨ q) ⇒ (¬q ∨ p) tautologija, zadovoljiva, poreciva ili nezadovoljiva?

p	q	¬p	¬q	¬p ∨ q	¬q ∨ p	A
1	1	0	0	1	1	1
1	0	0	1	0	1	1
0	1	1	0	1	0	0
0	0	1	1	1	1	1

Formula je poreciva i zadovoljiva.

5.20 Ako su iskazne formule A i A ⇒ B tautologije, da li je onda formula B tautologija, zadovoljiva, poreciva ili kontradikcija?

B je tada tautologija.

5.21 Kada za iskaznu formulu A kažemo da je logička posledica skupa formula Γ?

Ako je svaki model za skup iskaznih formula Γ istovremeno i model za iskaznu formulu A, onda se kaže da je A logička posledica skupa Γ i piše se Γ ⊨ A.

5.22 Da li nad konačnim skupom iskaznih promenljivih ima konačno ili prebrojivo ili neprebrojivo mnogo iskaznih formula od kojih nikoje dve nisu logički ekvivalentne?

Postoji konačno mnogo iskaznih formula.

5.23 Koliko ima klauza dužine *k* nad skupom od *n* iskaznih promenljivih

- ako je dozvoljeno da se u klauzi pojavljuje i literal i njegova negacija? $\binom{2*n}{k}$
- ako nije dozvoljeno da se u klauzi pojavljuje i literal i njegova negacija? $\binom{n}{k}$

(Smatra se da se u klauzi ne pojavljuju logičke konstante niti da se ponavlja isti literal, klauze se smatraju istim ako se razlikuju samo u poretku literala koje sadrže).

5.24 Kada kažemo da su iskazne formule A i B logički ekvivalentne?

Ako je svaki model iskazne formule A model i iskazne formule B i obratno (tj. ako važi {A} ⊨ B i {B} ⊨ A), onda se kaže se da su formule A i B logički ekvivalentne i piše se A ≡ B.

5.25 Kako se zapisuje da su formule A i B logički ekvivalentne?

A ≡ B.

5.26 Da li je A ≡ B formula ili meta-formula? Da li je A ⇔ B formula ili meta-formula? Kakva je veza između A ≡ B i A ⇔ B?

A ≡ B je meta-formula. A ⇔ B je formula iskazne logike. A ≡ B ako i samo ako je A ⇔ B tautologija.

5.27 Navesti teoremu o zameni za iskaznu logiku.

Ako je C ≡ D, onda je A[C → D] ≡ A.

5.28 Ako važi $\Gamma \models A$, kakav treba da bude odnos između skupova Γ i Δ da bi važilo i $\Delta \models A$?

$\Gamma \subset \Delta$.

5.29 Da li je u iskaznoj logici odlučiv problem ispitivanja:

- zadovoljivosti (***da***/ne)
- porecivosti (***da***/ne)
- valjanosti (***da***/ne)
- kontradiktornosti (***da***/ne)

5.30 Da li je za iskaznu formulu jednoznačno određena njena konjunktivna normalna forma?

Ne.

5.31 Navesti jedan algoritam za transformiranje iskazne formule u KNF.

```
Ulaz: iskazna formula F
Izlaz: konjuktivna normalna forma formule F
   1. dok god je to moguće radi
   2.     eliminiši veznik ⇔: A ⇔ B ≡ (A ⇒ B) ∧ (B ⇒ A).
   3. dok god je to moguće radi
   4.     eliminiši veznik ⇒: A ⇒ B ≡ ¬A ∨ B.
   5. dok god je to moguće radi
   6.     primeni neku od logičkih ekvivalencija: ¬(A∧B) ≡ ¬A∨¬B
                                                  ¬(A∨B) ≡ ¬A∧¬B.
   7. dok god je to moguće radi
   8.     eliminiši višestruke veznike ¬: ¬¬A ≡ A.
   9. dok god je moguće radi
   10.    primeni neku od logičkih ekvivalencija:
             A ∨ (B ∧ C) ≡ (A ∨ B) ∧ (A ∨ C)
             (B ∧ C) ∨ A ≡ (B ∨ A) ∧ (C ∨ A).
```

5.32 Šta tokom primene algoritma KNF važi nakon primene logičke ekvivalencije $\neg\neg A \equiv A$?

U tekućoj formuli postoje jedino logicki veznici $\wedge$, $\vee$ i $\neg$, pri čemu uz svaku iskaznu promenljivu stoji najviše jedna negacija.

5.33 Navesti teoremu o korektnosti algoritma KNF za iskaznu logiku.

Algoritam KNF se zaustavlja i ima sledeće svojstvo: ako je F ulazna formula, onda je izlazna formula F` u konjuktivnoj normalnoj formi i logički je ekvivalentna sa F.

5.34 Zašto se zaustavlja prvi korak algoritma KNF?

Formula sadrži konačan broj ekvivalencija. Navedena transformacija eliminiše svaku ekvivalenciju u samoj formuli.

5.35 Zašto se zaustavlja četvrti korak algoritma KNF?

Formula sadrži konačan broj višestrukih negacija. Navedena transformacija eliminiše svaku višestruku negaciju u samoj formuli.

5.36 Da li se može konstruisati iskazna formula za koju se algoritam KNF ne zaustavlja?
Ne.

5.37 Kako se definišu binarni veznici ↑ i ↓?
- ↓ (nili ili Lukašijevičeva funkcija): $A \downarrow B = \neg(A \vee B)$
- ↑ (ni ili Šeferova funkcija): $A \uparrow B = \neg(A \wedge B)$

5.38 Koliko ima binarnih veznika koji pojedinačno čine potpun skup veznika za iskaznu logiku?
Postoje samo 2, ↑ i ↓.

5.39 Kako se zove problem ispitivanja zadovoljivosti iskazne formule u KNF obliku? Da li je ovaj problem odlučiv?
SAT problem. Odlučiv je.

5.40 Da li problem SAT pripada klasi P? Da li problem SAT pripada klasi NP? Da li je problem SAT NP-kompletan? Da li je problem SAT np-težak?
Ne znamo da li problem SAT pripada klasi P. Problem SAT je NP-kompletan. Svaki NP-kompletan problem je i NP-težak, dakle SAT je NP-težak.

5.41 U kom obliku mora da bude formula na koju se primenjuje DPLL procedura?
U KNF-u.

5.42 Koji odgovor vraća DPLL procedura ako ulazna formula ne sadrži nijednu klauzu?
Skup klauza je tada zadovoljiv.

5.43 Kako glasi pravilo tautology procedure DPLL?
Ako neka klauza C_i sadrži $\top$ ili neki literal i njegovu negaciju, onda vrati vrednost koju vraća $DPLL(D \setminus C_i)$.

5.44 Kako glasi pravilo split DPLL procedure?
Ako $DPLL(D[p \rightarrow \top])$ (gde je p jedno od iskaznih slova koja se javljaju u D) vraća DA onda
    vrati DA;
Inače
    vrati vrednost koju vraća $DPLL(D[p \rightarrow \bot])$.

5.45 Koja su pravila DPLL procedure primenljiva na formulu: $(\neg a \vee b \vee c) \wedge (a \vee b \vee \neg c) \wedge (\neg a \vee b \vee \neg c)$?
Jedino može da se primeni *split* pravilo.

5.46 Da li se može konstruisati iskazna formula u KNF formi za koju se algoritam DPLL ne zaustavlja?
Ne.

5.47 Koja je složenost DPLL procedure u najgorem slučaju?
Eksponencijalne vremenske složenosti po broju iskaznih promenljivih u formuli.

5.48 Da li postoje iskazne formule za koje je vreme izvršavanja procedure DPLL polinomsko u odnosu na veličinu formule?

Da, ako se ne koristi pravilo split.

5.49 Ako želimo da DPLL procedurom ispitamo da li je iskazna formula A tautologija, šta treba da bude ulaz za DPLL proceduru? U kom slučaju je onda formula A valjana?

Ulaz za DPLL je ¬A i formula A je valjana ako u tom slučaju izlaz Ne.

6. Automatsko rasuđivanje u logici prvog reda

6.1 Kako se još nazivaju funkcijski simboli arnosti 0?
Konstante.

6.2 Šta je term u logici prvog reda?
Za signaturu *L:* promenljive i funkcijski simboli arnosti 0 su termovi. Term je i objekat dobijen primenom funkcijskog simbola *f* arnosti *n* na termove $t_1,..., t_n$.

6.3 Šta je literal u logici prvog reda?
Literal je atomička formula ili negacija atomičke formule. Atomička formula je logička konstanta ili objekat dobijen primenom predikatskog simbola *p* arnosti *n* na termove $t_1,..., t_n$

6.4 Šta je klauza u logici prvog reda?
Klauza je disjunkcija više literala.

6.5 Da li je u formuli ∀x(p(x, y) ∧ q(y, z) ∧ r(z)), promenljiva x slobodna ili vezana, da li je promenljiva y slobodna ili vezana, da li je promenljiva z slobodna ili vezana?
x je vezana, y je slobodnna i z je slobodna.

6.6 Za datu signaturu *L*, šta je to *L*-struktura *D*?
Za datu signaturu *L*, *L*-struktura *D* je par (D, I), gde je D skup, a I funkcija i važi:

- D je neprazan skup i zovemo ga univerzum (ponekad i domen ili nosač)
- svakom simbolu konstante c iz *L* (tj. svakom funkcijskom simbolu arnosti 0), funkcija I pridružuje jedan element c_I iz D
- svakom funkcijskom simbolu *f* iz *L* arnosti *n*, n > 0, funkcija I pridružuje jednu totalnu funkciju f_I iz D^n u D
- svakom predikatskom simbolu *p* iz *L* arnosti *n*, n > 0, funkcija I pridružuje jednu relaciju, tj. totalnu funkciju p_I iz D^n u {0, 1}.

6.7 U šta se, u svakoj interpretaciji logike prvog reda, preslikava funkcijski simbol?
Funkcijski simbol se preslikava u totalnu funkciju $f_I : D^n \rightarrow D$, gde je *n* arnost funkcijskog simbola *f,* a D je domen koji se nalazi u *L*-strukturi *D*.

6.8 U šta se, u svakoj interpretaciji logike prvog reda, preslikava predikatski simbol?
Predikatski simbol se preslikava u totalnu funkciju $p_I : D^n \rightarrow \{0, 1\}$, gde je *n* arnost predikatskog simbola *p,* a D je domen koji se nalazi u *L*-strukturi *D*.

6.9 U semantici logike prvog reda, ako je x promenljiva, čemu je jednako I_v(x)?
Interpretacija promenljive x je jednaka valuaciji promenljive x.

6.10 U logici prvog reda, čemu je, za neku interpretaciju I_v, jednaka vrednost $I_v(\forall x A)$?
$I_v((\forall x)A) = 1$, ako za svaku valuaciju *w* takvu da je $w \sim_x v$, važi $I_w(A) = 1$
$\quad = 0$, inače

6.11 U logici prvog reda, čemu je, za neku interpretaciju I_v, jednaka vrednost $I_v(\exists xA)$?

$I_v((\exists x)A) = 1$, ako postoji valuacija w takva da je $w \sim_x v$, važi $I_w(A) = 1$
$= 0$, inače

6.12 Ako, u logici prvog reda, za dve valuacije *v* i *w* važi v(x) = 1, v(y) = 2, w(x) = 3 i $v \sim_x w$, šta važi za w(y)?

$v(y) = w(y) \rightarrow w(y) = 2$.

6.13 Da li je problem zadovoljivosti u logici prvog reda odlučiv ili poluodlučiv ili neodlučiv?

Neodlučiv.

6.14 Da li je problem valjanosti u logici prvog reda odlučiv ili poluodlučiv ili neodlučiv?

Poluodlučiv.

6.15 Kada kažemo da su formule logike prvog reda A i B logički ekvivalentne?

Kažemo da su formule A i B nad istom signaturom *L* logički ekvivalentne i pišemo $A \equiv B$ ako važi $\{A\} \models B$ i $\{B\} \models A$.

6.16 Da li je formula $(\forall x)(A \wedge B)$ logički ekvivalentna nekim od formula:

- $(\forall x)A \wedge (\forall x)B$ (***da***/ne)
- $(\forall x)A \wedge B$ (da/ne) - da, ako se u B ne nalazi promenljiva x; inače, ne.
- $(\forall x)A \vee (\forall x)B$ (da/***ne***)
- $(\forall x)A \vee B$ (da/***ne***)

6.17 Da li su formule $((\forall xA) \wedge B)$ i $(\forall x(A \wedge B))$ logički ekvivalentne?

Jesu samo ako se u B ne nalazi promenljiva x.

6.18 Da li su formule $((\forall xA) \wedge \forall xB)$ i $(\forall xA \wedge B)$ logički ekvivalentne?

Jesu samo ako se u B ne nalazi promenljiva x.

6.19 Šta treba da važi za promenljivu x da formule $(\forall x(A \wedge B))$ i $(\forall xA \wedge B)$ nisu nužno logički ekvivalentne?

Da se ona nalazi u B.

6.20 Navesti teoremu o zameni za logiku prvog reda. Gde se ona koristi?

Ako važi $B_1 \equiv B_2$, onda je $A \equiv A[B_1 \rightarrow B_2]$. Koristi se u svođenju proizvoljne formule logike prvog reda u Prenex normalnu formu.

6.21 Dokazati da je formula dobijena algoritmom PRENEX logički ekvivalentna polaznoj formuli.

U svakom koraku, tekuća formula se transformiše dalje koristeći pravila koja čuvaju logičku ekvivalentnost. Pošto je ulazna formula konačna, i svaki korak algoritma Prenex se završava, sledi da je formula dobijena algoritmom Prenex logički ekvivalentna početnoj formuli.

6.22 Navesti algoritam PRENEX.

Ulaz: Zatvorena formula logike prvog reda
Izlaz: Preneks normalna forma zadate formule

1. dok god je to moguće radi
2. primeni neku od logičkih ekvivalencija:
   $$A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A),$$
   $$A \Rightarrow B \equiv \neg A \vee B;$$
3. dok god je to moguće radi
4. primeni neku od logičkih ekvivalencija:
   $$\neg(A \wedge B) \equiv \neg A \vee \neg B,$$
   $$\neg(A \vee B) \equiv \neg A \wedge \neg B,$$
   $$\neg(\forall x)A \equiv (\exists x)\neg A,$$
   $$\neg(\exists x)A \equiv (\forall x)\neg A;$$
5. dok god je to moguće radi
6. primeni neku od logičkih ekvivalencija (eliminiši višestruke veznike koristeći zakon dvojne negacije): $\neg\neg A \equiv A$;
7. dok god je to moguće radi
8. primeni neku od logičkih ekvivalencija:
   $$(\forall x A) \wedge B \equiv (\forall x)(A \wedge B),$$
   $$(\forall x A) \vee B \equiv (\forall x)(A \vee B),$$
   $$B \wedge (\forall x)A \equiv (\forall x)(B \wedge A)$$
   $$B \vee (\forall x)A \equiv (\forall x)(B \vee A),$$
   $$(\exists x A) \wedge B \equiv (\exists x)(A \wedge B),$$
   $$(\exists x A) \vee B \equiv (\exists x)(A \vee B),$$
   $$B \wedge (\exists x)A \equiv (\exists x)(B \wedge A),$$
   $$B \vee (\exists x)A \equiv (\exists x)(B \vee A),$$
   pri čemu x nema slobodna pojavljivanja u formuli B; ako x ima slobodna pojavljivanja u B, onda treba najpre preimenovati promenljivu x u formuli $(\forall x)A$ (odnosno u formuli $(\exists x)A$).

6.23 Kako se zove postupak kojim se formula prvog reda transformiše u formulu bez kvantifikatora?

Skolemizacija.

6.24 Navesti teoremu o skolemizaciji.

Ako je formula *B* nad signaturom *L'* dobijena skolemizacijom od rečenice *A* nad signaturom *L* koja je u preneks normalnoj formi, onda je *A* zadovoljiva akko je *B* zadovoljiva.

6.25 Ako je formula B dobijena od formule A skolemizacijom, kakav odnos važi za ove dve formule?

B je zadovoljivo akko je A zadoovoljivo. One su slabo ekvivalentne.

6.26 Zašto formula A i formula dobijena od nje skolemizacijom nisu logički ekvivalentne?

Zato što se menja signatura početne formule uvođenjem novih funkcijskih simbola.

6.27 Kada za dve formule A i B logike prvog reda kažemo da su slabo ekvivalentne?

Kada je A zadovoljivo akko je i B zadovoljivo.

6.28 Primenom koja tri koraka se dobija klauzalna forma formule A?

1. Transformisanje formule u preneks normalnu formu
2. Transformiranje dela formule bez kvantifikatora u konjuktivnu normalnu formu
3. Skolemizacija

6.29 U kakvom su odnosu formula A i njena klauzalna forma?

One su slabo ekvivalentne.

6.30 Ako je za neka dva izraza σ neki unifikator, a λ najopštiji unifikator, kakav onda postoji unifikator μ?

$\mu = \lambda\sigma$.

6.31 Do na šta dva izraza imaju jedinstven najopštiji unifikator?

Do na preimenovanje promenljivih.

6.32 Kako glasi pravilo cycle algoritma Najopštiji unifikator?

Ako postoji par (x, t) takav da je *x* promenljiva i *t* term koji sadrži *x* onda
zaustavi rad i kao rezultat vrati *neuspeh*;

6.33 U kojim koracima algoritam Najopštiji unifikator može da vrati neuspeh?

U koracima *collision* i *cycle*.

6.34 U kom slučaju je primenljivo pravilo *decomposition* u algoritmu Najopštiji unifikator?

Ako postoji par (s, t), gde ni *s* ni *t* nisu promenljive i ako je *s* jednako $\varphi(u_1, u_2,..., u_k)$ i *t* je jednako $\varphi(v_1, v_2,..., v_k)$ (gde je φ funkcijski ili predikatski simbol).

6.35 Ako dva izraza nisu unifikabilna, da li je moguće da se algoritam Najopštiji unifikator zaustavi sa uspehom?

Ne, vraća se *neuspeh*.

6.36 Ako dva izraza nisu unifikabilna, da li je moguće da se algoritam Najopštiji unifikator zaustavi sa neuspehom?

Da.

6.37 Ako dva izraza nisu unifikabilna, da li je moguće da se algoritam Najopštiji unifikator ne zaustavi?

Ne. Ovo je odlučiv problem.

6.38 Da li algoritam Najopštiji unifikator pripada klasi P? Zašto?

Pripada jer postoje algoritmi koji u polinomijalnom vremenu pronalaze najopštiji unifikator.

6.39 Navesti algoritam Najopštiji unifikator.

Ulaz: Niz parova izraza (s_1, t_1), (s_2, t_2),..., (s_n, t_n)

Izlaz: Najopštiji unifikator (ako on postoji) σ takav da važi $s_1\sigma = t_1\sigma$, $s_2\sigma = t_2\sigma$,..., $s_n\sigma = t_n\sigma$.

dok god je moguće primeniti neko od navedenih pravila radi
  {Korak factoring}
  ako postoji par koji ima više od jednog pojavljivanja onda
    obriši sva njegova pojavljivanja osim jednog.
  {Korak tautology}
  ako postoji par (t, t) onda
    obriši ga.
  {Korak orientation}
  ako postoji par (t, x), gde je x promenljiva, a t nije promenljiva onda
    zameni par (t, x) parom (x, t).
  ako postoji par (s, t) gde ni s ni t nisu promenljive onda
    ako je s jednako $\varphi(u_1, u_2, \ldots, u_k)$ i t je jednako $\varphi(v_1, v_2, \ldots, v_k)$ onda
      {Korak decomposition}
      dodaj parove $(u_1, v_1), \ldots, (u_k, v_k)$ i obriši par (s,t)
    inače
      {Korak collision}
      zaustavi rad i kao rezultat vrati *neuspeh*;
  {Korak cycle}
  ako postoji par (x, t) takav da je x promenljiva i t term koji sadrži x onda
    zaustavi rad i kao rezultat vrati *neuspeh*;
  {Korak application}
  ako postoji (x, t), gde je x promenljiva a t term koji ne sadrži x i x se pojavljuje i u nekim drugim parovima onda
    primeni supstituciju $[x \to t]$ na sve druge parove.
vrati tekući skup parova kao najopštiji unifikator.

6.40 Koje korake je potrebno primeniti da bi se metodom rezolucije ispitalo da li je formula logike prvog reda A valjana?

Najpre se formula mora prevesti u Prenex oblik. Potom se deo formula iza kvantifikatora transformiše u KNF, i konačno, eliminišu se egzistencijalni kvantifikatori skolemizacijom.

6.41 Da bi se primenio metod rezolucije u kakvoj formi formula čija se nezadovoljivost ispituje mora da bude?

U klauzalnoj formi (KNF).

6.42 Šta je rezolventa klauza Γ′ ∨ A′ i Γ″ ∨ ¬A″ (σ je najopštiji unifikator za A′ i A″)?

(Γ\` ∨ Γ\`\`)σ

6.43 Navesti pravilo rezolucije za logiku prvog reda.

Pravilo rezolucije za logiku prvog reda (u njegovom osnovnom obliku) može se prikazati na sledeći način: $\dfrac{\Gamma' \vee A', \Gamma'' \vee \neg A''}{(\Gamma' \vee \Gamma'')\sigma}$

6.44 Koji su mogući ishodi primene metoda rezolucije za iskaznu logiku, a koji za logiku prvog reda?

Primena metoda rezolucije nad formulama iskazne logike nam garantuje da će se algoritam završiti i da će se izvesti prazna klauza akko je polazna formula nezadovoljiva. Pri prelasku na logiku prvog reda problem postaje poluodlučiv. Ako je ulazna formula nezadovoljiva, onda se metodom rezolucije može izvesti prazna klauza.

6.45 Da li se metod rezolucije za iskaznu logiku i metod rezolucije za logiku prvog reda uvek zaustavljaju?

U iskaznoj logici: Da. U logici prvog reda: Ne.

6.46 Da li se metodom rezolucije za svaku formulu logike prvog reda koja je valjana može dokazati da je valjana?

Da.

6.47 Da li se metodom rezolucije za svaku formulu logike prvog reda koja nije valjana može dokazati da nije valjana?

Ne, poluodlučiv problem.

6.48 U iskaznoj logici, da li će u konačnom broju koraka biti izvedena prazna klauza, ma kako god se primenjivalo pravilo rezolucije

- ako je početni skup klauza zadovoljiv? (da/***ne***)
- ako je početni skup klauza nezadovoljiv? (***da***/ne)

6.49 Ukoliko je skup klauza logike prvog reda nezadovoljiv, onda se iz njega metodom rezolucije:

(a) uvek mora izvesti prazna klauza; (b) uvek može izvesti prazna klauza;
(c) ne može izvesti prazna klauza; (d) nikad ne može izvesti prazna klauza.