GLAVA 1: RESAVANJA PROBLEMA KORISCENJEM PRETRAGE

BITNO
PITANJE:
Sta je kombinatorna eksplozija?
Odgovor:
Kombinatorna eksplozija je pojava koja se javlja u problemima i cini da njihovo resavanje zahteva razmatranje ogromnog broja mogucnosti.

Pitanje 1.1.
Navesti barem 5 opstih elemenata svakog problema pretrage.
Odgovor:
Polazno stanje, skup mogucih stanja, skup mogucih akcija, cena akcije, funkcija prelaska, test cilja.

Pitanje 1.2.
Kako se, prema dostupnosti informacija koje mogu pomoci u pronalazenju ciljnog stanja u toku pretrage, dele problemi pretrage?
Odgovor:
Dele se na probleme informisane i neinformisane pretrage.

GLAVA 2 NEMA SEKCIJU SA PITANJIMA, ALI NE ZNACI DA NECE DOCI – PROCITATI

GLAVA 3: INFORMISANA PRETRAGA

Pitanje 3.1.
Kako se naziva algoritam pretrage koji uvek bira lokalno optimalne akcije?
Odgovor:
Algoritam pohlepne pretrage.

Pitanje 3.2.
Sta, umesto globalnog ekstremuma, pohlepna pretraga moze da vrati kao rezultat?
Odgovor:
Lokalni maksimum.

Pitanje 3.3.
Sta je plato u problemima pretrage?
Odgovor:
Plato je oblast prostora pretrage u kome funkcija cilja ima const. vrednost.

Pitanje 3.5.
Cemu je jednaka vrednost f(n) koja se u algoritmu A* pridruzuje cvoru n?
Odgovor:
f(n) predstavlja funkciju evaluacije za cvor n; f(n)=g(n)+h(n), gde je g(n) cena puta od pocetnog cvora do n, a h(n) heuristicka procena od cvora n do cilja.

Pitanje 3.6.
Sta, za razliku od Dijkstrinog algoritma, algoritam A* uzima u obzir?
Odgovor:
A* uzima u obzir heuristicku funkciju h(n), koja se koristi za usmeravanje pretrage.

Pitanje 3.7.
Da li je algoritam A* opstiji od Dijkstrinog algoritma? Da li je Dijkstrin algoritam opstiji od A*?
Odgovor:
Dijkstrin se moze dobiti od A*, kad je h(n)=0, stoga je A opstiji.

Pitanje 3.9.
Da li se tokom primene algoritma A* moze primeniti vrednost g(n) za cvor n?
Da li se tokom primene algoritma A* moze primeniti vrednost h(n) za cvor n?
Da li se tokom primene algoritma A* moze primeniti vrednost f(n) za cvor n?
Odgovor:
Moze.
Ne moze.
Moze (indirektno), jer f(n)=g(n)+h(n).

Pitanje 3.10.
Kako se zove skup iz kojeg se u glavnoj petlji algoritma A* bira tekuci cvor?
Odgovor:
Zove se otvorena lista.

Pitanje 3.11.
Kakva struktura se koristi za cuvanje vrednosti funkcije evaluacije u okviru algoritma A*? Obrazloziti.
Odgovor:
Koristi se ili hes-mapa ili min-hip (?). Verovatno se koristi da bi se u konstantnom vremenu izmenila ili procitala neka vrednost funkcije evaluacije za cvor.

Pitanje 3.12.
Da li je na samom pocetku primene algoritma A* lista zatvorenih cvorova prazna?
Odgovor:
Da.

Pitanje 3.13.
Koji cvor se, prilikom primene algoritma A*, prvi dodaje u listu otvorenih cvorova?
Odgovor:
Polazni cvor.

Pitanje 3.14.
Kada se, u okviru algoritma A*, u listu zatvorenih cvorova dodaje novi element?
Odgovor:
Kada obradimo sve susede tekuceg cvora.

Pitanje 3.15.
Tokom primene algoritma A*, ako se ispituje tekuci cvor i naidje na njegov susedni cvor v koji nije u zatvorenoj listi, ali jeste u otvorenoj, sta treba uciniti?
Odgovor:
Treba proveriti da li je put od polaznog cvora do cvora v preko tekuceg cvora bolji (kraci/jeftiniji) od poslednjeg puta do v (trenutna vrednost g(v)). Ako jeste, treba promeniti informaciju o roditelju cvora v na tekuci cvor i azurirati vrednost f(v).

Pitanje 3.16.
Da li je na kraju primene algoritma A* lista otvorenih cvorova nuzno prazna?
Odgovor:
Ne.

Pitanje 3.17.
Da li je na kraju primene algoritma A* lista zatvorenih cvorova nuzno prazna?
Odgovor:
Ne.

Pitanje 3.18.
Sta je uslov zaustavljanja algoritma A*?
Odgovor:
Uslov je da je broj stanja konacan.

Pitanje 3.19.
Za koje grafove je algoritam A* najpogodniji za primenu?
Odgovor:
Za retke grafove; kad graf ima manji broj grana, A* ce brze izvrsiti obradu cvorova.

Pitanje 3.20.
Kada kazemo da je funkcija heuristike h u algoritmu A* dopustiva, a kad da je konzistentna?
Odgovor:
Dopustiva -> $0 <= h(n) <= h^*(n)$.
Konzistentna -> $c(n,m) + h(m) >= h(n)$.

Pitanje 3.21.
Sta znaci da je algoritam A* potpun?
Odgovor:
Znaci da se zavrsava i da daje NEKO resenje (ne mora biti optimalno).

Pitanje 3.22.
Pod kojim uslovom je algoritam A* potpun?
Odgovor:
Pod uslovima da je broj cvorova i broj akcija konacan, i da postoji neki put izmedju pocetnog cvora i cilja.

Pitanje 3.23.
Pod kojim uslovom je algoritam A* optimalan (nalazi najkraci put)?
Odgovor:
Pod uslovom da je heuristika dopustiva.

Pitanje 3.24.
Ukoliko je f* funkcija koja odgovara optimalnom putu izmedju dva cvora u grafu, koje cvorove obradjuje algoritam A*?
Odgovor:
U ovom slucaju nije nam ni potreban algoritam A*. Medjutim, ako bas zelimo da ga primenimo, koristimo f* kao heuristiku (f* je dopustiva jer 0 >= f*(n) >= f*(n)).

Pitanje 3.25.
Kada je algoritmu A* broj obradjenih cvorova polinomijalan?
Odgovor:
Kada je heuristika kvalitetna, odnosno ako zadovoljava: |h(x)-h*(x)| <= O(log h*(x)).

Pitanje 3.26.
Kako se zove rastojanje izmedju dva cvora u kojem se broji ukupan broj polja predjenih horizontalno ili vertikalno od prvog do drugog?
Odgovor:
Menhetn rastojanje.

Pitanje 3.27.
Koliko je Menhetn rastojanje izmedju donjeg levog i gornjeg desnog polja sahovske table?
Odgovor:
14.

Pitanje 3.28.
Kada se algoritam A* primenjuje na uniformnoj mrezi, koja funkcija se obicno primenjuje kao heuristika? Da li je ova heuristika dopustiva? Zasto se primenjuje ova heuristika, sta je njena kljucna osobina?
Odgovor:
Koristi se Menhetn rastojanje, a omogucava se i kretanje dijagonalno. Kretanje dijagonalno penalizujemo sa 1,41, a horizontalno i vertikalno sa 1. Cesto se navedene vrednosti mnoze sa, npr, 10^n, kako bi se uprostio racun. Ukoliko se koristi samo Menhetn rastojanje, heuristika je dopustiva, ali kada se koriste modifikacije, onda nije jer moze da preceni stvarno rastojanje. U praksi, pristup sa modifikacijama daje zadovoljavajuce i brze rezultate.

Pitanje 3.29.
Kada se algoritam A* primenjuje na uniformnoj mrezi, sta se obicno koristi kao cena puta od susednog cvora koji je desno, a sta od onog gore-desno?
Odgovor:
1 za desno, 1.41 za gore desno. Cesto se mnozi sa 10^n ta vrednost, radi lakseg racuna.

GLAVA 4: IGRANJE STRATESKIH IGRANA – CITAJ IZ KNJIGE!

GLAVA 5: GENETSKI ALGORITMI

Pitanje 5.1.
Navesti opšti genetski algoritam.
Odgovor:
1. Generisati pocetnu populaciju (generaciju) jedinki
2. Izracunati funkciju prilagodjenosti za svaku
3. Izvrsavaj petlju sve dok ne bude zadovoljen uslov zaustavljanja
    * Izaberi skup jedinki za reprodukciju
    * Primenom operatora ukrsatanja kreirati nove jedinke
        i izracunati njihovu prilagodjenost
    * Na osnovu starih i novih jedinki, kreiraj novu generaciju
4. Vrati najkvalitetniju jedinku u poslednjoj populaciji

Pitanje 5.2.
Da li, u genetskim algoritmima, ciljna funkcija mora da bude:
Odgovor:
* definisana za sve moguće jedinke? -> DA
* diskretna? -> NE
* neprekidna? -> NE
* diferencijabilna? -> NE

Pitanje 5.3.
Da li, u genetskim algoritmima, funkcija prilagodenosti mora da bude:
Odgovor:
* definisana za sve moguće jedinke? -> DA
* diskretna? -> NE
* neprekidna? -> NE
* diferencijabilna? -> NE

Pitanje 5.4.
Ukoliko je genetskim algoritmom potrebno odrediti minimum pozitivne funkcije f na nekom intervalu, onda je pogodno kao funkciju prilagodenosti koristiti funkciju:
(a) f ;
(b) −f
(c) inverznu funkciju of f ;
(d) f '
Odgovor:
-f.

Pitanje 5.5.
Ukoliko je genetskim algoritmom potrebno odrediti minimum pozitivne funkcije f na nekom intervalu, koju je funkciju koristiti kao funkciju prilagodenosti?
Odgovor:
-f.

Pitanje 5.6.
U genetskim algoritmima, koja se reprezentacija jedinki najčešće koristi?
Odgovor:
Pomocu bitova.

Pitanje 5.7.
Broj mogućih rešenja datog problema je 1000000. Ukoliko se za rešavanje ovog problema koristi genetski algoritam i binarna reprezentacija, onda je dužina hromozoma koji se koriste?
Odgovor:
1 000 000 -> milion :D mrzim slepljene duge brojeve
20 -> 20 bitova moze da reprezentuje broj milion.

Pitanje 5.8.
Ako je za potrebe primene genetskog algoritma, domen {3, 4, 5, 6, 7, 8, 9, 10} reprezentovan sa binarnim hromozomima dužine 3 (u istom poretku), kako će biti reprezentovana jedinka 9?
Odgovor:
Ne razumem bas pitanje...

Pitanje 5.9.
Kako se generiše inicijalna populacija u genetskim algoritmima?
Odgovor:
Nekad slucajno, nekad je dobijena nekim drugim (genetskim ili optimizacionim) algoritmom.

Pitanje 5.10.
Navesti dva genetska operatora.
Odgovor:
Mutacija, ukrstanje.

Pitanje 5.11.
Koliko genetski operatori ukrštanja i mutacije imaju ulaznih jedinki?
Odgovor:
Ukrstanje ime dve jedinke.
Mutacija se primenjuje nad jednom jedinkom.

Pitanje 5.12.
Šta je uloga selekcije u genetskim algoritmima?

Pitanje 5.13.
Najpopularnije vrste selekcije u genetskim algoritmima su:
(a) Menhetn i ruletska;
(b) Menhetn i turnirska;
(c) ruletska i turnirska;
(d) ruletska i uniformna;
Odgovor:
(c) ruletska i turnirska;

Pitanje 5.14.
Koje vrste selekcija se najčešće koriste u genetskim algoritmima?
Odgovor:
Turnirska i ruletska

Pitanje 5.15.
Kako se jedinka bira ruletskom selekcijom?
Odgovor:
Neka imamo n jedinki.
p(i) -> Verovatnoca da bude izabrana jedinka i
f(i) -> Funkcija prilagodjenosti

p(i) = f(i) / (suma svih f(j)), j = 1, n

Izracuna se p(i) za svako i = 1, n
Za svaku jedinku sada imamo verovatnocu kojom je biramo, te se generise slucajan broj i izvrsi biranje.
Zove se ruletska jer podseca na rulet. lol. Verovatnoce koje dobiju jedinke mogu da cine udeo ruleta, a biranje se svodi na igranje runde ruleta. U zavisnosti od dela ruleta gde se zaustavi kuglica, bira se ta jedinka.

Pitanje 5.16.
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 pi = f(i)/x , gde je x jednako:
(a) 1;
(b) suma svih j do N: f(j)
(c) suma svih j do N, i!=j: f(j)
(d) proizvod svih j do N:f(j)
Odgovor:
(b) suma svih j do N: f(j)

Pitanje 5.17.
Ukoliko su vrednosti prilagodenosti jedinka a, b i c 2, 5, 8 redom, koja je verovatnoća da će u ruletskoj selekciji biti izabrana jedinka b?
Odgovor:
a: 2
b: 5
c: 8

S = 2+5+8 = 15

p(b) = f(b)/S
p(b) = 5/15 = 1/3

Pitanje 5.18.
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?
Odgovor:
Trazimo maksimum f(x) = 20 - x^2
1 : 18
-4: 4
2 : 16
3 : 11
S = 18+4+16+11=49

p(1) = 18/49 = 0.3673
p(-4) = 4/49 = 0.0816
p(2) = 16/49 = 0.3265
p(3) = 11/49 = 0.2245

Pitanje 5.19.
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 udje u proces reprodukcije?
Odgovor:
A: 1
B: 2
C: 3

S = 1+2+3 = 6

p(B) = f(B)/S
p(B) = 2/6 = 1/3

Pitanje 5.20.
Opisati algoritam turnirske selekcije.
Odgovor:
Parametri: k, p
k -> velicina turnira
p -> verovatnoca sa kojom se bira

Biramo k jedinki, sortiramo ih po funkciji prilagodjenosti.
Verovatnoca da izaberemo i-tu jedinku je p(1-p)^(i-1)

Pitanje 5.21.
Ako je u turnirskoj selekciji veličina turnira k jednaka 1, čemu je ona ekvivalentna?
Odgovor:
Ruletskoj.

Pitanje 5.22.
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;
Odgovor:
Pomalo tautologicno pitanje jer ukoliko se koristi uniformno ukstanje sa verovatnocom p,
a bitovi koji se ukrstaju su komplementarni, moze se desiti da se dobije bilo koje od ponudjenog.

Pitanje 5.23.
U genetskim algoritmima, dve jedinke-roditelja imaju reprezentacije 0011 i 1010. Da li se u nekom njihovom potomku (dobijenom ukrštanjem) može javiti:
Odgovor:
(1) na prvoj poziciji (zdesna nalevo) vrednost 0 (da/ne); da
(2) na prvoj poziciji (zdesna nalevo) vrednost 1 (da/ne); da
(3) na drugoj poziciji (zdesna nalevo) vrednost 0 (da/ne); da
(4) na drugoj poziciji (zdesna nalevo) vrednost 1 (da/ne); da
(5) na trećoj poziciji (zdesna nalevo) vrednost 0 (da/ne); da
(6) na trećoj poziciji (zdesna nalevo) vrednost 1 (da/ne); ne (oba roditelja imaju 0 na ovoj poziciji)
(7) na četvrtoj poziciji (zdesna nalevo) vrednost 0 (da/ne); da
(8) na četvrtoj poziciji (zdesna nalevo) vrednost 1 (da/ne). da

Bitovi na pozicijama 1, 2 i 4 su komplementni te se uvek mogu dobiti uniformnim ukrstanjem.

0011
1010
x

Pomalo tautologicno pitanje jer ukoliko se koristi uniformno ukstanje sa verovatnocom p,
moze se desiti da se dobije bilo koje od ponudjenog.
Ostavljam citaocu da se pogira sa k-pozicionim ukrstanjem.

Pitanje 5.24.
Opisati uniformno ukrštanje koje se koristi u genetskim algoritmima.
Odgovor:
Ucestvuju roditelj R1, roditelj R2, a dobijaju se dete D1 i dete D2.
Parametar ukrstanja je verovatnoca p.
D1 dobija bit od R1 sa verovatnocom p, a za isti bit konkurise D2 sa verovatnocom (1-p).
Od R2 dobija bit Di koji nije dobio bit od R1.

Pitanje 5.25.
U genetskim algoritmima, kolika je obično verovatnoća da neki bit neke jedinke mutira?
Odgovor:
Izuzetno mala i eksperimentalno se utvrdjuje. Npr 0.1% - 1%

Pitanje 5.26.
Da li se od jedinke 1010 mutacijom može dobiti jedinka:
(a) 0000;
(b) 0011;
(c) 1111.
Odgovor:
Opet filozofsko pitanje.

(1): Ukoliko definisemo mutaciju da svaki bit jedinke moze da promeni sa verovatnocom p,
onda se od bilo koje jedinke moze dobiti bilo koja jedinka.

(2):Ukoliko definisemo mutaciju tako da moze da promeni najvise jedan bit sa verovatnocom p onda:
1010 -> 0000 -> ne
1010 -> 0011 -> ne
1010 -> 1111 -> ne

Matematika jeste poezija hahah
(1): 1010 -> 0000 -> da
(1): 1010 -> 0011 -> da
(1): 1010 -> 1111 -> da

(2): 1010 -> 0000 -> ne
(2): 1010 -> 0011 -> ne
(2): 1010 -> 1111 -> ne

Pitanje 5.27.
Ako tokom primene genetskog algoritma ima N jedinki, svaka je reprezentovana sa M bitova,
a verovatnoća mutacije je p, koliki je očekivani broj mutiranih gena u jednoj generaciji?
Odgovor:
N = 6
M = 3
p = 0.5

jedinke: 000 011 110 111 101 010
ocekujemo: 3
Opsti slucaj:
Na primer p*N, pa to zaokruzeno na nizi ili visi ceo broj.

Pitanje 5.28.
Šta je to elitizam u genetskim algoritmima?
Odgovor:
Kada se odredjeni broj hromozoma u generaciji stiti zbog nekih svojih karakteristika, visoke
funkcije prilagodjenosti i slicno.

Pitanje 5.29.
Navesti bar četiri moguća uslova za zaustavljanje genetskog algoritma.
Odgovor:
* Dostignut odredjeni broj generacija
* Dostignut ekstrem
* Funkcija cilja je zadovoljena
* Funkcija prilagodjenosti je izracunati zadati broj puta
* Kombinacija nekoliko uslova
* Vrednost funkcije prilagodjenosti najbolje jedinke se neko vreme ne menja

GLAVA 7: ISKAZNA LOGIKA

Pitanje 7.1.
Da li nad konačnim skupom iskaznih promenljivih ima konačno ili prebrojivo ili neprebrojivo mnogo (sintaksički) različitih iskaznih formula?
Odgovor:
Konacno mnogo. Logickih veznika ima konacno mnogo (6 preciznije), iskaznih promenljivih ima konacno, te mozemo konstruisati najvise konacan broj logickih formula koje su medjusobno sintaksno razlicite.
Opaska: Ukoliko koriscenje zagrade unutar formule menja njenu sintaksu, onda postoji beskonacno zapisa svake formule jer mozemo dodavati koliko god hocemo zagrada (dokle god su pravilno uparene).

Pitanje 7.2.
Da li nad prebrojivim skupom iskaznih promenljivih ima konačno ili prebrojivo ili neprebrojivo mnogo (sintaksički) različitih iskaznih formula?
Odgovor:
Ima ih beskonacno, ali nisam siguran da li prebrojivo ili neprebrojivo.
Nad svakim podskupom skupa promenljivih postoji konacno mnogo formula koje su medjusobno sintaksno razlicite, no pitanje je da li mozemo sve te formule prebrojati. Pretpostavljam da mozemo.

Pitanje 7.3.
Šta je literal u iskaznoj logici?
Odgovor:
Logicka konstanta ili iskazno slovo.

Pitanje 7.4.
Ako za iskazne formule A i C važi A = C , čemu je jednako A[C ↦→ D]?
Odgovor:
A[C->D] = D
C menjamo sa D, a C je zapravo A tako da se u stvari menja A sa D.

Pitanje 7.5.
Ako za iskazne formule A i C važi A != C i A je atomička formula, čemu je jednako A[C ↦→ D]?
Odgovor:
A[C->D] = A
Jedini nacin da ne bude jednako A je da C = A sto nije ispunjeno u postavci pitanja.

Pitanje 7.6.
Čemu je jednako (p ∧ (¬q ∨ r))[¬q ∨ r ↦→ q ⇒ r]?
Odgovor:
(p & (~q | r))[~q | r ---> q => r] = (p & (q => r))

Pitanje 7.7.
Navesti primer iskazne formule koja je:
Odgovor:
* zadovoljiva: Postoji valuacija u kojoj je formula tacna. Iv(A) = 1 za neko v.
* valjana: Formula je tacna za svaku valuaciju v. Iv(A) = 1 za svako v. Kazemo da je tautologija.
* poreciva: Postoji valuacija u kojoj je formula netacna. Iv(A) = 0 za neko v.
* kontradikcija: Formula je netacna za svaku valuaciju v. Iv(A) = za svako v.

zadovoljiva;	p	
valjana;	p | ~p	
poreciva;	p	
kontradikcija;	p & ~p	
zadovoljiva i valjana;	p | ~p	(isto sto i valjana)
zadovoljiva i nije valjana;	p	
zadovoljiva i poreciva;	p	
zadovoljiva i nije poreciva;	p | ~p	(tajno ime za tautologiju)
zadovoljiva i nije kontradikcija;	p	
valjana i nije poreciva;	p | ~p	(tajno ime za tautologiju)
valjana i nije kontradikcija;	p | ~p	(ako je valjana, svakako nije kontradikcija)
poreciva i nije zadovoljiva;	p & ~p	(tajno ime za kontradikciju)
poreciva i nije valjana;	p	
poreciva i kontradikcija;	p & ~p	(tajno ime za konstradikciju)
poreciva i nije kontradikcija;	p	
kontradikcija i nije zadovoljiva;	p & ~p	(svaka kontradikcija je nezadovoljiva, tj poreciva)
kontradikcija i nije valjana.	p & ~p	(svaka kontradikcija je poreciva i svakako nije valjana)

Pitanje 7.8.
Da li je formula (¬p ∨ q) ⇒ (¬q ∨ p) tautologija, zadovoljiva, poreciva ili nezadovoljiva?
Odgovor:

```
(~p | q ) => (~q | p) = A
  0 1 0   1    1 1 0
  1 1 1   0    0 0 0
  0 0 0   1    1 1 1
  0 1 1   1    0 1 1
```

Formula je zadovoljiva jer postoji:
 * valuacija v1(p) = 0, v1(q) = 0 za koju Iv1(A) = 1
 * valuacija v2(p) = 1, v2(q) = 0 za koju Iv2(A) = 1
 * valuacija v3(p) = 1, v2(q) = 1 za koju Iv2(A) = 1

Formula je poreciva jer postoji:
* valuacija v4(p) = 0, v4(q) = 1 za koju Iv4(A) = 0

Posto je formula istovremeno i poreciva i zadovoljiva, onda ona nije ni valjana ni kontradikcija.

Pitanje 7.9.
Da li je formula (¬p∧p∧¬r) ⇒ (¬q∨r) tautologija, zadovoljiva, poreciva ili nezadovoljiva?
Isto kao primer iznad, naravno ne jednako detaljno :)

Pitanje 7.10.
Ako iskazna formula ima barem jedan model, kakva je onda ona?
Odgovor:
Zadovoljiva. Postoji valuacija v tako da interpretacija Iv(A) = 1 cime je v model za formulu A.

Pitanje 7.11.
Ako iskazna formula nema nijedan model, kakva je onda ona?
Odgovor:
Iskazna formula je onda kontradikcija. Ne postoji ni jedna valuacija za koju Iv(A) = 1 sto je po definiciji kontradikcija.

Pitanje 7.12.
Ako iskazna formula nije poreciva, kakva je onda ona?
Odgovor:
Onda je valjana (tautologija). Ne postoji valuacija v za koju Iv(A) = 0.

Pitanje 7.13.
Ako iskazna formula nije zadovoljiva, kakva je onda ona?
Odgovor:
Onda je kontradikcija. Ne postoji valuacija v za koju Iv(A) = 1.

Pitanje 7.14.
Ako iskazna formula nije kontradikcija, kakva je onda ona?
Odgovor:
Iskazna formula je onda zadovoljiva. Ako pored toga nije poreciva, onda je i valjana.
Primetimo da to sto formula nije kontradikcija, ne znaci da je poreciva.

Pitanje 7.15.
Ako je formula ¬F zadovoljiva, kakva je onda formula F ?
Odgovor:
Poreciva. Valuacija v a koju Iv(~F) = 1 ce dati Iv(F) = 0

Pitanje 7.16.
Ako su iskazne formule A i A ⇒ B tautologije, da li je onda formula B tautologija, zadovoljiva, poreciva ili kontradikcija?
Odgovor:
B je tautologija, ali ako i samo ako su A i A=>B tautologije.

Pitanje 7.17.
Ako su iskazne formule A i A ⇒ B zadovoljive, onda formula B nije nužno zadovoljiva.
Napraviti jedan takav primer (u kojem B nije zadovoljiva, a A i A ⇒ B jesu).
Odgovor:

```
A:      p
B:      q & ~q          (sto je kontradikcija)
A=>B:   p => q
```

Na primer:

Iv(A) = 1 ako v(p) = 1 <- A je zadovoljiva
Iv(B) = 0 jer v(p) = 0 za svako v <- B nije zadovoljiva
Iv(A=>B) = 1 ako Iv(A) != 1 i Iv(B) != 0 <- A=>B je zadovoljiva

Pitanje 7.18.
Kada je iskazna formula A⇒B tačna u valuaciji v ?
Odgovor:
Iv(A) != 1 & Iv(B) != 0

Pitanje 7.19.
Kada je Iv (A ⇒ B) = 0?
Odgovor:
Iv(A) = 1 & Iv(B) = 0

Pitanje 7.20.
U iskaznoj logici, za neku valuaciju v, čemu je jednaka vrednost Iv(A ⇔ B)?
Odgovor:
Iv(A <=> B) =
* 1, ako Iv(A) = Iv(B)
* 0, inace

Pitanje 7.21.
Kako se definiše interpretacija u iskaznoj logici?
Odgovor:
Interpretacija je funkcija koja dodeljuje istinitosnu vrednost slozenim formula na osnovu jedne valuacije v.

Pitanje 7.22.
Da li je u iskaznoj logici odlučiv problem proveravanja
Odgovor:
* zadovoljivosti? DA
* valjanosti? DA
* porecivosti? DA
* kontradiktornosti? DA

Pitanje 7.23.
Kada za iskaznu formulu A kažemo da je logička posledica skupa formula Γ?
Odgovor:
Iskazna formula A je logicka posledica skupa formula GAMA kada je svaki model za GAMA i model za A.

Pitanje 7.24.
Da li nad konačnim skupom iskaznih promenljivih ima konačno ili prebrojivo ili neprebrojivo (zaokružiti ispravan odgovor) mnogo iskaznih formula od kojih nikoje dve nisu logički ekvivalentne?
Odgovor:
G = {p, q} <- konacno mnogo iskaznih promenljivih
Logicka ekvivalencija znaci da su modeli za nastale formule isti, a postoji teorema koja tvrdi da ce takodje
A <=> B biti tautologija.

Samim tim, pitamo se koliko formula mozemo da napravimo nad G tako da im modeli ne budu isti?
p
p & p
p & p & p
...
ali ovo su sve logicki ekvivalentne formule. Slicno za bilo koje iskazno slovo, i slicno za disjunkcije.
A = p & q
B = p & q & p
C = p & q & p & q
Opet vazi A ≡ B ≡ C.

Mozemo konstruisati veliki broj formula (poput gore navedenog) no primecujemo da logicka ekvivalencija elegantno
obuhvata te konstruisane formule. Posto ce u formulama figurisati samo iskazna slova iz skupa G, i imamo
konacno mnogo iskaznih veznika, sledi da postoji konancno mnogo iskaznih formula medju kojima nikoje dve
nisu logicki ekvivalentne.

Pitanje 7.25.
Šta sem Γ |= A mora da važi da bi važilo Δ |= A?
Odgovor:
Ako je podskup model za A, onda je i i nadskup tog podskupa model za A.
Γ ⊂ Δ

Pitanje 7.26.
Kada kažemo da su iskazne formule A ≡ B logički ekvivalentne?
Odgovor:
A ≡ B ako i samo ako je svaki model za A istovremeno i model za B, i ako je svaki model za B istovremeno i model za A.

Pitanje 7.27.
Ako su formule A i B logički ekvivalentne, kako to zapisujemo?
Odgovor:
A ≡ B (trostruka jednakost)

Slicno moze i:
A |= B & B |= A

Pitanje 7.28.
Da li je A ≡ B formula ili meta-formula? Da li je A ⇔ B formula ili meta-formula? Kakva je veza izmedu A ≡ B i A ⇔ B ?
Odgovor:
A ≡ B je meta-formula, odnosno formula o iskaznoj formuli.
A <=> B je formula iskazne logike.

Teorema: A ≡ B ako i samo ako je A <=> B tautologija.

Pitanje 7.29.
Koliko ima klauza dužine k nad skupom od n iskaznih promenljivih:
(a) ako je dozvoljeno da se u klauzi pojavljuje i literal i njegova negacija?
(b) ako nije dozvoljeno da se u klauzi pojavljuje i literal i njegova negacija?
(Podrazumeva se da nije dozvoljeno da se u klauzi pojavljuju logičke konstante niti da se ponavlja isti literal, klauze a se smatraju istim ako se razlikuju samo u poretku literala koje sadrže).
Odgovor:
Klauza: Disjunkcija vise literala.
k1: p | q
k2: p | q | r
...

Sa jednom promenljivom postoji tacno jedna klauze duzine k:
p | p | ... | p, gde postoji k literala.

A = {p1, p2, p3, p4}, n = 4, k = 3
Biramo kombinaciju 3 literala iz skupa od 4 literala.

Binomni koef: (4 nad 3) = 4
p1 | p2 | p3
p1 | p2 | p4
p1 | p3 | p4
p2 | p3 | p4

(a): Pojavljuju se i literal i njegova negacija.
A = {p1, ~p1, p2, ~p2, p3, ~p3, p4, ~p4} i time n = 8, k = 3
Binomni koef: (8 nad 3) = 6720

Odnosno, (2n nad k) -> Kombinacije bez ponavljanja (n broj iskaznih slova).

(b): Nije dozvoljeno pojavljivanje literala i njegove negacije.
(n nad k), gde je n broj iskaznih slova.

Pitanje 7.30.
Koliko ima klauza nad skupom od n iskaznih promenljivih:
(a) ako je dozvoljeno da se u i klauzi pojavljuje i literal i njegova negacija?
(b) ako nije dozvoljeno da se u klauzi pojavljuje i literal i njegova negacija?
(Podrazumeva se da nije dozvoljeno da se u klauzi pojavljuju logičke konstante niti da se ponavlja k isti literal. klauze se smatraju istim ako se razlikuju samo u poretku literala s koje sadrže).

Odgovor:
A = {p1, p2, p3, p4}

(a): Pojavljuju se i literal i njegova negacija.
duzine 1: p1, p2, p3, p4 => 4 (n)
duzine 2: (8 nad 2) => 28 (n nad 2)
duzine 3: (8 nad 3) => 56 (n nad 3)
...
duzine n: (2n nad n) => ...
Dakle suma po i od 1 do n, tako da:
rez := suma (2n, i)

(b) ako nije dozvoljeno da se u klauzi pojavljuje i literal i njegova negacija.
Slicno, samo rez := suma(n, i)

Pitanje 7.31.
Navesti teoremu o zameni za iskaznu logiku.
Odgovor:
Ako vazi C ≡ D, tada A[C->D] ≡ A
Odnosno menjanje formule drugom koja joj je logicki ekvivalentna, ne menja semantiku celokupne formule.
Isti modeli za C ce da vazi i za formulu D koja se nalazi u A.

Pitanje 7.32.
Da li je za iskaznu formulu jednoznačno odredena njena konjunktivna normalna forma?
Odgovor:
Ne u opstem slucaju.

Pitanje 7.33.
Navesti jedan algoritam za transformiranje iskazne formule u knf.
Odgovor:
Ulaz: formula iskazne logike F
Izlaz: formula iskazne logike u KNF-u
1. Sva pojavljivanja <=> zameniti po pravilu: A<=>B ≡ A=>B & B=>A
2. Sva pojavljivanja => zameniti po pravilu: A=>B ≡ ~A | B
3. Primenjivati De Morganove transformacije dok je to moguce:
* ~(A & B) ≡ ~A | ~B
* ~(A | B) ≡ ~A & ~B
4. Ukloniti visestruke negacije.
5. Primenjivati navedena pravila dok je moguce:
* A | (B & C) ≡ (A | B) & (A | C)
* (A | B) & C ≡ (A | C) & (B | C)

Pitanje 7.34.
Šta tokom primene algoritma za transformiranje iskazne formule u knf važi nakon primene logičke ekvivalencije ¬¬A ≡ A?
Odgovor:
U tekucoj formuli postoje jedino logicki veznici &, | i ~, pri cemu uz svaku iskaznu promenljivu postoji navise 1 negacija.

Pitanje 7.35.
Da li se može konstruisati iskazna formula za koju se algoritam KNF ne zaustavlja?
Odgovor:
Problem prebacivanja formule u KNF je odluciv problem, odnosno algoritam se zaustavlja.
Algoritam izlozen u skripti poseduje 5 koraka, pri cemu se moze dokazati da se u svakom koraku formula "blizi" KNF-u. Pored toga, ulazna formula ima konacan broj simbola, te se algoritam zaustavlja.
Moze se uvesti "mera" iskazne formule. Primenom mere formule na algorita, moze se pokazati da se formula smanjuje kroz algoritam.

Pitanje 7.36.
Zašto se zaustavlja prvi korak algoritma KNF?
Odgovor:
Formula sadrzi konacan broj logickog veznika 'ekvivalencija'.
Navedena transformacija eliminise svaku ekvivalenciju u samoj formuli, te se korak 1 zavrsava nakon toga.

Pitanje 7.37.
Zašto se zaustavlja četvrti korak algoritma KNF?

Odgovor:
(4. korak je eliminacija ~~)
Pretpostavimo da postoji k>1 negacija nad iskaznom promenljivom p koja je deo formule A.
Primena pravila 4. nad iskaznom promenljivom p smanjuje broj negacija.
Pravilo se moze primenjivati sve dok broj negacije uz promenljivu p ne postane 0 ili 1. Time se korak zavrsava, a nakon njega u formuli ne postoje suvisne negacije (ima ih najvise 1 uz promenljivu).

Pitanje 7.38.
Navesti teoremu o korektnosti algoritma KNF za iskaznu logiku.
Odgovor:
Neka je ulaz algoritma KNF formula A iz iskazne logike, a na njegovom izlazu formula B.
Algoritam KNF se zaustavlja i nakon njegovog zavrsetka vazi da je B u KNF-u i A ≡ B.

Pitanje 7.39.
Navesti primer skupa formula A veličine n za koje se algoritmom KNF dobijaju formule veličine p(n) (p(n) je polinom po n)?
Odgovor:
A = {p,q, p & q}
Posto se u pitanju trazi primer skupa formula, dovoljno je dati i neki jednostavan primer gde ne dobijamo formulu cija je duzina eksponencijalna u odnosu na duzinu ulazne formule (npr DNF u KNF).

Pitanje 7.40.
Navesti primer skupa formula A veličine n za koje se algoritmom KNF dobijaju formule veličine p(2^n) (p(2^n) je polinom po 2^n)?
Odgovor:
A = {(a & b) | (c & d) | (e & f) | (g & h) | (i & j)}
Ako je formula u DNF-u (disj. norm. forma) i sadrzi n disjunkata, onda ce logicki ekvivalentna formula u KNF-u sadrzati 2^n konjukata (tj eksponencijalno duza).

Pitanje 7.41.
Kako se definišu binarni veznici ↑ i ↓?
Odgovor:
↑ <=> ~(A & B),
↓ <=> ~(A | B)

Pitanje 7.42.
Koliko ima binarnih veznika koji pojedinačno čine potpun skup veznika za iskaznu logiku? Predstaviti te veznike u terminima osnovnih logičkih veznika.
Odgovor:
Seferova i Lukasijeviceva funkcija, odnosno ni i nili, tj postoje samo 2.
↑ <=> ~(A & B)
↓ <=> ~(A | B)

Pitanje 7.43.
Kako se zove problem ispitivanja zadovoljivosti iskazne formule u knf obliku? Da li je ovaj problem odlučiv?
Odgovor:
Problem se zove SAT, problem iskazne zadovoljivosti. Problem jeste odluciv.

Pitanje 7.44.
Da li problem sat pripada klasi p? Da li problem sat pripada klasi np? Da li li je problem sat np-kompletan? np-težak?
Odgovor:
Ne znamo da li problem SAT pripada klasi P, no postoji verovanje unutar naucne zajednice da je P != NP, a samim tim SAT ne bi pripadao klasi P.
Problem SAT jeste NP kompletan, stavise, problem SAT je prvi problem za koji je pokazano da je NP kompletan.
Svaki NP kompletan problem je i NP tezak, dakle SAT je NP tezak.

Pitanje 7.45.
U kom obliku mora da bude formula na koju se primenjuje dpll procedura?
Odgovor:
U konjuktivnoj nomalnoj formi.

Pitanje 7.46.
Koji odgovor vraća dpll procedura ako ulazna formula ne sadrži nijednu klauzu?
Odgovor:
Usvojili smo konvenciju da je prazna klauza zadovoljiva, samim tim DPLL treba da vrati DA kao odgovor. To i radi, i to u prvom koraku.

Pitanje 7.47.
Kako glasi pravilo tautology procedure dpll?
Odgovor:
Neka je dat skup klauza D = {d1, d2, ..., dn}
Pravilo trazi klauzu di u kojoj postoji konstanta tacno. Ukoliko takva postoji,
onda se ona moze eliminisati, odnosno rekurzivno se poziva DPLL(D\di).

Pitanje 7.48.
Kako glasi pravilo split dpll procedure?
Odgovor:
Neka je dat skup klauza D = {d1, d2, ..., dn}
Bira se iskazno slovo p koje pripada nekoj od klauza di.
Poziva se DPLL(D[p -> T]).
Ako DPLL vrati DA, algoritam se zavrsava.
Ako DPLL vrati NE, poziva se DPLL(D[p -> T]).
Pravilo split je najbitnije pravilo jer se tu upravo moze usmeriti pretraga resenja.

Pitanje 7.49.
Koja su pravila dpll procedure primenljiva na formulu: (¬a ∨ b ∨ c) ∧ (a ∨ b ∨ ¬c) ∧ (¬a ∨ b ∨ ¬c)?
Odgovor:
DPLL:
ulaz: D = {d1, d2, ..., dn}
1. Ako je D prazan, vrati DA -> NE, D nije prazan
2. Negacije konstanti prebaci u komplementarnu konstantu -> NE, nema konstanti
3. Ukloni literale 'netacno' -> NE, nema takvih literala
4. Ako D sadrzi praznu klauzu, vrati NE -> NE, D ne sadrzi praznu klauzu
5. tautology -> NE, ne postoji literal T
6. unit prop -> NE, ne postoji klauza koju cini literal (ili neg.)
7. pure literal -> NE, literali unutar klauze su razlicita isk. slova
8. split -> DA, moze se izabrati a, b ili c

Primenljiva su pravila:
split.

Pitanje 7.50.
Da li se može konstruisati iskazna formula u knf formi za koju se algoritam dpll ne zaustavlja?
Odgovor:
Ne. Postoji teorema koja tvrdi da (ukoliko je ulaz za dpll adekvatan) se procedura DPLL zaustavlja.
Osim toga, sam problem SAT je odluciv problem.

Pitanje 7.51.
Koja je složenost dpll procedure u najgorem slučaju?
Odgovor:
Slozenost DPLL procedure u najgore slucaju je eksponencijalna. Neka postoji n iskaznih slova u formuli. Slozenost je O(2^n).

Pitanje 7.52.
Da li postoje iskazne formule za koje je vreme izvršavanja procedure dpll polinomijalno u odnosu na veličinu formule?
Odgovor:
Nisam siguran za ovo. Trebalo bi da postoje, inace bi SAT resavaci bili jako spori u svakoj situaciji (kao sto nisu).

Pitanje 7.53.
Ako želimo da dpll procedurom ispitamo da li je iskazna formula A tautologija, šta treba da bude ulaz za dpll proceduru? U kom je onda slučaju, formula A valjana?
Odgovor:
Ukoliko zelimo da ispitamo da li je A tautologija, ulaz za DPLL je ~A i nadamo se da ce rezultat biti NE.
To znaci da ~A nije zadovoljiva, samim tim, A je uvek zadovoljiva, dakle jeste tautologicna.

GLAVA 8: LOGIKA PRVOG REDA

Pitanje 8.1.
Kako se još nazivaju funkcijski simboli arnosti 0?
Odgovor:
Konstante.

Pitanje 8.2.
Koliko ima formula logike prvog reda nad konačnim skupom predikatskih i funkcijskih simbola, a koliko nad prebrojivim skupom 0 iskaznih promenljivih?
Odgovor:
P = {p, q} <- skup predikatskih simbola
F = {f1, f2} <- skup funkcijskih simbola
Nisam siguran.

Pitanje 8.3.
Šta je literal u logici prvog reda?
Odgovor:
Literal je atomicka formula ili njena negacija.

Pitanje 8.4.
Šta je klauza u logici prvog reda?
Odgovor:
Klauza je disjunkcija vise literala.

Pitanje 8.5.
Šta je term u logici prvog reda?
Odgovor:
Term je:
* funkcijski simbol arnosti 0;
* promenljiva;
* objekat dobijem primenom funkcijskog simbola arnosti n nad termove t1, ..., tn.

Pitanje 8.6.
Da li je u formuli $\forall x(p(x, y) \wedge q(y, z) \wedge r(z))$, promenljiva x slobodna ili vezana, da li je promenljiva y slobodna ili vezana, da li je promenljiva z slobodna ili vezana?
Odgovor:
Promenljiva x je vezana.
Promenljive y i z su slobodne.

Pitanje 8.7.
Za datu signaturu L, šta je to L-struktura D?
Odgovor:
Struktura D je uredjeni par (D, I^L) gde:
* D predstavlja skup koji se naziva domen (ili univerzum),
* I^L predstavlja preslikavanje koje:
    * Funkcijskim simbolima arnosti 0 iz L (konstante) dodeljuje CI iz D
    * Funkcijskim simbolima arnosti n iz L dodeljuje totalnu funckiju fI: D^n --> D
    * Predikatskim simbolima arnosti n iz L dodeljuje totalnu funkciju pI: D^n --> {0, 1}

Pitanje 8.8.
U šta se, u svakoj interpretaciji jezika logike prvog reda, preslikava funkcijski simbol f ?
Odgovor:
Funkcijski simbol se preslikava u totalnu funkciju fI: D^n -> D, gde je n = ar(f), a D domen koji se nalazi u L-strukturi D.

Pitanje 8.9.
U šta se, u svakoj interpretaciji jezika logike prvog reda, preslikava predikatski simbol p?
Odgovor:
Predikatski simbol p se preslikava u totalnu funkciju pI: D^n -> {0, 1}, gde je n = ar(p), a D domen koji se nalazi u L-strukturi D.

Pitanje 8.10.
U standardnoj semantici logike n prvog reda, ako je x promenljiva, čemu je jednako Iv (x) = Iv(x) = v(x)?
Odgovor:
Odnosno interpretacija promenljive x jednaka je valuaciji promenljive x.
Valuacija je preslikavanje koje slika skup promenljivih V u elemente skupa D (domen).
v: V -> D

v(xi) = di: Kazemo da je di vrednost promenljive xi u valuaciji v.

Pitanje 8.11.
Kada u interpretaciji Iv formula ∃xC ima vrednost 0?
Odgovor:
Iv(∃xC) = {
    1, ako postoji valuacija w sa domenom D tako da v ~x w i vazi Iw(∃xC) = 1
    0, inace
}

Pitanje 8.12.
Kada u interpretaciji Iv formula Iv(∀xA) ima vrednost 0?
Odgovor:
Iv(∀xA) = {
    0, ako postoji valuacija w sa domenom D tako da v ~x w i vazi Iw(∀xA) = 0,
    1, inace
}

Pitanje 8.13.
U logici prvog reda, čemu je, za neku valuaciju v, jednaka vrednost Iv(∀xA)?
Odgovor:
Iv(∀xA) = {
    0, ako postoji valuacija w sa domenom D tako da v ~x w i vazi Iw(∀xA) = 0,
    1, inace
}

Pitanje 8.14.
U logici s prvog reda, čemu je, za neku valuaciju v, jednaka vrednost Iv (∃xA)?
Odgovor:
Iv(∃xA) = {
    1, ako postoji valuacija w nad domenom D tako da v ~x w i vazi Iw(∃xA) = 1,
    0, inace
}

Pitanje 8.15.
Ako, u logici prvog reda, za dve valuacije v i w važi v(x) = 1, v(y) = 2, w(x) = 3 i v ~x w, šta važi za w(y)?
Odgovor:
v(x) = 1
v(y) = 2
w(x) = 3
v ~x w
w(y) = ?
Definicija: v ~x w ako za svako y != x vazi v(y) = w(y). v(x) moze ali ne mora biti jednako w(x).
Dakle: v(y) = w(y), y != x, odnosno w(y) = 2

Pitanje 8.16.
Da li je problem zadovoljivosti u logici prvog reda odlučiv ili poluodlučiv k ili neodlučiv?
Odgovor:
Problem zadovoljivosti u logici prvog reda je neodluciv problem.

Pitanje 8.17.
Da li je problem valjanosti u logici prvog reda odlučiv ili poluodlučiv ili neodlučiv?
Odgovor:
Poluodliciv. Ako je formula valjana, mozemo da pokazemo da je valjana. Ako nije valjana, ne mozemo da pokazemo da nije valjana.

Pitanje 8.18.
Ako je formula prvog reda A logička posledica skupa formula Γ, a skup Γ je podskup skupa Δ, šta onda važi?
Odgovor:
Δ |= A. Odnosno A je logicka posledica skupa delta.

Pitanje 8.19.
Kada kažemo da su formule logike prvog reda A i B logički ekvivalentne?
Odgovor:
Formula A i B iz logike prvog reda su logicki ekvivalentne kada je A logicka posledica od B, i B logicka posledica od A.

Pitanje 8.20.
Da li je formula (∀x)(A ∧ B) je logički ekvivalentna nekim od formula:
* (∀x)A ∧ (∀x)B DA (∀ moze da udje pod konjunkciju)
* (∀x)A ∧ B DA (jer kvantori imaju najvisi prioritet (samim tim A ∧ B je isto kvantifikovano)
* (∀x)A ∨ (∀x)B NE (jer je disjunkcija izmedju)
* (∀x)A ∨ B NE (jer je disjunkcija izmedju)

Odgovor:
Data formula je logicki ekvivalentna sa prve dve.
Primetimo da su poslednje dve medjusobno logicki ekvivalentne.

Pitanje 8.21.
Da li su formule (∀xA) ∧ B i (∀xA ∧ B) logički ekvivalentne?
Odgovor:
Ne. Promenljiva x je u univerzalno kvantifikovana i vezana za A, dok je u drugoj formuli, promenljiva x univerzalno kvantifikovana i vezana za formulu A ∧ B.
(∀xA) ∧ B (∀xA ∧ B)

Pitanje 8.22.
Da li su formule (∀xA)∧∀xB i (∀xA∧B) logički ekvivalentne?
Odgovor:
Da, prema zakonu distributivnosti unviverzalnog kvantifikatora u odnosu na konjunkciju.

Pitanje 8.23.
Šta treba da važi za promenljivu x da formule ∀x(A ∧ B) i ∀xA ∧ B nisu nužno logički ekvivalentne?
Odgovor:
Nista, formule su logicki ekvivalentne.

Pitanje 8.24.
Navesti teoremu o zameni za logiku prvog reda? Gde se ona koristi?
Odgovor:
Neka je B1 ≡ B2. Tada A[B1 -> B2] ≡ A.

Teorema garantuje da ako se formula B1 koja je deo formule A zameni logicki ekvivalentnom formulom B2, novodobijena formula je i dalje logicki ekvivalentna sa polaznom formulom.
Odnosno, svaki model za pocetnu formulu ce i dalje biti model za novu, i obrnuto.

Gde se koristi? Pa svuda gde uvodi zamenu u deo formule. Npr u svodjenju proizvolje formule logike prvog reda
u prenex normalu formu.

Pitanje 8.25.
Navesti algoritam PRENEX.
Odgovor:
1. Primenjivati dok je moguce:
(A<=>B) ≡ (A=>B) ∧ (B=>A)
(A=>B) ≡ (~A ∨ B)

2. Primenjivati dok je moguce:
~(A ∨ B) ≡ (~A ∧ ~B)
~(A ∧ B) ≡ (~A ∨ ~B)
~∀x(A) ≡ ∃x(~A)
~∃x(A) ≡ ∀x(~A)

3. Eliminisati visetruke negacije
4. Forsirati kvantifikatore ispred formule po pravilima u skripti (ima ih puno).

Pitanje 8.26.
Dokazati da je formula dobijena algoritmom PRENEX logički ekvivalentna ulaznoj formuli.
Odgovor:
U svakom koraku, tekuca formula se transformise dalje koristeci pravila koja cuvaju logicku ekvivalentnost.
Posto je ulazna formula konacna, i svaki korak algoritma prenex se zavrsava, sledi da je formula dobijen algoritmom PRENEX logicki ekvivalentna pocetnoj formuli.
(Savet: nemojte se uplasiti kod ovakvih pitanja, uglavnom su jednostavni odgovori.)

Pitanje 8.27.
Kako se zove postupak kojim se formula prvog reda transformiše u formulu bez kvantifikatora?
Odgovor:
Prenex + skolemizacija.
To jest, skolemizacija, ali kako bi skolemizacija mogla biti primenjena, potrebno je da formula bude u prenex normalnoj formi.

Pitanje 8.28.
Navesti teoremu o skolemizaciji.
Odgovor:
Neka je *D* = (D, I^L) L-struktura i A formula. Neka je P skup predikatskih simbola i F skup funkcijskih simbola, i ar funkcija arnosti. Dakle L = (F, P, ar).
Skolemizacijom formule A koja je u prenex normalnloj formi iz L-strukture *D* se dobija formula A' nad signaturom L' koja je slabo ekvivalentna formuli A.
Ako je A slabo ekvivalentno B, znaci da je A zadovoljivo ako i samo ako je B zadovoljivo.

Pitanje 8.29.
Ako je formula B dobijena od n formule A skolemizacijom, kakav odnos važi za ove dve formule?
Odgovor:
B je zadovoljivo ako i samo ako je A zadovoljivo. One su slabo ekvivalentne.

Pitanje 8.30.
Zašto formula A i formula dobijena od nje skolemizacijom nisu logički ekvivalentne?
Odgovor:
Menja se signatura pocetne formule uvodjenjem novih funkcijskih simbola.

Pitanje 8.31.
Kada za dve formule A i B logike prvog reda kažemo da su slabo ekvivalentne?
Odgovor:
Kada je A zadovoljivo ako i samo ako je B zadovoljivo.

Pitanje 8.32.
Primenom koja tri koraka se dobija klauzalna forma formule A?
Odgovor:
1. Transformacija formule A u A', gde je A' u prenex normalnoj formi.
2. Transformacija dela bez kvantifikatora uz konjuktivnu normalnu formu cime se dobija formula A''.
3. Skolemizacija formule A'' cime se dobija A'''.
Formula A''' sada vizuelno izgleda poput formule iskazne logike koja je u KNF-u.
Ovakav oblik je pogodan za resavanjem pobijanjem.

Pitanje 8.33.
U kakvom su odnosu formula A i njena klauzalna forma?
Odgovor:
Klauzalna forma je oblik ∀x1,∀x2,∀x3, ...,∀xn A. Sta cemo ukoliko formula A sadrzi egzistencijalni kvantifikator? Na zalost, onda ne postoji klauzalna forma formule A.
Dakle, ako postoji klauzalna forma formule A, onda su one logicki ekvivalentne.
Ako ne postoji, onda se moze pronaci formula B koja je u klauzalnoj formi i slabo je ekvivalentna formuli A.

Pitanje 8.34.
Navesti primer izraza koji pokazaju da relacija unifikabilnosti nije tranzitivna.
Odgovor:
Podsecanje: tranzitivnost relacije ~ je svojstvo da a~b i b~c => a~c
Dakle hocemo da pokazemo da ako je A unifikabilno sa B, i B unifikabilno sa C, onda je i A unifikatbilno sa C.

A = p(x)
B = p(f(y))
C = p(f(x))

UF1 = [x->f(y)]
A[x->f(y)] = p(f(y)) = B = p(f(y))

UF2 = [y->x]
B[y->x] = p(f(x)) = C = p(f(x))

Da li iz pokazanog sledi da A unifikabilno sa C?

Prisetimo se da ako su termovi unifikabilni, onda postoji i najopstiji unifikator.
Dakle dovoljno je pokazati da ne postoji najopstiji unifikator.

Po teoremi od algoritmu najopstiji unifikator znamo da ako algoritam
vrati neuspeh, onda ne postoji najopstiji unifikator.

Primenimo algoritam na paru:
(p(x), p(f(x)))

Koraci 1, 2, 3 nisu moguci.
Primenimo korak 4: DECOMPOSITION (izbacujemo tekuci par a dodajemo nove parove koji su unutar funkcije/predikata fi)
(x, f(x))

Primenimo korak 5: CYCLE
Ako postoji par oblika (x, t), gde je t term koji sadrzi x,
onda vrati neuspeh.

=> NEUSPEH => Ne postoji najopstiji unifikator za A i C => A i C nisu unifikabilni.

Pitanje 8.35.
Ako je za neka dva izraza a σ neki unifikator, a λ najopštiji unifikator, kakav onda postoji unifikator d μ?
Odgovor:
Paznja: Profesor ovde voli da obrne simbole! :)
μ = λ σ
^ ^ ^
| | |
| | neki unifikator
| najopstiji unifikator
Trazeni unifikator

Pitanje 8.36.
Do na šta dva izraza imaju jedinstven najopštiji unifikator?
Odgovor:
Do na supstituciju. Neka je σ najopstiji unifikator, a μ neka supstitucija.
Tada bilo koji unifikator λ mozemo dobiti kao λ = σμ

Pitanje 8.37.
Kako glasi pravilo cycle algoritma za pronalaženje najopštijeg unifikatora?
Odgovor:
Ukoliko postoji par (x, t), gde je x promenljiva, a t term koji sadrzi x, prekini proceduru i vrati neuspeh.

Pitanje 8.38.
U kom slučaju je primenljivo pravilo decomposition u algoritmu za odredivanje najopštijeg unifikatora?
Odgovor:
Ukoliko termovi s i t u paru (s, t) nisu istovremeno promenljive i oblika su:
s = φ(u1, u2, ..., un)
t = φ(v1, v2, ..., vn)
gde je φ funkcijski ili predikatski simbol arnosti n (ar(φ) = n).

Pitanje 8.39.
U kojim koracima algoritam za nalaženje najopštijeg unifikatora može da vrati neuspeh?
Odgovor:
U dve situacije.
U prvoj se neuspeh prijavljuje ukoliko postoji par (s, t), gde su s i t termovi koji nisu promenljive i nisu
oblika:
s = φ(u1, u2, ..., un)
t = φ(v1, v2, ..., vn)
gde je φ funkcijski ili predikatski simbol arnosti n (ar(φ) = n).
U drugoj situaciji se neuspeh prijavljuje kod pravila cycle, koje kaze da ukoliko postoji par (x, t), gde je x promenljiva, a t term koji sadrzi promenljivu x, onda algoritam vraca neuspeh.

Pitanje 8.40.
Navesti algoritam za odredivanje najopštijeg unifikatora.
Odgovor:
ulaz: niz parova: (s1, t1), ..., (sn, tn)
izlaz: najopstiji unifikator σ (ako takav postoji)
Ponavljati sve dok je moguce:
1. FACTORING

* Ako postoji vise od jednog para koji su isti,
    izabrati jedan a ukloniti ostale iz niza parova.

2. TAUTOLOGY
* Ukloniti sve parove oblika (s, s)

3. ORIENTATION
* Za svaki par oblika (t, x), gde je t term, a x promenljiva,
    zameniti ga parom (x, t)

4. Pretpostavimo da je par (s, t) oblika tako da ni s ni t nisu promenljiva.
    4.1 DECOMPOSITION
    * Ako su s i t oblika:
        s = φ(u1, u2, ..., un)
        t = φ(v1, v2, ..., vn)
        gde je φ funkcijski ili predikatski simbol arnosti n (ar(φ) = n),
        par (s, t) zameniti parovima: (u1, v1), (u2, v2), ..., (un, vn)
    4.2 CONFLICT
    * Ako s i t nisu gore navedenog oblika, vratiti neuspeh.

5. CYCLE
* Ako postoji par (x, t), gde je t term koji sadrzi x, vratiti neuspeh.

6. APPLICATION
* Ako postoji par (x, t), gde je x promenljiva, a t term koji ne sadrzi promenljivu x
i x se pojavljuje u nekom drugom paru, primeniti supstituciju [x -> t] na sve parove
osim na par (x, t).

Ako nije pravilo ne moze biti primenjeno,
vratiti tekuci skup parova kao najopstiji unifikator.

Pitanje 8.41.
Ako dva izraza nisu unifikabilna, da li je moguće da se algoritam Najopštiji unifikator zaustavi sa uspehom?
Odgovor:
Ne. Algoritam najopstiji unifikator vraca neuspeh ukoliko izrazi nisu unifikabilni.

Pitanje 8.42.
Ako dva izraza nisu unifikabilna, da li je moguće da se algoritam Najopštiji unifikator zaustavi sa neuspehom?
Odgovor:
Ne da je moguce, nego ce upravo to i da se desi.

Pitanje 8.43.
Ako dva izraza nisu unifikabilna, da li je moguće da se algoritam Najopštiji unifikator ne zaustavi?
Odgovor:
Ne, problem najopstijeg unifikatora je odluciv problem.

Pitanje 8.44.
Da li algoritam za odredivanje najopštijeg unifikatora pripada klasi P? Zašto?
Odgovor:
Pripada klasi P. Postoje algoritmi koji u polinomijalnom vremenu pronalaze najopstiji unifikator. Sto se tice situacije u kojoj dobijeni unifikator moze biti eksponencijalne duzine u odnosu na ulaz, moze se resiti tako sto se koristi DAG (direktni aciklicki graf) umesto implicitnog stabla ili neke druge strukture podataka. Postoji lepa slika na wikipediji koja ilustruje ovu situaciju.

Pitanje 8.45.
Šta je najopštiji unifikator za termove f (x, g(a, y)) i f (z, g(x, z)) (x, y i z su simboli promenljivih, a je simbol konstante)?
Odgovor:
f(x, g(a, y))
f(z, g(x, z))

Zapisimo kao ulaz za algoritam:
(f(x, g(a, y)), f(z, g(x, z)))

START:
(4) Decomposition
(x, z), (g(a, y), g(x, z))

(6) Application [x->z]
(x, z), (g(a, y), g(z, z))

(4) Decomposition
(x, z), (a, z), (y, z)

Vise nije moguce primeniti pravila, dakle najopstiji unifikator je:
σ = [x->z, a->z, y->z]

Proverimo:
f(x, g(a, y))σ = f(z, g(z, z))
f(z, g(x, z))σ = f(z, g(z, z))

Pitanje 8.46.
Šta je najopštiji unifikator za termove f(x, g(a, z)) i f(b, g(y, x)) (x, y i z su simboli promenljivih, a i b su simboli konstanti)?
Odgovor:
PROMENLJIVE: x, y, z
KONSTANTE: a, b

f(x, g(a, z))
f(b, g(y, x))

Zapisimo kao ulaz za algoritam:
(f(x, g(a, z)), f(b, g(y, x)))

START:
(4) Decomposition
(x, b), (g(a, z), g(y, x))

(6) Application [x->b]
(x, b), (g(a, z), g(y, b))

(4) Decomposition
(x, b), (a, y), (z, b)

opaska:
(6) Application nije moguc jer su su jedine promenljive na levoj strani para x i z,
ali se one ne pojavljuju u nijednom drugom paru.

(3) Orientation (NE ZABORAVITI)
Mozemo da primenimo jer pravilo kaze da ako postoji (t, x), gde je t term koji nije promenljiva,
a x jeste promenljiva, onda menjamo taj par sa (x, t). Konstante a i b nisu promenljive.
(x, b), (y, a), (z, b)

Ne mozemo primenjivati vise pravila, najopstiji unifikator je:
σ = [x->b, y->a, z->b]

Proverimo:
f(x, g(a, z))σ = f(b, g(a, b))
f(b, g(y, x))σ = f(b, g(a, b))

Pitanje 8.47.
Šta je najopštiji unifikator za termove f(a, g(x, y)) i f(z, g(a, z)) (x, y i z su simboli promenljivih, a je simbol konstante)?
Odgovor:
PROMENLJIVE: x, y, z
KONSTANTE: a

f(a, g(x, y))
f(z, g(a, z))

Zapisimo kao ulaz za algoritam:
(f(a, g(x, y)), f(z, g(a, z)))

START:
(4) Decomposition
(a, z), (g(x, y), g(a, z))

(3) Orientation
(z, a), (g(x, y), g(a, z))

(4) Decomposition
(z, a), (x, a), (y, z)

(6) Application [z->a]
(z, a), (x, a), (y, a)

Ne mozemo primenjivati vise pravila, najopstiji unifikator je:
σ = [z->a, x->a, y->a]

Proverimo:
f(a, g(x, y))σ = f(a, g(a, a))
f(z, g(a, z))σ = f(a, g(a, a))

Pitanje 8.48.
Šta je rezolventa klauza Γ'∨ A' i Γ''∨ ¬A'' je (σ je najopštiji unifikator za A' i A'')?
Odgovor:
Binarni metod rezolucije za logiku prvog reda definisemo kao:
Γ'∨ A' i Γ''∨ ¬A''

(Γ' ∨ Γ'')σ

Odakle je rezolventa (Γ' ∨ Γ'')σ.

Pitanje 8.49.
Navesti pravilo rezolucije za logiku prvog reda.
Odgovor:
Γ'∨ A' i Γ''∨ ¬A''

(Γ' ∨ Γ'')σ
gde je:
σ najopstiji unifikator za A' i A'', a Γ' i Γ'' su klauze.

Pitanje 8.50.
Da bi se primenio metod rezolucije u kakvoj formi formula čija se nezadovoljivost ispituje mora da bude?
Odgovor:
Najpre se formula mora prevesti u PRENEX oblik. Potom se deo formula iza kvantifikatora transformise u KNF, i konacno, eliminisu se egzistencijalni kvantifikatori skolemizacijom.

Pitanje 8.51.
Navesti teoremu o potpunosti metode rezolucije za iskaznu i predikatsku logiku.
Odgovor:
==========
Teorema (potpunost metoda rezolucije za iskaznu logiku):
==========
Metod rezolucije za iskaznu logiku se zavrsava, i prazna klauza se moze izvesti ako i samo je ako ulazna formula nezadovoljiva.

==========
Teorema (potpunost metoda rezolucije za predikatski logiku):
==========
Ako je ulazna formula nezadovoljiva, onda se metodom rezolucije moze izvesti prazna klauza. Izvodjenje prazne klauze znaci da je formula nezadovoljiva.

==========
Teorema (potpunost sistematicnog metoda rezolucije za predikatski logiku):
==========
Ako je ulazna formula nezadovoljiva, onda se sistematicnim metodom rezolucije mora izvesti prazna klauza.

Pitanje 8.52.
Koje korake je potrebno primeniti da bi se metodom rezolucije ispitalo da li je formula logike prvog reda A valjana?
Odgovor:
1. Transformacija formule u PRENEX formu (cime kvantori izlaze ispred)

2. Transformacija formule iza kvantora u KNF
3. Skolemizacija, tj eliminisanje egzistencijalnih kvantora.

Time se dobija formula oblika (∀x1)(∀x2)...(∀xn)A, gde je A u KNF-u, odnosno:
A = C1 ∧ C2 ∧ ... ∧ Cn
Metod rezolucije se primenjuje nad klauzama Ci, i = 1, 2, ..., n

Pitanje 8.53.
Da li se metodom rezolucije za svaku formulu logike prvog reda koja je valjana može dokazati da je valjana?
Odgovor:
1.Dokazivanje valjanosti radimo tako sto pokusamo da dokazemo da je negacije formule nezadovoljiva.
2.Ako je formula valjana, onda njena negacija jeste nezadovoljiva.
3.
=======
Teorema: Ako je formula logike prvog reda nezadovoljiva, metod rezolucije moze da izvede praznu klauzu.
=======
Na osnovu 1), 2) i 3), odgovor je DA.

Pitanje 8.54.
Da li se metodom rezolucije za svaku formulu logike prvog reda koja nije valjana može dokazati da nije valjana?
Odgovor:
Ne. Problem ispitivanja formule logike prvog reda je poluodluciv problem.
Mozemo da pokazemo da je formula valjana jer metodom rezolucije mozemo da izvedemo praznu klauzu u njenoj negaciji.

Pitanje 8.55.
Koji su mogući ishodi primene metoda rezolucije za iskaznu logiku, a koji za logiku prvog reda?
Odgovor:
Primena metoda rezolucije nad formulama iskazne logike nam garantuje da ce se:
* zavrsiti;
* i da ce se izvesti prazna klauza ako i samo ako je polazna formula nezadovoljiva.

Problem se komplikuje pri prelasku na logiku prvog reda jer problem sa odlucivog postaje poluodluciv.
Navedeno najbolje ilustruju sledece dve teoreme:

==========
Teorema (potpunost metoda rezolucije za predikatski logiku):
==========
Ako je ulazna formula nezadovoljiva, onda se metodom rezolucije moze
izvesti prazna klauza. Izvodjenje prazne klauze znaci da je formula nezadovoljiva.

==========
Teorema (potpunost sistematicnog metoda rezolucije za predikatski logiku):
==========
Ako je ulazna formula nezadovoljiva, onda se sistematicnim metodom rezolucije
mora izvesti prazna klauza.

Pri cemu je sistematski metod rezolucije metod u kojem se rezolvente tako
biraju da se nikad necemo vrteti u krug ukoliko postoji prazna klauza.

Dakle, ukoliko je formula nezadovoljiva, mozemo da pokazemo da je nezadovoljiva
(metod rezolucije izvodi praznu klauzu).

Samim tim, ukoliko dokazujemo valjanost formule (koja je zaista valjana),
onda je njena negacija nezadovoljiva, i metod rezolucije ce izvesti praznu klauzu.

Sta ukoliko pokusavamo da dokazemo da je formula valjana iako to ona nije?
Njena negacija onda nije nezadovoljiva, cime po navedenim teoremama,
nemamo garanciju da ce metod rezolucije izvesti praznu klauzu, a time se
ikada i zavrsiti. U ovoj situaciji se oseca poluodlucivost.

Pitanje 8.56.
Da li se metod rezolucije za iskaznu logiku uvek zaustavlja?
Odgovor:
Ne. Ukoliko je ulazna formula metoda rezolucije nezadovoljiva, onda se metod zaustavlja.
Ukoliko nije, onda treba biti strpljiv :P

Pitanje 8.57. U iskaznoj logici, da li će kako god se primenjivalo pravilo rezolucije u konačnom broju koraka biće izvedena prazna klauza?
Odgovor:
* ako je početni skup klauza zadovoljiv?
* ako je početni skup klauza nezadovoljiv?
Odgovor daje teorema.
=======
TEOREMA: Metod rezolucije primenjen nad formulom iskazne logike se zavrsava i izvodi praznu klauzu
======= ako i samo ako je polazna formula nezadovoljiva.

Dakle, ako je pocetni skup klauza zadovoljiv, ne mozemo izvesti praznu klauzu.
Ukoliko je pocetni skup klauza nezadovoljiv, u konacno mnogo koraka cemo svaki put izvesti praznu klauzu.

GLAVA 9: PROLOG

Pitanje 9.1.
Na kom metodu je zasnovan mehanizam izvodenja zaključaka u PROLOG-u?
Odgovor:
Na metodu rezolucije.

Pitanje 9.2.
Kakve klauze logike prvog reda odgovaraju PROLOG činjenicama, pravilima i upitima?
Odgovor:
N/A

Pitanje 9.3.
Koliko literala bez negacije se moze javiti u klauzama koje se koriste u PROLOG-u?
Odgovor:
Najvise jedan.
Zasto?
Prisetimo se formule: A=>B logicki ekvivalentno sa ~A | B.
Slicno A1 & A2 & A3 & ... & An => B logicki ekvivalentno sa ~A1 | ~A2 | ... | ~An | B.
Samim tim, sme se pojaviti najvise jedan literal bez negacije, B.

Pitanje 9.4.
Koja klauza prvog reda odgovara PROLOG pravilu
p(x1,x2,...,xn) :- q(y1,y2,...,ym),...,r(z1,z2,...,zk) ?
Odgovor:
q(y1, y2, ..., ym) & ... & r(z1, z2, ..., zk) => p(x1, x2, ..., xn)
odnosno
~q(y1, y2, ..., ym) | ... | ~r(z1, z2, ..., zk) | p(x1, x2, ..., xn)

Pitanje 9.5.
Da li se algoritam mergesort može implementirati u PROLOGU?
Da li se algoritam quicksort može implementirati u PROLOGU?
Odgovor:
Da. Da. Prolog poseduje liste i poseduje rekurziju.

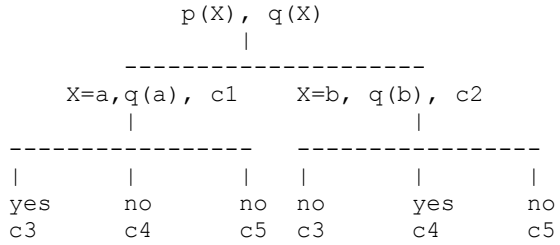
Pitanje 9.6.
Da li se u PROLOG-u stablo izvodenja obilazi u dubinu ili u širinu?
Odgovor:
Stablo se obilazi u dubinu.

Pitanje 9.7.
U Prologu svakom listu stabla pretrage odgovara ili unifikacija koja daje jedno rešenje ili _____?
Odgovor:
ili neuspela unifikacija, odnosno neuspesno ispunjenje cilja.

Pitanje 9.8.
Ako neki PROLOG cilj može da bude zadovoljen, koliko onda u odgovarajućemu stablu izvodenja postoji listova koji ne prestavaljaju praznu klauzu.
Odgovor:
Nije mi jasno pitanje. Svakako ce biti barem jedna prazna klauza jer je cilj moguce ispuniti, ali u pitanju nije nista receno o tome sta jos postoji unutar koda koji prolog posmatra, niti je ista receno o PROLOG predikatu koji treba ispuniti.

Pitanje 9.9.
U PROLOG bazi postoji skup činjenica:
c1: p(a).
c2: p(b).
c3: q(a).
c4: q(b).
c5: q(c).
Nacrtati stablo izvodenja za upit:
* p(X),q(X).
* p(X),q(Y).
* p(X),!,q(X).
* p(X),!,q(Y).

Odgovor:



Pitanje 9.10.
Kako se naziva operator sečenja čijim se dodavanjem ne menja skup rešenja, ali se povećava efikasnost izračunavanja?
Odgovor:
Zeleni operator secenja.
Primer:
```
max(X, Y, Y) :- X =< Y, !.
max(X, Y, X) :- X > Y.
```

Pitanje 9.11.
U čemu se razlikuju zeleni i crveni operator sečenja u PROLOG-u?
Odgovor:
Zeleni operator secenja ne menja semantiku predikata i njegov skup resenja. Upotrebljava se radi efikasnije pretrage.
Crveni operator secenja menja semantiku programa, a samim tim i skup resenja. Potencijalni problem sa ovim je sto nekada mozda nismo ni svesni da u nasem predikatu postoji crveni operator secenja jer se predikat ponasa naizgled isto.

Pitanje 9.12.
Da li zeleni operator sečenja može da odseca deo stabla pretrage u PROLOG-u?
Da li crveni operator sečenja može da odseca deo stabla pretrage u PROLOG-u?
Odgovor:
Da. Da. Kada se zadovolje ciljevi navedeni ispred operatora !, PROLOG prestaje da obilazi dalje stablo pretrage.

Pitanje 9.13.
Da li operator sečenja u PROLOG-u nužno menja rezultat programa?
Da li operator sečenja u PROLOG-u može da promeni rezultat programa?
Odgovor:
Operator secenja ne menja nuzno rezultat programa. Ukoliko je zeleni onda ne menja, ukoliko je crveni onda menja.

Pitanje 9.14. Kako treba modifikovati Prolog upit ?- r(X),s(Y). da on pronalazi samo jednu vrednost za X?
Odgovor:
?- r(X), s(Y) -> ?- s(Y), r(X), !.
Primetimo da ako bi bilo s(Y), !, r(X). onda bi prolog zadovoljio cilj s(Y) i zaustavio pretragu. Time smo promenili semantiku programa.
Ukoliko bi bilo r(X), !, s(Y). onda bi prolog zadovoljio jednom cilj r(X), ali bi zaustavio pretragu cime smo opet promenili semantiku programa.

Pitanje 9.15.
Kako se u PROLOG-u obično definiše operator not?
Odgovor:
```
not(X) :- call(X), !, fail.
not(X).
```

Pitanje 9.16.
Navesti PROLOG cilj koji uspeva ako i samo ako su termovi A i B unifikabilni ali ih pri tom ne unifikuje.
Odgovor:
```
\+(\+(A=B)).
```

Pitanje 9.17.
Da li su PROLOG termovi [[],[]] i [H|T] unifikabilni i ako jesu - sa kojim unifikatorom:
Odgovor:
A = [[], []]
B = [H|T]
Unifikator?
[H|T] je opsta lista u prologu. [[], []] je lista koja sadrzi dva elementa koji su prazne liste.
Unifikator: H = [], T = [[]] H = [] jer jer prvi element liste A prazna lista [].
T = [[]] jer je T po definiciji lista iz koje je uklonjen prvi element. Dakle kada se ukloni [] iz [[], []] ostaje [[]].

Pitanje 9.18.
Da li su PROLOG termovi [[]] i [H|T] unifikabilni i ako jesu - sa kojim unifikatorom:
Odgovor:
A = [[]]
B = [H|T]

H = [] <- prvi element [[]] je []
T = [] <- kada se ukloni prvi element iz [[]], ostaje []

Pitanje 9.19.
Kako se u PROLOG-u, u interaktivnom radu, dodaje nova činjenica?
Odgovor:
assert(X) dodaje cinjenicu X.

Pitanje 9.20.
Koji predikat prekida izvršavanje a Prolog interpretarora?
Odgovor:
halt.

Pitanje 9.21.
Kojim izrazom se u PROLOG-u promenljiva X veže za numeričku vrednost, npr. 5?
Odgovor:
X is 5.

Pitanje 9.22.
Kako se PROLOG program učitava iz datoteke 'datoteka'?
Odgovor:
consult(datoteka).
consult('datoteka').
Napomena: Na mom racunaru je instaliran swi-prolog. Nije bilo potrebno da navodim ekstenziju pl u imenu, ali je radilo i sa tim.

Pitanje 9.23.
Koji upit u PROLOG-u uspeva ako i samo ako je X neinstancirana promenljiva?
Odgovor:
TODO:
Nisam siguran.
var(X).

GLAVA 10 NEMA SEKCIJE SA PITANJIMA, STO NE ZNACI DA NISTA NECE DOCI ODATLE!

GLAVA 11: NADGLEDANO MASINSKO UCENJE

Pitanje 11.1.
Kako se naziva proces u kojem se znanje koje važi za neki skup instanci prenosi na neki njegov nadskup?
Odgovor:
Generalizacija. Na osnovu znanja koje posedujemo za odredjeni skup instanci, pokusavamo da dodjemo do opstih zakljucaka koji se mogu primeniti na siri skup (nadskup datog skupa instanci).

Pitanje 11.2.
U čemu se razlikuju nadgledano i nenadgledano učenje?
Odgovor:
Proces ucenja se razlikuje. Kod nadgledanog ucenja, podaci koje posedujemo su obelezeni (eng. labeled) i predstavljaju vrednosti koje zelimo da sistem koji se trenira pogodi. Proces ucenja u ovakvim situacijama se cesto izvodi tako sto se podaci dele na dva skupa:
* trening skup;
* test skup.
Trening skup se koristi u samom ucenju (treniranju), a potom se istrenirani sistem testira sa podacima (koje do sada nije video) iz test skupa.
Kod nenadgledanog ucenja, obelezja (cesto nazivane i labele) ne postoje, te je potrebno drugacije pristupiti problemu.
Na primer mozemo se vrsiti klasterovanje (eng. clustering) koje pokusava da slicne instance klasifikuje u istu klasu.
Dakle od samog sistema se ocekuje da uoci zakonitosti ili pravilnosti koje se javljaju.

Pitanje 11.3.
Kako se zove učenje kod kojeg se algoritmu zajedno sa podacima iz kojih uči daju i željeni izlazi?
Odgovor:
Nadgledano ucenje.

Pitanje 11.4.
Kako se u mašinskom učenju zovu atributi instanci čije vrednosti se ne mogu prirodno numerički opisati?
Odgovor:
Kategoricki atributi. Predstavljaju atribute koji za instancu govore kojoj kategoriji pripada. Na primer, ukoliko se klasifikuju studenti, kategoricki atribut bi mogao biti 'redovan student'. Cesto se kategorickim atributima dodeli neka numericka vrednost, npr 1 ako je student 'redovan student', a 0 ako 'nije redovan student'.

(dalje slede pitanja koja ne dolaze na kolokvijum, ukoliko se ne varam)

Pitanje 11.5. Kakve su promenljive koje predvidaju u slučaju klasifikacije, a kakve u slučaju regresije?
Klasifikacija kao izlaz daje informaciju o tome kojoj klasi pripada ulazni podatak, odnosno radi se o kategorickim
vrednostima. Kao sto je pomenuto u odgovoru na pitanje 11.4, cesto ce izlaz biti 'numericki' iz "tehnickih razloga",
no sam taj numericki izlaz se u stvari koristi da oznaci neku klasu, odnosno kategoriju.

Problem regresije za dati ulaz kao izlaz daje numericke vrednosti. Na primer, koliko ocekujemo da kosta nasa
nekretnina ukoliko ona poseduje 4 sobe, 1 terasu, garazu i 5 bazena.

Mašinsko učenje se bavi proučavanjem:
(a) dedukcije;
(b) pretrage;
(c) generalizacije;
(d) optimizacije;
(e) ne znam.
Najvise generalizacijom.
Masinsko ucenje sa sobom donosi i probleme pretrage i optimizacije takodje, no u manjoj meri.
Centralna tema jeste da se izvede sto bolja generalizacija problema.

Pitanje 12.1. Koji od narednih modela su linearni?
(1) y = β1x + β2z
(2) y = β0 + β1*x^2 + β2*x^3
(3) y = β0 + β1*log(x) + β1*log(x)

(4) y = β0 + β1*log(x) + β1*log(sin(x))
Linearne regresija se zove linearna iz razloga sto postoji linearna zavisnost
izmedju tezinskih koeficijenata (βi u primeru).

(1): Jeste linearan model

(2): y = β0 + β1*x^2 + β2*x^3
    Primetimo da mozemo transformisati izraz u:
    y = β0 + x^2(β1 + β2*x)
    Jasno je da ne postoji linearna zavisnost izmedju koefiicijenata βi.

(3): y = β0 + β1*log(x) + β1*log(x)
    Primetimo da mozemo transformisati izraz u:
    y = β0 + 2*β1*log(x)
    Jeste linearni model.
(4): y = β0 + β1*log(x) + β1*log(sin(x))
    y = β0 + β1(log(x) + log(sin(x)))
    Jeste linearni model.

Pitanje 12.2. Ako se učenje vrši sa siromašnim skupom dopustivih modela, da li to može dovesti do
loših rezultata?
PAZNJA: Nemojte se zaleteti i pomisliti na siromasan skup PODATAKA. U pitanju je skup dopustivih
modela,
    npr. {w1x1 + w2x2 + w3 | w1, w2, w3 realni brojevi}

Ukoliko skup dopustivih modela nije dovoljno bogat, postoji verovatnoca da nece biti u stanju da
izvede
dovoljno dobru generalizaciju cime moze dovesti do losih rezultata, no cesto u praksi i nedovoljno
bogat skupo modela bude dovoljno dobar.

Pitanje 12.3. Ako se učenje vrši sa bogatim skupom dopustivih modela, da li to može dovesti do
loših rezultata?
Prirodno bi bilo da ne, no ipak moze. Ukoliko nam je skup dopustivih modela prebogat, onda se
smanjuje
moc generalizacije i mozemo dobiti losije rezulte (ili katastrofalne rezultate).

U skripti je naveden primer sa koriscenjem polinoma n-tog stepena za linearnu regresiju i
klasifikaciju.

Pitanje 12.4. Šta je čest uzrok lošeg ponašnja modela koji ima dobre mere kvaliteta na trening
podacima?
Preprilagodjavanje (eng. overfitting) i neadekvatno odabrane mere kojima posmatramo i evaluiramo
kvalitet ucenja.

Pitanje 12.5. Koju raspodelu se pretpostavlja da ima šum pri korišćenju linearne regresije?
U linearnoj regresiji pretpostavljamo da vazi: y = wx + eps, eps ~ N(0, σ^2)
Dakle, eps ~ N(0, σ^2)

Pitanje 12.6. Šta je osnovna mera kvaliteta linearne regresije?
Osnovna mera kvaliteta linearne regresije je srednjekvadratna greska.

Pitanje 12.7. Navesti definiciju srednjekvadratne greške.
E(Y, Xw) = ∑□ (yi − w·xi)^2

Pitanje 12.8. Navesti barem dva algoritma klasifikacije.
* Logisticka regresija;
* k-najblizih suseda;
* stabla odlucivanja;
* metoda potpornih vektora.

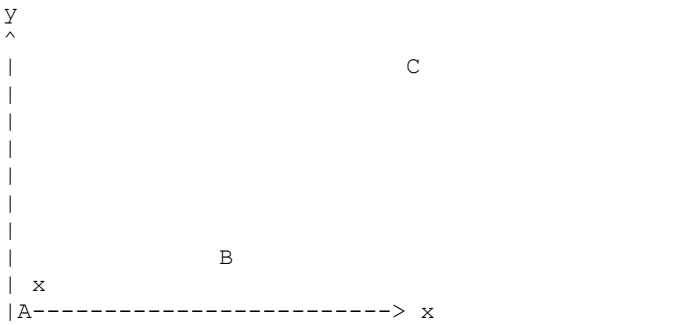
Pitanje 12.9. Da li su modeli koje grade metode zasnovane na instancama implicitni ili eksplicitni?
Implicitni. Model ne postoji eksplicitno, vec se u fazi same klasifikacije vrsi klasifikacija na
osnovu
modela koje se implicitno konstruise.

Pitanje 12.10. Kako se zove metod klasifikacije koji koristi n instanci za
koje je rastojanje do instance koja se klasifikuje najmanje?
Metod n-najblizih suseda.

Pitanje 12.11. Navesti primer funkcije rastojanja koja se može koristiti u metodi n najbližih suseda.
Euklidsko rastojanje. Videti u skripti jos neku ukoliko profesor trazi da se nabroje barem 2.

Pitanje 12.12. Da li su u metodu n najbližih suseda rezultati bolji za veće vrednosti n?
Da li u metodu n najbližih suseda kvalitet rezultata zavisi od n?
Da li u metodu n najbližih suseda postoji opšte gornje ograničenje 6 za n?
Da li su u metodu n najbližih suseda rezultati bolji za veće vrednosti n? Ne nuzno.
Da li u metodu n najbližih suseda kvalitet rezultata zavisi od n? Naravno.
Da li u metodu n najbližih suseda postoji opšte gornje ograničenje za n? Ne, kamo srece da postoji.

Pitanje 12.13. Instanca (1, 0) pripada klasi A, instanca (9, 1) pripada klasi B ,
a instanca (15, 19) pripada klasi C . Kojoj od ovih klasa bi algoritam n-
najbližih suseda pridružio instancu (2, 2) za n = 1?
Napomena: Nije navedena funkcija rastojanja, pretpostavljam da se posmatra euklidsko rastojanje.



=> Instanca (1, 0): A je najblizi sused, tako da pripada klasi A.

Pitanje 12.14. Koliko ima 2-grama u reči matematika i koje su njihove frekvencije u ovoj reči?
ma: 2
at: 2
te: 1
em: 1
ti: 1
ik: 1
ka: 1

Pitanje 12.15. Da li, za konačnu azbuku, n-grama za fiksno n ima: konačno
mnogo, prebrojivo mnogo ili neprebrojivo mnogo?
Ako je azbuka konacna, to znaci da npr moze da postoji k simbola.
Koliko reci je moguce konstruisati nad tom azbukom? Neprebrojivo mnogo.
No za fiksno n?
n-grami ce biti sve moguce kombinacije duzine dva nad skupom azbuke.
Takvih n-grama ima k^n, dakle konacno mnogo.

Pitanje 12.16. Šta čini n-gramski profil instance?
N-gramski profil cini skup svih n-grama zajedno sa brojem njihovim pojavljivanja.

Pitanje 12.17. Navesti barem dve funkcije rastojanja koje se mogu koristiti
za klasifikaciju n-gramskih profila metodom n najbližih suseda.
* Euklidsko rastojanje.
* Uopstenje euklidskog rastojanja $\sqrt[n]{\sum((x_i - y_i)^n)}$

Pitanje 12.18. Navesti ime barem jednog algoritma za konstrukciju stabla
odlučivanja na osnovu skupa instanci za trening.
ID3

Pitanje 12.19. Navesti algoritam ID3.
Lepse je to navedeno u skripti! :D

Pitanje 12.20. Šta vraća algoritam ID3 u slučaju da je lista atributa prazna?
Vraca cvor R koji je oznacen klasom koja se najcesce javlja u trening podacima.

Pitanje 12.21. Šta vraća algoritam ID3 u slučaju da sve ulazne instance pripadaju istoj klasi?
Vraca cvor R sa oznakom te klase.

Pitanje 12.22. Da li algoritam ID3 ima tendenciju da konstruiše plića ili dublja stabla odlučivanja?
Plica. To predstavlja posledicu samog algoritma koja pohlepno gradi stablo odozgo-nadole trazeci atribut ko kojem dobija najbolju podelu podataka.

Pitanje 12.23. Koje se mere obično koriste za izbor najpogodnijeg atributa prilikom izgradnje stabla odlučivanja?
Entropija i greska klasifikacija.
Entropija predstavlja meru neuredjenosti skupa.
Greska klasifikacija predstavlja gresku koja se cini ukoliko se sve instance nekog skupa klasifikuju kao najbrojnija klasa.

Pitanje 12.24. Navesti definiciju veličine Entropija(S).
Unicode je unicode ali LaTeX je LaTeX. Pogled u skriptu!

Pitanje 12.25. Ako se razmatra entropija kuglica rasporedenih u dve činije, kada ona najveća, a kada najmanja?
K = {k1, k2, ..., kn} <- n kuglica
A, B <- cinije
p(k = A) <- verovatnoca da kuglica k bude rasporedjena u ciniju A
p(k = B) <- verovatnoca da kuglica k bude rasporedjena u ciniju B

Entropija ce biti najveca ukoliko je p(k = A) = p(k = B) = 0.5,
odnosno ukoliko su kuglice ravnomerno rasporedjene.

Najmanja je ukoliko vazi:
p(k = A) = 1
p(k = B) = 0
ili
p(k = A) = 0
p(k = B) = 1

Cime su sve kuglice u jednoj od cinija.

Pitanje 12.26. Ako skup sadrži podjednako n instanci iz dve klase, kolika je vrednost entropije za taj skup?
muskarci i zene, n = 5.
S ce sadrzati 5 muskaraca i 5 zena.
c = 2, tj postoje 2 klase

log(x): Logaritam od x za osnovu 2.

Entropy(S) = -SUM(pi*log(pi)), i = 1, 2
Entropy(S) = -(1/2*log(1/2) + 1/2*log(1/2))
Entropy(S) = -(log(1/2))

Pitanje 12.27. Kako se definiše entropija skupa S podeljenog na podskupove veličina p1 , p2, . . ., pc?
Nije mi jasno u pitanju tacno sta znaci da je skup podeljen na podskupove, tj da li znaci da postoji c klasa?
Odgovor sledi pod pretpostavkom da postoji c klasa.

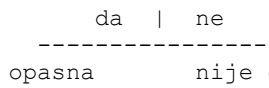
Entropy(S) = -SUM(pi*log(pi)), i = 1, 2, ..., c

Pitanje 12.28. U jednom skupu instanci, verovatnoća da proizvoljna instanca pripada klasi C1 jednaka o
je 1/4, verovatnoća da pripada klasi C2 jednaka je 1/4, a verovatnoća da pripada klasi C3 jednaka je 1/2. Kolika je
entropija ovog skupa?
Dakle postoje 3 klase, C1, C2, C3 cime c = 3
Imamo skup instanci S.
p1 = 1/4
p2 = 1/4
p3 = 1/2

Entropy(S) = -SUM(pi*log(pi)), i = 1, ..., c
Entropy(S) = -SUM(pi*log(pi)), i = 1, 2, 3
Entropy(S) = -(1/4*log(1/4) + 1/4*log(1/4) + 1/2*log(1/2))

log(x): Logaritam od x za osnovu 2.

Pitanje 12.29. Kakva su pravila koja se lako mogu izvesti iz stabla odlučivanja?
Neka je dato sledece stablo odlucivanja za opasne zivotinje.



Iz stabla mozemo zakljuciti da ako je zivotinja otrovna, onda je opasna.
Odnosno:
if A1=v1 & A2=v2 & ... & An=vn:
    Klasifikovana_klasa = Klasa_koja_odgovara_klistu

Pitanje 12.30. Koja je osnovna mera kvaliteta klasifikatora?
Preciznost.

P = (SP + SN)/(SN + SP + LP + LN)
SP -> Stvarno pozitivni -> pozitivni i klasifikovani kao pozitivni
SN -> Stvarno negativni -> negativni i klasifikovani kao negativni
LP -> Lazno pozitivni -> negativni i klasifikovani kao pozitivni
LN -> Lazno negativni -> pozitivni i klasifikovani kao negativni

Pitanje 12.31. Stablo odlučivanja je za 5 instanci ponudilo klase A, A, B, B, A,
dok su ispravne klase bile A, A, A, B, B.
Kolika je preciznost ovog stabla odlučivanja?
A A B B A <- klasifikator
A A A B B <- stvarne klase

SP = 3/5
Odnosno, ako bi A bilo 'pozitivno', B 'negativno'.
SP = Broj pogodjenih A-ova koji su zaista A = 2
SN = Broj pogodjenih B-ova koji su zaista B = 1

SP+SN+LP+LN = Broj svih klasifikacija = 5

Pitanje 12.32. Koji procenat podataka i se u mašinskom učenju obično uzima za trening podatke, a
koji za test podatke?
Najcesce, 1/3 se uzima za test podatke, 2/3 za trening podatke, pri cemu su ova dva skupa
disjunktna.

Pitanje 12.33. Šta se, radi pouzdanije evaluacije klasifikatora, često koristi
umesto jednog deljenja na trening i test podatke?
Unakrsna validacija (eng. cross validation)

Pitanje 12.34. Kako se zove postupak evaluacije modela mašinskog učenja kojem se skup raspoloživih
podataka deli na n delova, a zatim trenira izostavljajući po 1 jedan od njih?
Unakrsna validacija (eng. cross validation) (hmmm deja vu, kao da sam ovo vec negde rekao xD)

Pitanje 12.35. Kako se sprovodi unakrsna validacija?
Skup se deli na N priblizno jednakih podskupova.
Tih N skupova se deli na 1 i N-1, trenira se na N-1 podskupu i testira na ovom jednom.
To se radi unakrsno za ostale izabrane podskupove i na kraju se sve evaluira kao prosek dobijenih
ocena
za svaki od podskupova.

Pitanje 12.36. U problemu klasifikacije, za koje instance kažemo da su lažno pozitivne?
Instance koje su lazno pozitivne, su one koje je klasifikator klasifikovao kao pozitivne, a zapravo
su negativne.

Pitanje 12.37. Kako se definiše veličina USP (udeo stvarno pozitivnih)?
USP = SP/(SP + LN)

SP -> stvarno pozitivne
LN -> stvarno negativne