Dragan Bojić, Miloš Gligorić, Boško Nikolić

ZBIRKA ZADATAKA
IZ
EKSPERTSKIH SISTEMA

Radna verzija

Beograd, 2009.

Sadržaj

sestri Mirjani – Miloš

Lani i Maji – Boško

Predgovor

Zbirka zadataka iz Ekspertskih sistema treba da posluži bržem i potpunijem savladavanju teorijskih i praktičnih osnova oblasti Ekspertskih sistema. Ciljevi knjige su:

- ilustraciju niza tehnika i algoritama na problemima čiji je stepen složenosti takav da omogućava praćenje rešenja bez većeg napora;
- rešenja nekoliko odabranih problema programskim putem na programskom jeziku Java kako bi se razmotrili programerski aspekti u rešavanju ekspertskih zadataka;
- rešenja nekoliko pojednostavljenih praktičnih problema kako bi se Ekspertski sistemi povezali sa praktičnom stranom njihove primene.

U strukturnom pogledu, zbirka je podeljena u četiri poglavlja i dodatak s ukupno sto četrnaest zadataka i to:

- pretraživanje (39 zadatka),
- modeli predstavljanja znanja (47 zadataka),
- strategije rešavanja problema (13 zadataka),
- rad u neizvesnom okruženju (15 zadataka),
- dodatak: prikaz izabranih algoritama (16 algoritama)

Svaki zadatak je identifikovan svojim rednim brojem i imenom koje ima mnenoničku korespondenciju sa problemom koji se rešava. Struktura zadatka je sačinjena od:

- formulacije,
- rešenja

U poglavlju o pretraživanju zastupljeni su svi ključni algoritmi čije su formulacije navedene u dodatku kako bi se lakše pratilo rešavanje problema. Ovo poglavlje sadrži, pored odeljka o pretraživanju, odeljak o pretraživanju u igrama, odeljak o predstavljanju prostora stanja i odeljak sa primerima urađenim na programskom jeziku Java.

Poglavlje o modelima predstavljanja znanja obuhvata sve relevantne modele: formalnu logiku (u okviru koje se razmatra problematika nalaženja konjuktivne normalne forme, skolemizacije, unifikacije, primene rezolucije u zaključivanju, dokazivanja valjanosti formule), produkcione sisteme (zaključivanje direktnim/povratnim/hibridnim ulančavanjem, kompilacija produkcionih sistema, virtuelne činjenice i pamćenje zaključaka), semantičke mreže (nasleđivanje uz prisustvo podrazumevanih vrednosti i if-needed procedura) i okvire (hijerarhija okvira, kvalifikujući pregratci, višestruko nasleđivanje).

Treće poglavlje se bavi strategijama rešavanja problema. Njime je obuhvaćeno planiranje, metode zadovoljavanja ograničenja i sukcesivnih aproksimacija.

U četvrtom poglavlju je razmatran rad ekspertskog sistema u neizvesnom okruženju na osnovu faktora izvesnosti, dok je dat pregled pristupa preko verovatnoća i nenumeričkog izražavanja neizvesnosti, fuzzy logike, kao i sistema za održavanje istinitosti TMS.

Dodatak sadrži algoritme vezane za pretraživanje, pretraživanje u igrama, strategije rešavanja problema i zaključivanje za različite modele predstavljanja znanja.

Autori se nadaju da će jedna ovakva zbirka, pored toga što je udžbeničkog karaktera, poslužiti i onima koji su zainteresovani za upoznavanje oblasti Ekspertskih sistema i Veštačke inteligencije da sagledaju problematiku i osnovne principe ovih oblasti.

Autori se zahvaljuju Vladimiru Petroviću, za dizajn korica, Đorđu Soldu, za predlog zadataka, Nikoli Mihajloviću i Jovanu Bajiću, za pomoć prilikom obrade zbirke.

Na kraju, autori žele da se zahvale recenzentima, dr Milu Tomaševiću i dr Jelici Protić sa Elektrotehničkog fakulteta u Beogradu, i kolegama Zahariju Radivojeviću i Milošu Cvetanoviću za korisne sugestije u konačnom uobličavanju zbirke.

Beograd, jun 2009. god.

Autori

1. Pretraživanje

1.1. Predstavljanje prostora stanja

Zadatak 1: Problem dva krčaga

Na raspolaganju su dva krčaga zapremina 3 i 2 litra bez mernih oznaka. Krčazi mogu da se pune vodom sa česme, a voda može i da se prospe. Potrebno je postići da se u manjem krčagu nađe 1 litar vode.

a) Definisati prostor stanja problema.

b) Definisati operatore koji sistem prevode iz stanja u stanje.

c) Navesti jedan od redosleda primene operatora koji predstavlja rešenje problema.

Rešenje

Prostor stanja je skup svih stanja problema. U konkretnom slučaju količina vode u oba krčaga određuje stanje problema. Prelaz iz stanja u stanje je diskretan i odvija se pod dejstvom *operatora promene stanja* koji odgovaraju zadatom problemu - u konkretnom slučaju to su akcije koje menjaju količinu vode u jednom ili oba krčaga. Rešavanje problema svodi se na to da se odredi niz operatora koji prevodi sistem iz *početnog stanja* (u ovom slučaju oba krčaga prazna) u *završno (ciljno) stanje* (u ovom slučaju - u manjem krčagu 1 litar, u većem proizvoljna količina vode).

a) Stanja se mogu predstaviti uređenim parom (x, y) realnih brojeva x i y pri čemu:

- x predstavlja količinu vode u krčagu od 3 litra,
- y predstavlja količinu vode u krčagu od 2 litra,
- za x i y važe sledeća ograničenja: $0 \leq x \leq 3$, $0 \leq y \leq 2$.

Startno stanje je (0,0), a ciljna stanja su oblika $(x,1)$, pri čemu za x važi gore navedeno ograničenje.

b) Za zadati problem može se definisati 8 operatora promene stanja. Tabela 1 prikazuje definisane operatore.

Kolona *uslov primene* odnosi se na vrednosti koordinata x i y u tekućem stanju (to jest. u onom stanju na koje primenjujemo operator). Tako, na primer, operator `napuni veći krčag iz manjeg' može da se primeni u tekućem stanju samo ako je količina vode u manjem krčagu dovoljna da se veći krčag napuni do vrha (a eventualni višak vode ostaje u manjem krčagu).

c) Jedan od redosleda primene operatora koji vodi do rešenja je: 3,6,1,8,4,5.

redni broj	akcija	tekuće stanje	novo stanje	uslov primene
1.	isprazni veći krčag	(x, y)	$(0,y)$	$x > 0$
2.	isprazni manji krčag	(x, y)	$(x,0)$	$y > 0$
3.	napuni veći krčag iz česme	(x, y)	$(3,y)$	$x < 3$
4.	napuni manji krčag iz česme	(x, y)	$(x,2)$	$y < 2$
5.	napuni veći krčag iz manjeg	(x, y)	$(3,y-3+x)$	$x < 3$ i $y > 0$ i $x+y \geq 3$
6.	napuni manji krčag iz većeg	(x, y)	$(x-2+y,2)$	$x > 0$ i $y < 2$ i $x+y \geq 2$
7.	isprazni veći krčag u manji	(x, y)	$(0,x+y)$	$x > 0$ i $y < 2$ i $x+y \leq 2$
8.	isprazni manji krčag u veći	(x, y)	$(x+y,0)$	$x < 3$ i $y > 0$ i $x+y \leq 3$

Tabela 1

Čitaocu se preporučuje da pronađe niz stanja, počev od početnog (0,0), kroz koja prolazi sistem za sekvencu operatora iz tačke c) rešenja, kao i da odgovori na pitanje da li je navedeno rešenje *optimalno* (da li je broj operatora u sekvenci minimalan).

Definisanje količina vode u krčazima kao realnih brojeva odgovara postavci zadatka, jer je iz česme moguće sipati proizvoljnu količinu vode u krčag (slično važi i kod prosipanja vode i kod presipanja iz jednog krčaga u drugi). Na ovaj način prostor stanja sadrži beskonačno mnogo stanja. Pri izboru operatora, međutim, ograničili smo se na to da se krčazi potpuno napune vodom iz česme, jer je to jedini način da se zna tačna količina vode u njima što je preduslov za nalaženje rešenja. S obzirom na početno stanje i izabrane operatore, jasno je da se u procesu pretraživanja ne mogu pojaviti stanja oblika (x, y) gde x ne uzima neku od vrednosti iz skupa $\{0, 1, 2, 3\}$, a y iz skupa $\{0, 1, 2\}$. Dakle, od svih stanja u procesu pretraživanja dostižan je samo konačan broj stanja. Čitaocu se ostavlja da razmisli da li su i sva stanja definisana ovim strožijim uslovom dostižna.

Pri predstavljanju stanja, prema tome, nije uvek, bez složene analize, moguće izbeći suvišna stanja. Ovo ne predstavlja problem jer u procesu pretraživanja nedostižna stanja ne igraju nikakvu ulogu. Ono što je pri predstavljanju stanja bitnije to je jednoznačno predstavljanje svakog od relevantnih stanja pri rešavanju problema. Da je stanje u ovom zadatku predstavljeno skupom, a ne uređenim parom brojeva x i y, ne bi se, na primer, znalo u kojoj posudi se nalazi kolika količina vode.

Zadatak 2: Hanojske kule

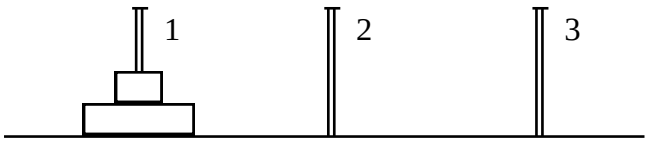
Posmatrajmo igru Hanojskih kula, sa dva diska različitih poluprečnika i tri stuba (slika 1). Cilj igre je da se oba diska sa stuba 1 prebace na stub 3, poštujući pritom sledeća ograničenja:

- u datom trenutku može se pomeriti samo jedan disk, i
- veći disk ne sme ni u jednom trenutku da se nađe iznad manjeg.

Ako sa (x, y) označimo stanje problema, pri čemu je x broj stuba na kojem se nalazi veći, a y broj stuba na kojem se nalazi manji disk, potrebno je:

a) odrediti dozvoljena stanja problema i

b) formirati tabelu dozvoljenih prelaza između stanja u jednom potezu.



Slika 1

Rešenje

a) Dozvoljena stanja su sva ona stanja (x, y) kod kojih je $x, y \in \{1,2,3\}$, dakle to su stanja:

(1,1) (1,2) (1,3) (2,1) (2,2) (2,3) (3,1) (3,2) (3,3).

Početno stanje je (1,1) a ciljno stanje je (3,3).

Primetiti da se kod stanja oblika (x, x) podrazumeva da se manji disk nalazi na većem. O tom ograničenju mora se voditi računa pri definisanju prelaza iz stanja u stanje.

b) Iz stanja (1,1) moguće je pomeriti samo manji disk i to na stub 2 ili stub 3 što odgovara prelascima u stanja (1,2) ili (1,3) respektivno. Iz stanja (1,2) moguće je pomeriti manji disk na stubove 1 ili 3 što odgovara stanjima (1,1) ili (1,3), a veći disk je moguće pomeriti na stub 3 što odgovara stanju (3,2). Veći disk nije, naravno, moguće pomeriti sa stuba 1 na stub 2 u ovom slučaju, jer bi se na taj način veći disk našao na manjem. Na sličan način se pronalaze i ostali dozvoljeni prelazi između stanja. Tabela 2 prikazuje kompletno rešenje. Ukoliko se u ulazu tabele u vrsti (x, y) i koloni (x', y') nalazi ✓ to znači da je dozvoljen prelazak iz stanja (x, y) u stanje (x', y').

	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,3)	(3,1)	(3,2)	(3,3)
(1,1)		✓	✓						
(1,2)	✓		✓					✓	
(1,3)	✓	✓				✓			
(2,1)					✓	✓	✓		
(2,2)				✓		✓			
(2,3)			✓	✓	✓				
(3,1)				✓				✓	✓
(3,2)		✓					✓		✓
(3,3)							✓	✓	

Tabela 2

Zadatak 3: Misionari i ljudožderi

Tri misionara i tri ljudoždera nalaze se na levoj obali reke koju treba da pređu. Na raspolaganju je čamac u koji staju najviše dve osobe. Ako u nekom trenutku broj ljudoždera nadmaši broj misionara na levoj ili desnoj obali, ljudožderi će pojesti misionare. Cilj je da svi bezbedno pređu reku.

a) Predstaviti na pogodan način stanja problema (pri tome uzeti da je čamac uvek na nekoj od obala).

b) Koliki je ukupan broj stanja?

c) Koliko stanja je bezbedno po misionare i koja su to stanja?

Rešenje

U postavci je sugerisano da se sistem posmatra u diskretnim trenucima kada se čamac nalazi uz obalu. Tada se broju misionara i ljudoždera na obali na kojoj je čamac dodaje broj istih iz čamca. Ovakvu pretpostavku je moguće uvesti iz dva razloga:

- u čamcu broj ljudoždera ne može da nadmaši broj misionara i
- ako je stanje sistema bezbedno po misionare kada je čamac uz obalu, ono će biti bezbedno i kada je čamac na sredini reke.

a) U stanju je potrebno imati informaciju o broju misionara i broju ljudoždera na jednoj od obala (na primer levoj). Individualnosti misionara i ljudoždera nisu bitne. Broj misionara i broj ljudoždera na desnoj obali mogu se izračunati oduzimanjem broja na levoj obali od ukupnog broja misionara odnosno ljudoždera, pa ih nije potrebno posebno pamtiti u stanju. U stanju je, pored navedenog, potrebno registrovati još položaj čamca (da li je na levoj ili na desnoj obali). Broj osoba u čamcu ne treba da se registruje posebno u stanju, nego je on pridodat broju ljudi na obali.

Na osnovu izloženog, stanje se može definisati kao uređena trojka (m, lj, $č$) gde je:

- $m \in \{0,1,2,3\}$ broj misionara na levoj obali,
- $lj \in \{0,1,2,3\}$ broj ljudoždera na levoj obali i
- $č \in \{0,1\}$ položaj čamca (0 - leva obala, 1 - desna obala)

b) Pošto prva i druga koordinata stanja mogu uzeti jednu od 4 vrednosti, a treća koordinata jednu od dve vrednosti, ukupan broj različitih stanja je jednak $4 \times 4 \times 2 = 32$.

c) Stanja sigurna po misionare spadaju u dve grupe:

- stanja u kojima su svi misionari na jednoj od obala. Ovakvih stanja je 16:

  (0,0,0) (0,0,1) (0,1,0) (0,1,1) (0,2,0) (0,2,1) (0,3,0) (0,3,1)

  (3,0,0) (3,0,1) (3,1,0) (3,1,1) (3,2,0) (3,2,1) (3,3,0) (3,3,1)

- stanja u kojima se misionari nalaze i na levoj i na desnoj obali. Tada mora biti broj misionara jednak broju ljudoždera na svakoj od obala. Ovakvih stanja ima 4: (1,1,0) (1,1,1) (2,2,0) (2,2,1)

Ukupno ima 20 stanja bezbednih po misionare.

1.2. Algoritmi pretraživanja

Zadatak 4: Misionari i ljudožderi

Koristeći postavku i rezultate zadatka 3 potrebno je:

a) Definisati operatore pretrage.

b) Predstaviti kompletan graf pretrage za dati problem. Da li su sva stanja bezbedna po misionare dostižna iz startnog stanja?

c) Koliko ima optimalnih rešenja problema? Navesti jedno od ovih rešenja.

Rešenje

Koristeći predstavu stanja definisanu u zadatku 3, operatore pretrage potrebno je definisati na taj način da se pretraga odvija isključivo u okviru stanja bezbednih po misionare. Na osnovu grafa pretrage moguće je odgovoriti na ostala pitanja. Optimalnost rešenja u ovom slučaju odnosi se na minimalan broj prelazaka reke.

a) Tabela 3 prikazuje usvojene operatore. Operator *1m* predstavlja prelazak jednog misionara u čamcu sa jedne obale na drugu, operator *1m1lj* označava prelazak jednog misionara i jednog ljudoždera u čamcu, i tako dalje. U koloni za novo stanje važi da je

$k = 1$ ako je $č = 0$,

$k = -1$ ako je č = 1.

U koloni za uslov primene važi da je

$x = m$, $y = lj$ ako je č = 0,

$x = 3\text{-}m$, y = 3- lj ako je č = 1.

Promenljiva x predstavlja broj misionara, a promenljiva y broj ljudoždera na polaznoj obali (to je ona obala na kojoj se pre primene operatora nalazi čamac). Navedenim uslovima obezbeđeno je da se pretraga vrši isključivo u domenu stanja bezbednih po misionare.

oznaka	tekuće stanje	novo stanje	uslov primene
1m	(m,lj,$č$)	(m-k,lj,1-$č$)	(x = 1 ili x-1 ≥ y) i 3-x+1 ≥ 3-y
1lj	(m,lj,$č$)	(m,lj-k,1-$č$)	(3-x ≥ 3-y+1 ili x = 3) i y ≥ 2
2m	(m,lj,$č$)	(m-2k,lj,1-$č$)	(x = 2 i x-2 ≥ y) ili 3-x+2 ≥ 3-y
lj	(m,lj,$č$)	(m,lj-2k,1-$č$)	(3-x ≥ 3-y+2 ili x = 3) i y ≥ 2
1m1lj	(m,lj,$č$)	(m-k,lj-k,1-$č$)	x ≥ 1 i y ≥ 1 i 3-x+1 ≥ 3-y+1

Tabela 3

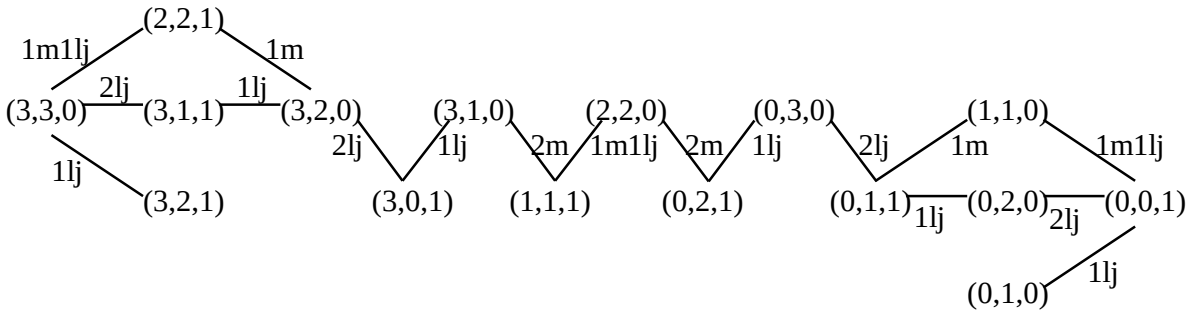
b) Grafom pretrage obuhvataju se sva stanja u prostoru stanja do kojih se može doći iz početnog stanja primenom operatora. Svakom stanju odgovara jedan i samo jedan čvor u grafu pretrage. Ako se prelaz iz stanja X u stanje Y odvija pod dejstvom operatora Op, tada u grafu pretrage postoji orijentisana grana od čvora X do čvora Y obeležena sa Op.

Graf pretrage konstruiše se na sledeći način:

- Odrede se operatori koje je moguće primeniti na startno stanje, kao i stanja u koja se prelazi pod dejstvom ovih operatora - ovaj proces naziva se *ekspanzijom* startnog stanja. U graf se unesu čvorovi koji odgovaraju startnom i novodobijenim stanjima i grane koje odgovaraju pojedinim operatorima.
- Svako od novodobijenih stanja ekspanduje se i pri tome ažurira graf pretrage. Pri ovome se eventualno dobijaju nova stanja. Procedura ekspandovanja se ponavlja sve dok u grafu pretrage postoje stanja koja još nisu ekspandovana.

Slika 2 prikazuje kompletan graf petrage za problem misionara i ljudoždera. Grane ovoga grafa su dvosmerno orijentisane jer isti operator deluje u oba smera promene stanja.

Treba primetiti da se stanja (0,0,0), (3,0,0), (0,3,1) i (3,3,1) ne nalaze u grafu pretrage. Ova stanja, iako bezbedna po misionare, nisu dostižna iz startnog stanja.



Slika 2

c) Rešenje problema predstavlja niz primena operatora na putu od početnog stanja (3,3,0), do ciljnog stanja (0,0,1). Pri traženju rešenja eliminišu se iz razmatranja zatvoreni putevi u grafu. U ovom slučaju samo u početnom i u pretposlednjem koraku pretrage moguć je izbor jednog od dva alternativna operatora dok su ostali operatori jednoznačno određeni. Na taj način definisana su četiri različita optimalna rešenja:

(1) 2lj, 1lj, 2lj, 1lj, 2m, 1m1lj, 2m, 1lj, 2lj, 1lj, 2lj

(2) 1m1lj, 1m, 2lj, 1lj, 2m, 1m1lj, 2m, 1lj, 2lj, 1lj, 2lj

(3) 2lj, 1lj, 2lj, 1lj, 2m, 1m1lj, 2m, 1lj, 2lj, 1m, 1m1lj

(4) 1m1lj, 1m, 2lj, 1lj, 2m, 1m1lj, 2m, 1lj, 2lj, 1m, 1m1lj

Svako od rešenja zahteva 11 prelazaka reke.

U tački a) rešenja, promenljive k, x i y uvedene su da bi se omogućila primena istih operatora u oba smera kretanja čamca (alternativa bi bila da se za svaki smer kretanja uvede poseban operator što bi povećalo broj operatora ali i pojednostavilo izračunavanje uslova primene istih). Razmotrimo na primeru operatora 1m način određivanja uslova za primenu operatora. U tekućem stanju, na polaznoj obali nalazi se x misionara i y ljudoždera, a na odredišnoj obali se nalazi 3-x misionara i 3-y ljudoždera. Pošto se pretraga vrši u okviru bezbednih stanja, na

svakoj od obala u tekućem stanju ili nema misionara ili je njihov broj veći od broja ljudoždera, što se izražava sa

($x = 0$ ili $x \geq y$) i ($3-x = 0$ ili $3-x \geq 3-y$).

Posle primene operatora 1m, na polaznoj obali nalazi se x-1 misionara i y ljudoždera, a na dolaznoj obali $3-x+1$ misionara i $3-y$ ljudoždera. Da bi novo stanje bilo bezbedno po misionare, moraju biti zadovoljeni sledeći uslovi:

- Na polaznoj obali nema više misionara što je ispunjeno ako je $x = 1$ ili
- Broj preostalih misionara mora biti veći od broja ljudoždera, to jest, $x-1 \geq y$. Primetiti da ovi uslovi pokrivaju i ograničenje da u tekućem stanju (pre primene operatora 1m) mora biti bar jedan misionar na polaznoj obali.
- Na dolaznoj obali broj misionara mora biti veći ili jednak broju ljudoždera, to jest $3-x+1 \geq 3-y$ (ovaj uslov nije ispunjen uvek, jer u tekućem stanju ne mora uopšte biti misionara na dolaznoj obali).

Ovi uslovi izraženi su matematičkim formulama u tabeli 0. Čitaocu se preporučuje da prouči uslove primene i za ostale operatore.

Zadatak 5: Hanojske kule

Formulacija problema je ista kao u zadatku 2. Koristeći rezultate zadatka 2:

a) prikazati kompletan graf pretrage za navedeni problem

b) prikazati kompletno stablo pretrage.

c) prikazati stablo pretrage i navesti redosled obilaženja čvorova, ako se za nalaženje rešenja koristi metod traženja po dubini (*depth-first*).

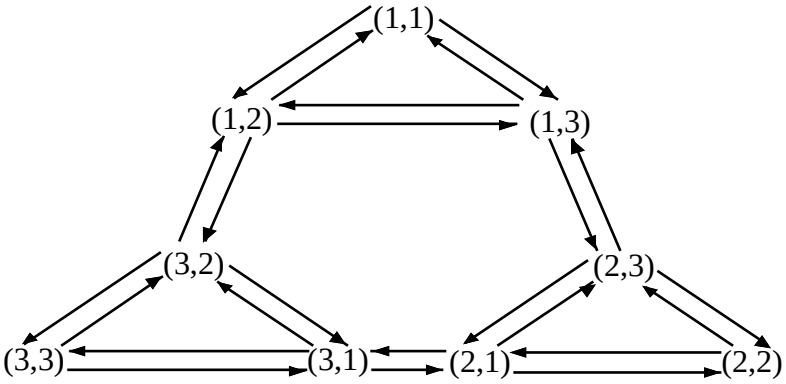
Rešenje

a) Na osnovu tabele 2 u rešenju zadatka 2, metodologijom primenjenom u prethodnom zadatku, lako se dobija traženi graf prikazan na slici 3. Operatori promene stanja nisu eksplicitno naznačeni; oni se mogu odrediti na osnovu čvorova stanja koje povezuju.

b) Kompletno stablo pretrage obuhvata sve otvorene putanje u grafu pretrage koje počinju u startnom čvoru i završavaju se ili u ciljnom čvoru ili u čvoru iz koga svaka dalja primena operatora dovodi do zatvaranja putanje. Ukoliko cilj nije eksplicitno zadat, putanja se završava kada se obiđu sva stanja iz grafa pretrage.

Za putanju u grafu pretrage kaže se da je *zatvorena* ako se na toj putanji dva puta pojavljuje isti čvor, u suprotnom je putanja *otvorena*.

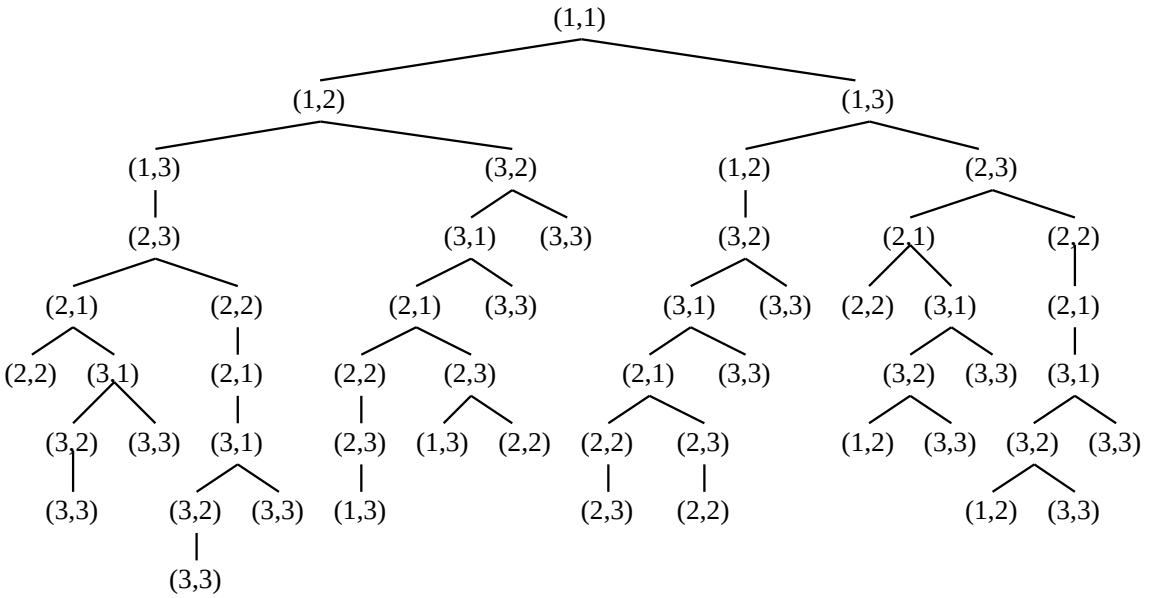
Čvorovi stabla pretrage odgovaraju stanjima, s tim što jednom stanju generalno odgovara više čvorova u stablu. Grane stabla predstavljaju operatore promene stanja. Procedura konstrukcije stabla pretrage je sledeća:



Slika 3

- Startnom stanju odgovara koren stabla pretrage. Po ekspandovanju startnog stanja u stablo pretrage unose se sinovi korenog čvora i odgovarajuće grane, pri čemu svakoj primeni operatora na početno stanje odgovara poseban čvor u stablu i grana koja od korena vodi do tog čvora. Time je koren stabla obrađen.
- Bira se jedan od neobrađenih čvorova u stablu pretrage. Ukoliko je reč o ciljnom čvoru (ako je cilj definisan), nikakva dalja akcija nije potrebna i čvor se može smatrati obrađenim. U suprotnom se ekspanduje stanje koje odgovara izabranom čvoru. U stablo se unose čvorovi koji odgovaraju svakom od stanja dobijenih pri ekspanziji, ukoliko se stanje već nije pojavilo na putanji od korena do ekspandovanog čvora. Proces se ponavlja sve dok u stablu pretrage postoje neobrađeni čvorovi.

Kompletno stablo pretrage za ovaj problem prikazano je na slici 4.

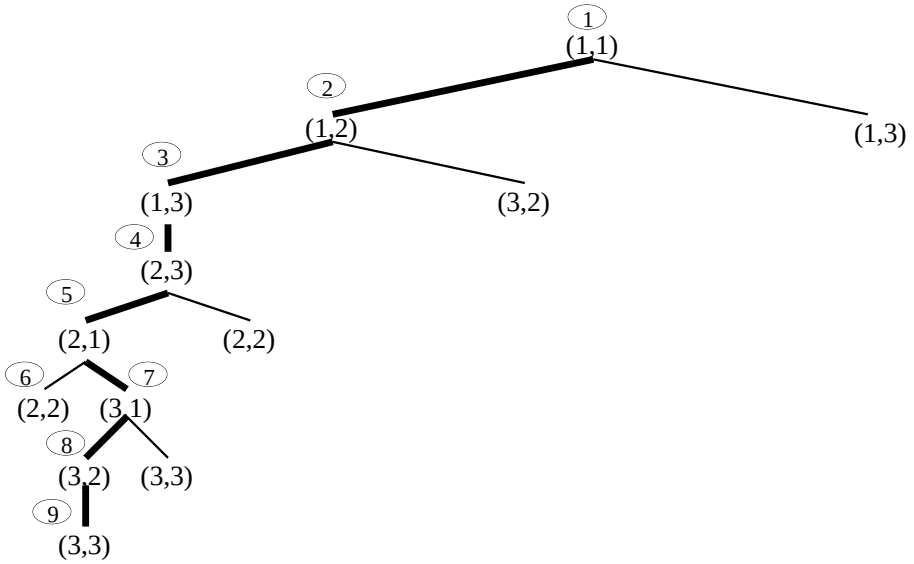


Slika 4

c) Algoritam pretraživanja po dubini naveden je u dodatku 1 (algoritam 2). Stablo pretrage pri pretraživanju po dubini (slika 5) predstavlja deo kompletnog stabla pretrage sa slike 4. Za ekspanziju se uvek (kada postoji) bira levi sin prethodno ekspandovanog čvora. Kada levi sin ne postoji, kontrola se vraća na viši nivo stabla pretrage gde se ekspanduje čvor naslednik koji još nije bio obiđen.

Zaokruženi brojevi na slici 5 prikazuju redosled ekspanzije čvorova koji glasi: (1,1), (1,2), (1,3), (2,3), (2,1), (2,2), (3,1), (3,2), (3,3).

U stablu su prikazani i čvorovi koji nisu obiđeni, to jest ekspandovani, (na primer, desni sin korenog čvora) koji su u stablo uneti kao rezultat ekspanzije čvora-roditelja.



Slika 5

U konkretnom slučaju, pretraživanje po dubini inicijalno prati 'slepu' putanju do čvora (2,2), levog sina čvora (2,1). S obzirom da se iz čvora (2,2) može dospeti samo u čvorove (2,1) i (2,3) koji se već nalaze na putanji, mora se odabrati alternativna putanja. Ovo se realizuje vraćanjem iz čvora (2,2) na roditeljski nivo (engl. *backtracking*) gde postoji mogućnost grananja na alternativnu putanju. Interesantno je primetiti da se pri obilasku čvora (3,1) ne primećuje rešenje u jednom potezu, već se 'slepo' bira levi naslednik. Nađeno je rešenje u sedam poteza, predstavljeno nizom čvorova:

(1,1), (1,2), (1,3), (2,3), (2,1), (3,1), (3,2), (3,3)

na putanji kroz graf pretrage od početnog do ciljnog čvora prikazanoj podebljanim linijama. Navedeno rešenje nije optimalno; iz grafa pretrage sa slike 3 vidi se da optimalno rešenje zahteva samo tri poteza: iz čvora (1,1) premeštanjem manjeg diska dolazi se u čvor (1,2), odatle se premeštanjem većeg diska dolazi u čvor (3,2), a odatle premeštanjem manjeg diska u čvor (3,3).

Metod pretraživanja po dubini koristi se u situacijama kada se traži bilo koje (a ne isključivo optimalno) rešenje. Obratiti pažnju na to da se redosled obilaženja čvorova u grafu pretrage razlikuje od redosleda čvorova koji predstavlja rešenje problema u opštem slučaju. Ovo generalno važi za sve metode pretrage. U idealnom slučaju ova dva redosleda se poklapaju - tada pretraživanje ide striktno po ciljnoj putanji i nema 'lutanja' po alternativnim putanjama. U realnom slučaju obilazi se veći broj čvorova nego što je neophodno da bi se iz početnog stanja došlo u ciljno.

Zadatak 6: Problem dva krčaga

Postavka problema je ista kao u zadatku 1. Potrebno je, na osnovu definisane predstave stanja i operatora:

a) Tabelarno predstaviti kompletan graf pretraživanja za dati problem.

b) Naći rešenje problema koristeći metod pretraživanja po širini (*breadth-first*) - prikazati stablo pretrage i navesti redosled obilaženja čvorova.

c) Naći rešenje problema koristeći metod pretraživanja po širini (*breadth-first*) uz dodatan uslov da se pri pretrazi isto stanje može ekspandovati najviše jednom.

Rešenje

a) Tabela 4 opisuje prelaze između stanja pod dejstvom operatora. Ukoliko u ulazu tabele u vrsti X i koloni Y stoji oznaka Op, to znači da se iz stanja X može preći u stanje Y pod dejstvom operatora Op; ukoliko je odgovarajući ulaz tabele prazan, to znači da se iz stanja X ne može preći u stanje Y primenom samo jednog operatora. Ulazi tabele popunjavaju se na osnovu tabele 1 iz zadatka 1, gde su definisani uslovi primene pojedinih operatora. Tabela se popunjava po vrstama, ekspandovanjem stanja, počev od startnog stanja. Za svako novo stanje dodaju se odgovarajuća vrsta i kolona tabele.

	(0,0)	(3,0)	(0,2)	(3,2)	(1,2)	(2,0)	(1,0)	(2,2)	(0,1)	(3,1)
(0,0)		op3	op4							
(3,0)	op1			op4	op6					
(0,2)	op2			op3		op8				
(3,2)		op2	op1							
(1,2)		op5,8	op1	op3			op2			
(2,0)	op1	op3	op6,7					op4		
(1,0)	op1	op3			op4				op7	
(0,1)	op2		op4				op8			op3
(3,1)		op2		op4				op6	op1	

Tabela 4

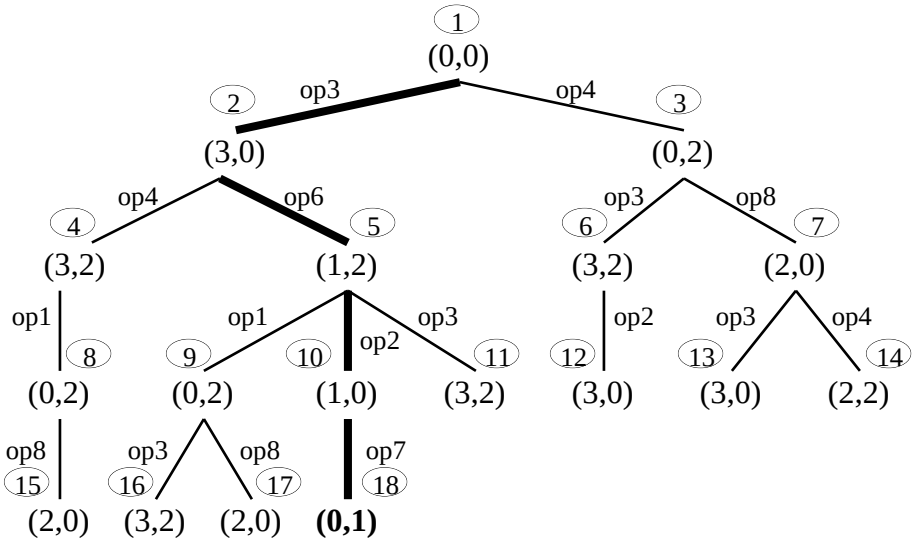
Zanimljivo je primetiti da se prelazi iz stanja (1,2) u stanje (3,0) mogu obaviti kako primenom operatora op5, tako i primenom operatora op8. U pitanju je granični slučaj kada se presipanjem jedan krčag isprazni a drugi napuni vodom. Slično je i sa prelazom iz stanja (2,0) u stanje (0,2).

b) Graf pretrage u slučaju pretraživanja po širini prikazan je na slici 6. Svaki čvor na roditeljskom nivou stabla pretrage mora biti obiđen pre nego što se pređe na nivo potomaka. Algoritam pretraživanja po širini naveden je u dodatku 1 (algoritam 1).

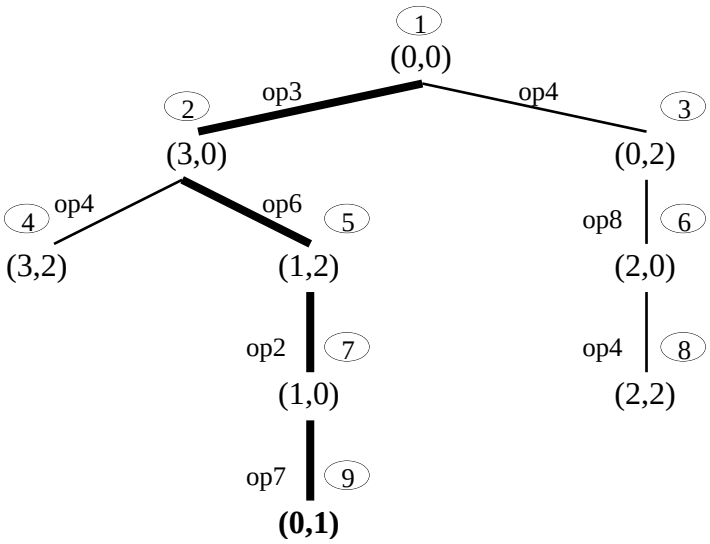
Redosled obilaženja čvorova daju zaokruženi brojevi pored pojedinih stanja. Rešenje problema je predstavljeno nizom od četiri primene operatora: op3, op6, op2 i op7.

c) Stablo pretrage za ovaj slučaj prikazano je na slici 7. Dodatni uslov realizovan je na sledeći način:

Pri ekspanziji čvora proverava se za svakog od potomaka da li odgovarajuće stanje već postoji u stablu pretrage. Ako postoji, taj potomak eliminiše se iz razmatranja. Na primer, u čvoru (3,2) stabla pretrage do koga se dolazi iz startnog stanja primenom operatora op3 pa op4, eliminisan je potomak (0,2) jer je navedeno stanje već ranije obiđeno.



Slika 6



Slika 7

U navedenom slučaju pronađeno je optimalno rešenje, jer se broj primena operatora uzima kao kriterijum minimalnosti. Pretraživanje u širinu spada u metode koje ne garantuju optimalnost rešenja u opštem slučaju (kada operatori imaju različite cene). U slučaju pod c) broj putanja pri pretrazi znatno se redukuje u odnosu na slučaj pod b) jer u slučaju pod c), kada se pronađe jedna putanja do određenog stanja, ostale putanje više nisu od interesa.

Ovakav metod može se primeniti i na druge metode koje traže bilo koje (ne obavezno optimalno) rešenje.

Čitaocu se preporučuje da razmisli o načinu eliminacije zatvorenih putanja u stablu pretrage što NIJE deo algoritama pretraživanja iz dodatka 1. Koja od varijanti b) i c) po tom pitanju zahteva vođenje opširnije evidencije o pređenim putanjama?

Zadatak 7: Tri operatora

Poznato je da za neki problem pretraživanja postoje tri operatora, *op1*, *op2* i *op3*, i da se u polaznom stanju može primeniti bilo koji od njih. Ako prvi primenjeni operator nije bio *op3*, tada se u sledećem koraku može primeniti operator različit od prvog. Svaka dalja primena nekog od operatora nije dozvoljena. Kako cilj nije dat, pretraživanje mora da obuhvati sva moguća stanja.

a) Prikazati graf pretraživanja i označiti čvorove prema redosledu obilaženja, pretpostavljajući da *op1* ima prednost nad *op2*, a ovaj takođe ima prednost nad *op3*. Koristiti strategiju pretraživanja po dubini.

b) Ako se dati graf pretražuje metodom 'prvo najbolji' (*best-first*), navesti redosled obilaženja čvorova. Date su heurističke funkcije za svaki čvor, kao i sekvenca primene operatora koja vodi do čvora:

6 - op1, op2

4 - op1, op3

9 - op2, op1

11 - op2, op3

8 - op1

7 - op2

5 - op3

10 - polazno stanje

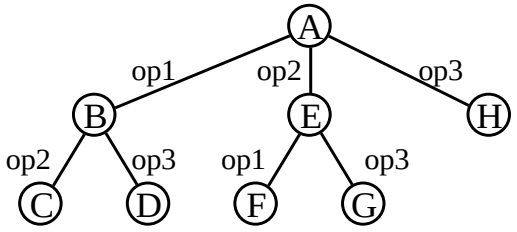
c) Ako bi se pretraživanje sprovelo metodom A*, polazeći od heurističkih funkcija datih pod b), navesti redosled pretraživanja ako su cene primene operatora : 2 za *op1*, 5 za *op2*, i 9 za *op3*.

Rešenje

Iz postavke se može zaključiti da se svakom primenom operatora na neko od postojećih stanja (inicijalno samo startno stanje) dolazi u novo stanje koje je različito od svih postojećih stanja. Uz ovu pretpostavku graf pretrage svodi se na stablo tako da je moguće neposredno primeniti algoritme pretrage bez vođenja računa o eliminaciji zatvorenih puteva. Cilj pretrage je zadovoljen kada, primenom operatora, nije moguće generisati nova stanja. S obzirom da nije moguće više od dva puta sukcesivno primeniti operatore, stablo ima dva nivoa.

a) Stanja će biti obeležavana simbolički. Neka je A oznaka za startno stanje. Stanje A predstavlja koren stabla pretrage prikazanog na slici 8. Na stanje A mogu se primeniti sva tri operatora. S obzirom da operator op1 ima prioritet, njegovom primenom prelazi se u novo

stanje B i pri tom se unosi odgovarajući čvor u stablo pretrage kao sin čvora A. S obzirom da se radi o pretraživanju po dubini, razmatra se čvor B. Mogući operatori su, prema uslovu zadatka op2 ili op3. Operator op2 ima prioritet i njegovom primenom prelazi se iz stanja B u stanje C i unosi odgovarajući čvor u stablo pretrage kao sin čvora B. Na čvor C nije više moguće primeniti nijedan operator prema uslovu zadatka, pa se pretraga vraća u čvor B. Na ovo stanje moguće je primeniti (od do sada neprimenjenih operatora) op3, te se tako dobija novo stanje D i novi čvor u stablu pretrage. Razmatranjem stanja D ustanovljava se da se na njega ne mogu primeniti operatori, pa se kontrola vraća prvo na stanje B, pa zatim, s obzirom da smo ekspandovali sve sinove čvora B, na stanje A. Nadalje se na stanje A primenjuje operator op2 kao najprioritetniji operator koji još nije primenjen čime se dobija stanje E i odgovarajući sin čvora A u stablu pretrage. Procedura se nastavlja sve dok se kompletno stablo ne konstruiše i konstatuje da generisanje novih stanja nije moguće. Redosled obilaženja čvorova u ovom slučaju je, prema tome: A, B, C, D, E, F, G, H.

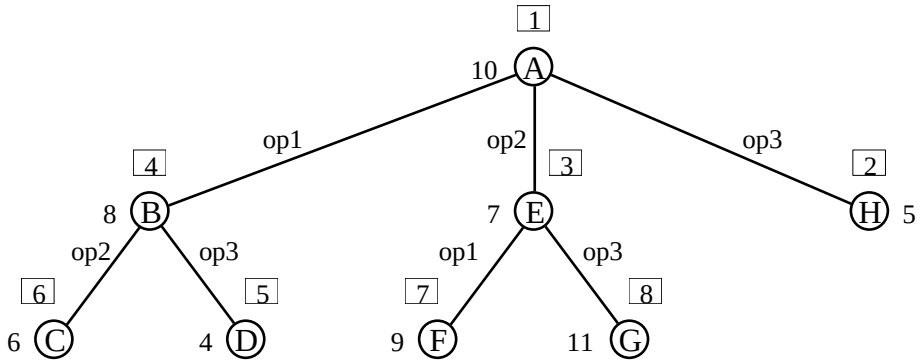


Slika 8

b) Pri pretraživanju metodom 'prvo najbolji' (algoritam 4 u dodatku 1) u stablu pretrage bira se najbolji od svih čvorova stabla pretrage koji su otkriveni, a još nisu obiđeni do određenog trenutka. Kriterijum za selekciju najboljeg čvora je vrednost *heurističke funkcije* koja predstavlja meru udaljenosti određenog stanja problema od konačnog rešenja. Što je vrednost heurističke funkcije za određeno stanje manja, to je stanje bliže ciljnom stanju.

Za zadati problem, stablo pretrage sa ucrtanim redosledom obilaženja čvorova prikazano je na slici 9. Pored svakog čvora stabla pretrage navedena je vrednost heurističke funkcije definisana postavkom zadatka. Pri pretraživanju se, kao i uvek polazi od korena stabla pretrage. Ekspandovanjem korena dobijaju se čvorovi B, E i H. Prvi se obilazi čvor H jer ima najmanju vrednost heurističke funkcije (H=5, E=7, B=8). Pošto čvor H nema naslednika, izbor se svodi na preostala dva otkrivena a neobiđena čvora B i E. Bira se čvor E i vrši njegova ekspanzija čime se u stablo dodaju čvorovi F i G. Sada se vrši izbor između čvorova B, F, i G. Bira se čvor B kao najbolji i vrši njegova ekspanzija. Za izbor sada ostaju čvorovi C, D, F i G. S obzirom da ovi čvorovi nemaju potomaka nema dodavanja novih čvorova. Postojeći čvorovi biće obiđeni u rastućem redosledu njihovih heurističkih funkcija. Kompletan redosled obilaska čvorova stabla pretrage je: A, H, E, B, D, C, F, G.

c) Metod *A** (algoritam 6 u dodatku 1) spada u metode koji pronalaze optimalno rešenje, to jest rešenje sa najmanjom *kumulativnom cenom*. Kumulativna cena za određenu putanju u stablu pretrage jednaka je zbiru cena primene pojedinih operatora na datoj putanji. U opštem slučaju različiti operatori mogu imati različite cene primene, kao što je to slučaj i u ovom zadatku. U ovom zadatku ne traži se nalaženje optimalnog rešenja, već je cilj obići sva stanja. I pored toga, cene putanja utiču na redosled obilaženja čvorova.

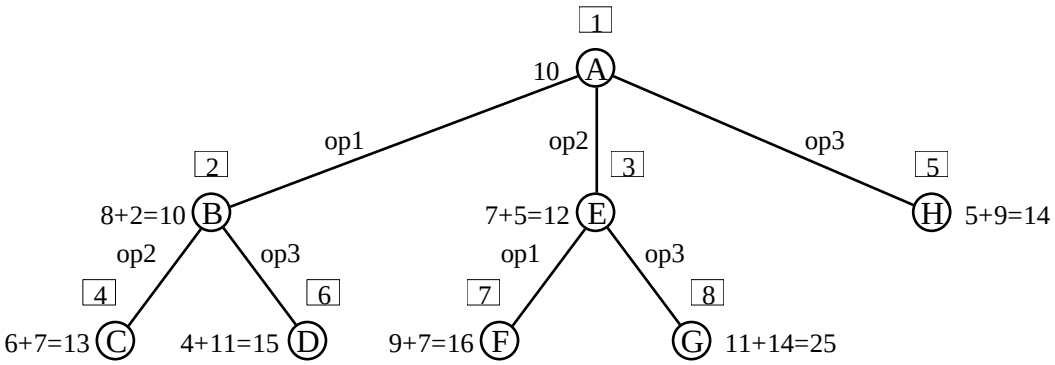


Slika 9

Metod A* kao *funkciju procene f* parcijalne putanje od startnog stanja do nekog drugog stanja S koristi zbir heurističke funkcije *h* za stanje S i funkciju kumulativne cene *c* putanje od početnog stanja do stanja S:

$$f = h + c$$

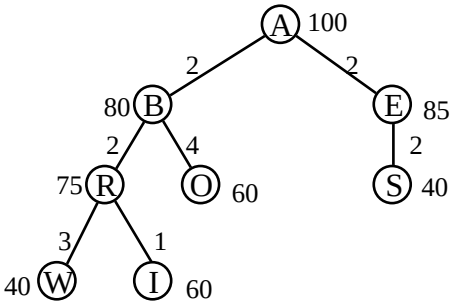
Vrednosti funkcije *h* i cene primene operatora zadate su postavkom za svaki čvor u stablu pretrage. U okviru algoritma iz dodatka 1, tokom pretrage evidentiraju se, ne čvorovi stabla pretrage, već *parcijalne putanje* od korena stabla do otkrivenih čvorova stabla pretrage. U ovom slučaju pretraga počinje sa putanjom nulte dužine od korena stabla (stanja A). Ekspandovanjem korena A, dobijaju se tri nove parcijalne putanje, AB, AE i AH (slika 10). Funkcije procene za ove putanje navedene su pored odgovarajućih čvorova. Od otkrivenih putanja bira se ona sa najmanjom funkcijom procene, a to je u konkretnom slučaju putanja AB. Ekspandovanjem čvora B otkrivaju se dve nove parcijalne putanje ABC i ABD i računaju njihove funkcije procene. U sledećem koraku vrši se izbor najbolje između svih otkrivenih a neobiđenih putanja, a to su u ovom trenutku ABC, ABD, AE i AH. Putanja AE bira se kao najbolja i u sledećem koraku vrši ekspanzija čvora E čime se dobijaju dve nove putanje AEF i AEG. Sada se izbor vrši između putanja ABC, ABD, AEF, AEG i AH i tako dalje, sve dok se ne obiđe kompletno stablo pretrage. Redosled kompletnog obilaska naznačen je na slici.



Slika 10

Zadatak 8: Samoglasnici i suglasnici

Posmatrajmo potpuni graf pretrage prikazan na slici 11. Vrednosti heurističke funkcije prikazane su pored svakog čvora, dok su cene operatora promene stanja prikazane pored grana grafa pretrage.



Slika 11

a) Koje je stanje ekspandovano četvrto u pretraživanju metodom po dubini uz primenu heuristike da stanja čija su imena samoglasnici imaju prednost u odnosu na ostala?

b) Koje je stanje ekspandovano četvrto primenom metode pretraživanja 'prvo najbolji'?

Rešenje

a) Pri ovom pretraživanju ne koristi se (numerički definisana) heuristička funkcija niti cene operatora promene stanja. Heuristika koja daje prednost samoglasnicima nam daje lokalno najbolji čvor među sledbenicima tekućeg čvora u pretrazi (to jest poslednjeg čvora koji je razvijen do tog trenutka). Ovakvoj vrsti heuristike prirodno odgovara pretraživanje po dubini. S obzirom da se vrši eksplicitan izbor među sledbenicima tekućeg čvora pretrage, radi se u stvari o pretraživanju metodom planinarenja (algoritam 3 u dodatku 1).

Pretraga počinje od čvora A koji je inicijalno jedini čvor u stablu pretrage. Razvijanjem čvora A dobijaju se njegovi sledbenici, čvorovi B i E koji se dodaju, zajedno sa operatorima koji vode do njih u stablo pretrage. Heuristika daje prednost čvoru E, tako da se on razvija sledeći i pri tome u stablo pretrage dodaje čvor S. Čvor S se razvija sledeći pošto je jedini naslednik čvora E. Pošto čvor S nema naslednika, pretraga je zapala u 'ćorsokak', pa s obzirom da čvor E nema drugih naslednika osim S, pretraga se nastavlja od čvora B kao drugog naslednika korenog čvora A. Prema tome, četvrti ekspandovani čvor, to jest četvrto stanje čiji su sledbenici dobijeni pri pretrazi, je čvor B.

b) Pri ovoj pretrazi koristi se numerički definisana heuristička funkcija ali ne i cene operatora promene stanja. Za razliku od planinarenja, gde se izbor sledećeg čvora za razvijanje vrši lokalno među naslednicima poslednjeg ekspandovanog čvora, u slučaju pretrage 'prvo najbolji' izbor se vrši globalno među svim čvorovima koji su uneti u stablo pretrage ali nisu razvijeni.

Pretraga polazi od čvora A, njegovim razvijanjem u stablo pretrage se unose čvorovi B i E. Između čvorova B i E za ekspanziju se bira B jer ima manju vrednost heurističke funkcije (80 naspram 85). Razvijanjem čvora B, u stablo pretrage unose se čvorovi R i O. Izbor se u ovom trenutku vrši između čvorova R, O i E kao jedinih neekspandovanih čvorova koji se nalaze u stablu pretrage u tekućem trenutku. Izbor pada na O jer ima najmanju vrednost heurističke funkcije. Razvijanjem čvora O ne unose se novi čvorovi u stablo pretrage, pa kandidati za

razvijanje ostaju samo čvorovi R i E. Heuristička funkcija diktira izbor čvora R kao četvrtog za razvijanje pri ovoj pretrazi.

Zadatak 9: Viktorija

Poznata dečja igra Viktorija je vrsta slagalice u kojoj se na tabli nalazi niz pločica sa brojevima, kao na slici 12. Jedno polje table je prazno, to jest nije pokriveno pločicom. U svakom potezu moguće je pomeriti jednu od pločica sa brojem koje su susedne praznom polju, na mesto praznog polja. Cilj igre je složiti pločice u neki karakterističan raspored. Konstruisati stablo pretraživanja koristeći metod A*, ako iz početne pozicije prikazane na slici 12 treba preći u završnu poziciju sa iste slike. Kao procenu rastojanja od ciljne pozicije koristiti funkciju

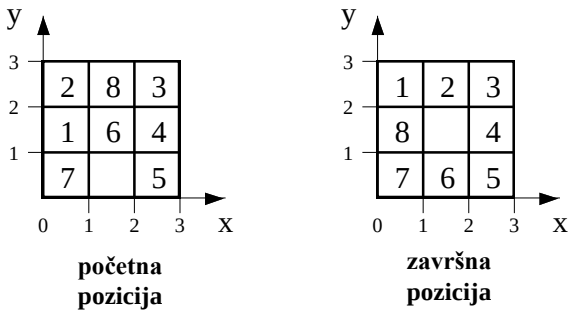
$$h = \sum_{i=1}^{8} MD_i + 3* \sum_{i=1}^{8} pt_i$$

gde MD_i predstavlja *Manhattan rastojanje* (zbir apsolutnih vrednosti razlika x i y koordinata, $h' = |\Delta x| + |\Delta y|$) broja i u tekućoj poziciji od ciljne pozicije, a pt_i koeficijent za merenje uređenosti brojeva u tekućem stanju u odnosu na ciljno stanje koji se računa na sledeći način:

- ako je broj i u tekućem stanju u centru, $pt_i = 1$.

- ako broj i nije u centru, a iza njega (u smeru kazaljke na satu) se nalazi onaj broj koji je njegov sledbenik i u ciljnoj poziciji, $pt_i = 0$.

- ako broj i nije u centru, a iza njega se ne nalazi njegov sledbenik iz ciljne pozicije, $pt_i = 2$.



Slika 12

Rešenje

Algoritam A* za procenu pozicija koristi funkciju procene f koja predstavlja zbir heurističke funkcije h i funkcije kumulativne cene c.

U postavci nije eksplicitno naglašeno, ali je za ovaj problem prirodno uzeti da svaki potez u igri ima jediničnu cenu, tako da je kumulativna cena c određene pozicije P jednaka broju poteza na putanji između početne i pozicije P. Koristeći algoritam A*, dobija se stablo pretrage prikazano na slici 13. Uz svako stanje prikazana je vrednost funkcije f kao zbir funkcija h i c. Za krajnje levog sina korenog čvora, na primer, funkcija f izračunava se na sledeći način:

- Zbir menhetn rastojanja pojedinih brojeva je

$$\sum_{i=1}^{8} MD_i = (0+1)+(1+0)+(0+0)+(0+0)+(0+0)+(0+0)+(0+0)+(1+1) = 4$$

  Iz prethodne formule može se videti da je, na primer, Menhetn rastojanje za broj 8, (poslednja zagrada u zbiru), jednako 2 jer je apsolutna udaljenost broja 8 u tekućoj poziciji i po x i po y osi jednaka 1 u odnosu na ciljnu poziciju.

- Zbir koeficijenata uređenosti brojeva je:

$$\sum_{i=1}^{8} pt_i = 0+2+0+0+0+0+2+2 = 6$$

  jer se u navedenom stanju brojeve 2, 7 i 8 ne prate njihovi sledbenici (3, 8 i 1 respektivno) u ciljnom stanju, a u sredini se nalazi praznina. Heuristička funkcija *h* za ovo stanje iznosi, prema tome, *h* = 4 + 3*6 = 22.

- Cena pozicije *c* = 1, jer se do ove pozicije dolazi u jednom potezu iz početne pozicije. Prema tome funkcija *f* = 22 + 1 = 23.

Može se videti da je, zahvaljujući dobrom izboru heurističke funkcije, pretraga sve vreme pratila ciljnu putanju, radi se dakle o idealnom slučaju. Ovo ne mora biti ispunjeno za neki drugi par početnog i ciljnog stanja.

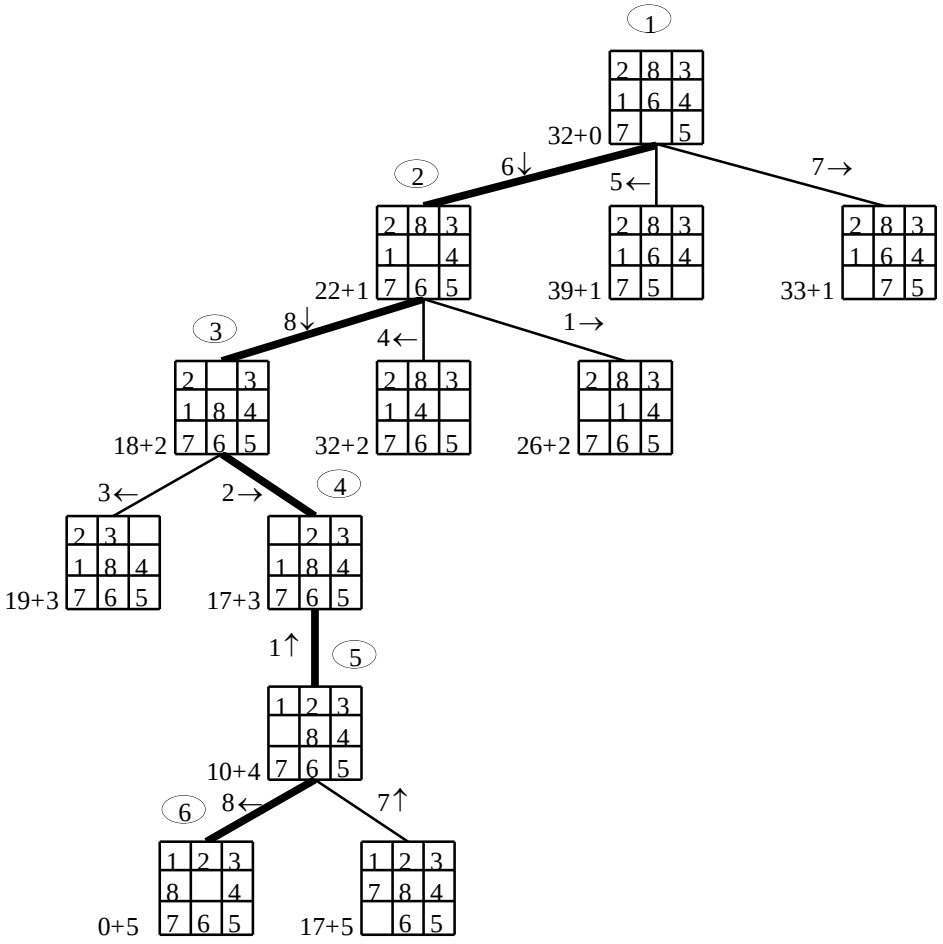
Rešenje nije uvek moguće dobiti za svaku kombinaciju početne i ciljne pozicije jer ciljna pozicija nije uvek dostižna iz početne. Sve moguće pozicije u ovoj igri (koliki je njihov broj?) dele se u dve nezavisne *klase parnosti*, tako da je rešenje moguće naći samo ako su i početna i ciljna pozicija iste klase parnosti. Suština pojma klasa parnosti može se sagledati iz sledećeg: Posmatrajmo pozicije u igri na polju 2 x 2. U ovom slučaju ima ukupno 4! = 24 različite pozicije podeljene u dve klase. Odgovarajući grafovi prelaza prikazani su na slici 14. Veze između čvorova su dvosmerne.

Provera da li početna i ciljna pozicija pripadaju istoj klasi parnosti, odnosno, da li je problem rešiv ili nije za proizvoljnu veličinu polja, može se izvršiti na sledeći način:

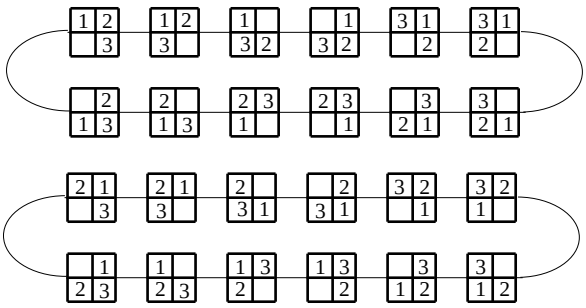
1. Iz početne pozicije, uz poštovanje pravila igre, doći u novu poziciju u kojoj se prazno polje nalazi na istom mestu kao i u ciljnoj poziciji.

2. Svako mesto u novoj poziciji, počev od gornjeg levog ugla zaključno sa donjim desnim, obraditi na sledeći način: ako broj X na tom mestu nije jednak broju Y na istom mestu u ciljnoj poziciji, zameniti u posmatranoj poziciji mesta brojevima X i Y (zamena je direktna, ne po pravilima igre).

3. Ako je broj zamena u tački 2. bio paran, pozicije su u istoj klasi parnosti. U suprotnom, pozicije nisu u istoj klasi i zadatak nema rešenja.

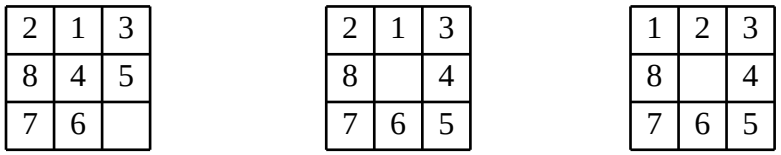


Slika 13



Slika 14

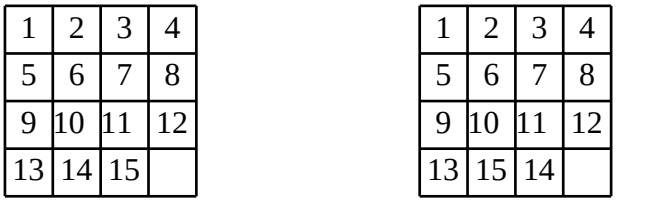
Kao primer primene ovog algoritma, na slici 15 prikazana je situacija u kojoj se početna pozicija (a) transformiše, pomeranjem broja 5 na dole i broja 4 na desno, u poziciju (b). Ova pozicija se svodi na ciljnu (c) primenom samo jedne zamene (brojeva 1 i 2) što znači da početna i ciljna pozicija ne pripadaju istoj klasi parnosti. Ukoliko se u program za pretraživanje ne ugradi ispitivanje parnosti početne i ciljne pozicije, algoritam pretrage bi detektovao da je problem nerešiv tek pošto obiđe sve dostižne pozicije prostora stanja.



(a) (b) (c)

Slika 15

Zanimljivo je da je, početkom ovog veka, Semjuel Lojd u Sjedinjenim Američkim Državama, lansirao varijantu ove igre na polju 4 x 4 pri čemu je nudio nagradu od 50 000 dolara onome ko nađe rešenje za početnu i ciljnu poziciju prikazanu na slici 16. Nagradu, naravno, niko nije dobio, ali je to bila odlična reklama za prodaju ove igre.



(a) početna pozicija (b) završna pozicija

Slika 16

Zadatak 10: Putna mreža (Menhetn norma)

Na slici 17 prikazan je deo putne mreže, a zadatak je da se nađe put između tačke *a* i tačke *h*. Razdaljine su date razmerom.

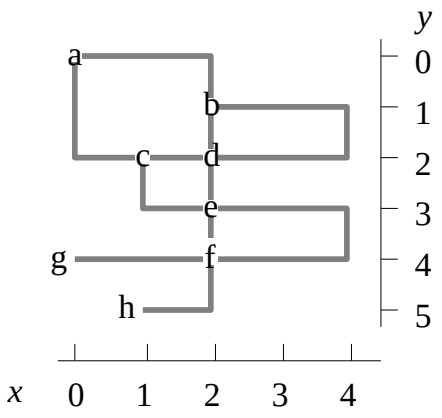
a) Prikazati stablo pretraživanja ukoliko se koristi metoda A*, pri čemu se udaljenost do cilja meri Manhattan normom ($\Delta s=\Delta x+\Delta y$), a cena određene parcijalne putanje od početnog čvora do čvora *X* jednaka je dužini puta od početnog čvora do čvora X po datoj putanji.

b) Prikazati stablo pretraživanja, ukoliko se ne uzima u obzir udaljenost do cilja, već samo cena, odnosno pređeni put. O kojoj se vrsti pretraživanja radi?

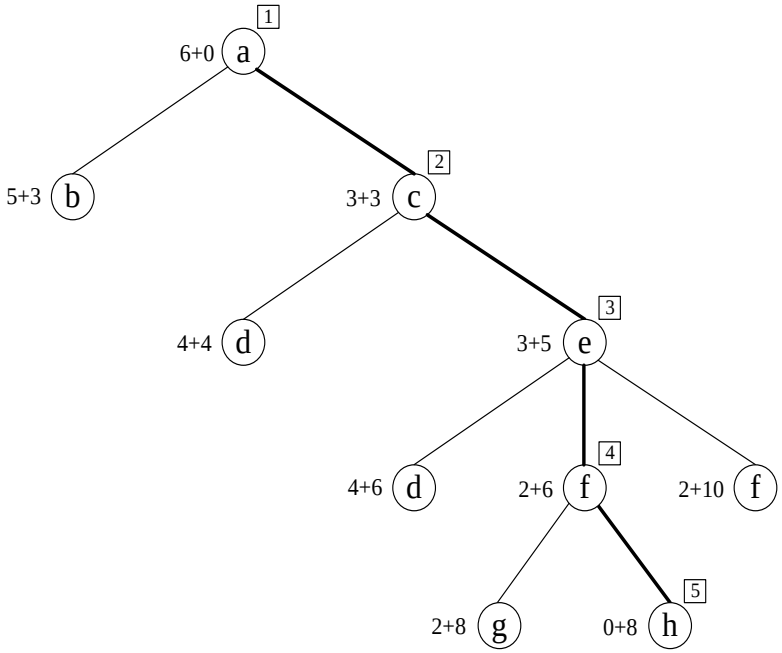
c) Prikazati stablo pretraživanja, ukoliko se ne uzima u obzir pređeni put, već samo udaljenost do cilja. O kojoj se vrsti pretraživanja se radi u ovom slučaju?

Rešenje

a) Stablo pretrage prikazano je na slici 18. U konkretnom slučaju pretraga sve vreme sledi najkraću putanju (ukoliko više od jednog čvora ima minimalnu vrednost kumulativne cene, bira se čvor sa manjom vrednošću heurističke funkcije).



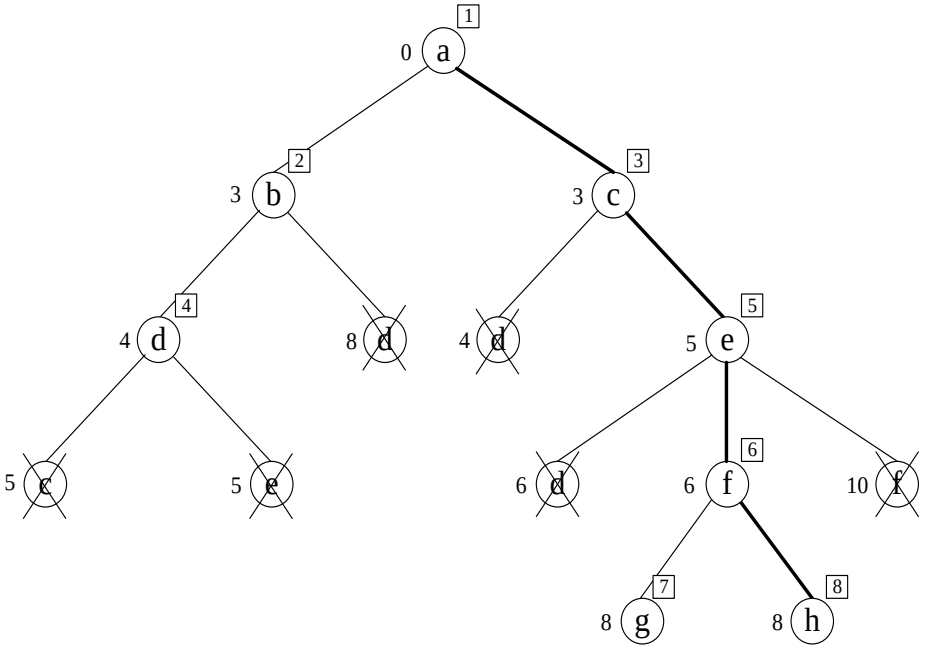
Slika 17



Slika 18

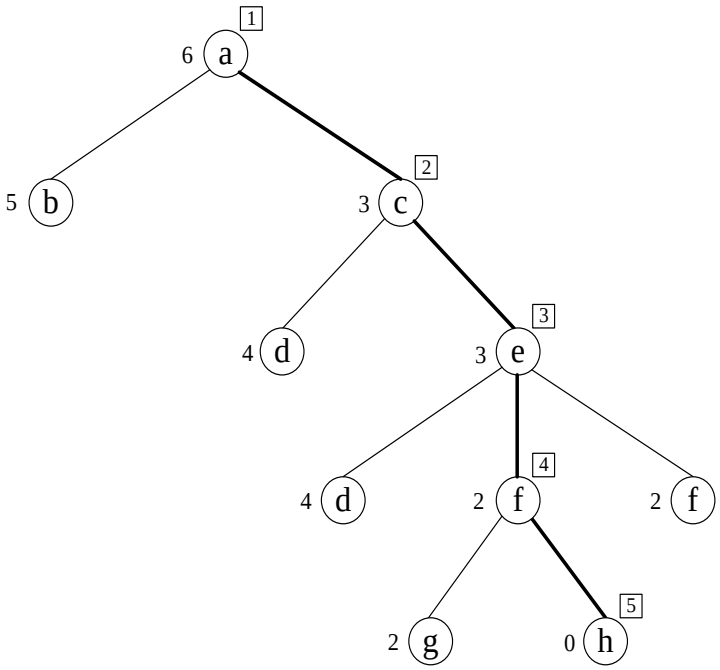
b) Ukoliko se algoritam A* modifikuje na taj način da se od svih parcijalnih putanja koje se razmatraju bira ona koja je najkraća, dobija se metod *grananja i ograničavanja* sa dinamičkim programiranjem. Pod *dinamičkim programiranjem* podrazumeva se izbacivanje iz razmatranja svih parcijalnih putanja koje vode do istog stanja osim one najkraće.

Na slici 19 prikazano je stablo pretrage za ovaj slučaj. Pored svakog čvora u stablu pretrage označena je dužina parcijalne putanje od početnog do tog čvora (kumulativna cena). Čvorovi stabla pretrage eliminisani primenom principa dinamičkog programiranja precrtani su na slici 19. Ukoliko više putanja ima jednaku kumulativnu cenu, vrši se izbor na osnovu leksikografskog poretka poslednjih čvorova na putanjama.



Slika 19

c) U ovom slučaju radi se o metodu 'prvo najbolji'. Stablo pretrage (slika 20) u konkretnom slučaju je identično stablu pretrage za metod A*. Na slici 20 uz svaki čvor označene su vrednosti heurističke funkcije za taj čvor.



Slika 20

Zadatak 11: Planinarenje

Na slici 21 dati su podaci o visini u svakoj tački terena pod gustom maglom. Planinar sa visinomerom koji se nalazi u tački A1 može u jednom trenutku da korakne u pravcu jedne od strana sveta.

a) Kojim putem će planinar doći u najvišu tačku terena metodom planinarenja (*hill-climbing*)? Planinar bira najvišu od susednih tačaka. U slučaju postojanja više susednih tačaka iste visine, prioritet ima pravac severa, pa istoka, pa juga i na kraju zapada.

b) Prikazati stablo pretraživanja metodom A* ako heuristička funkcija predstavlja razliku visina pomnoženu Manhattan rastojanjem ciljne i tekuće tačke, a cena je broj koraka koje planinar načini. Koji je najkraći put do cilja?

E	0	1	2	4	3
D	1	1	3	5	4
C	3	2	2	3	2
B	2	4	2	1	0
A	0	2	1	0	0
	1	2	3	4	5

Slika 21

Rešenje

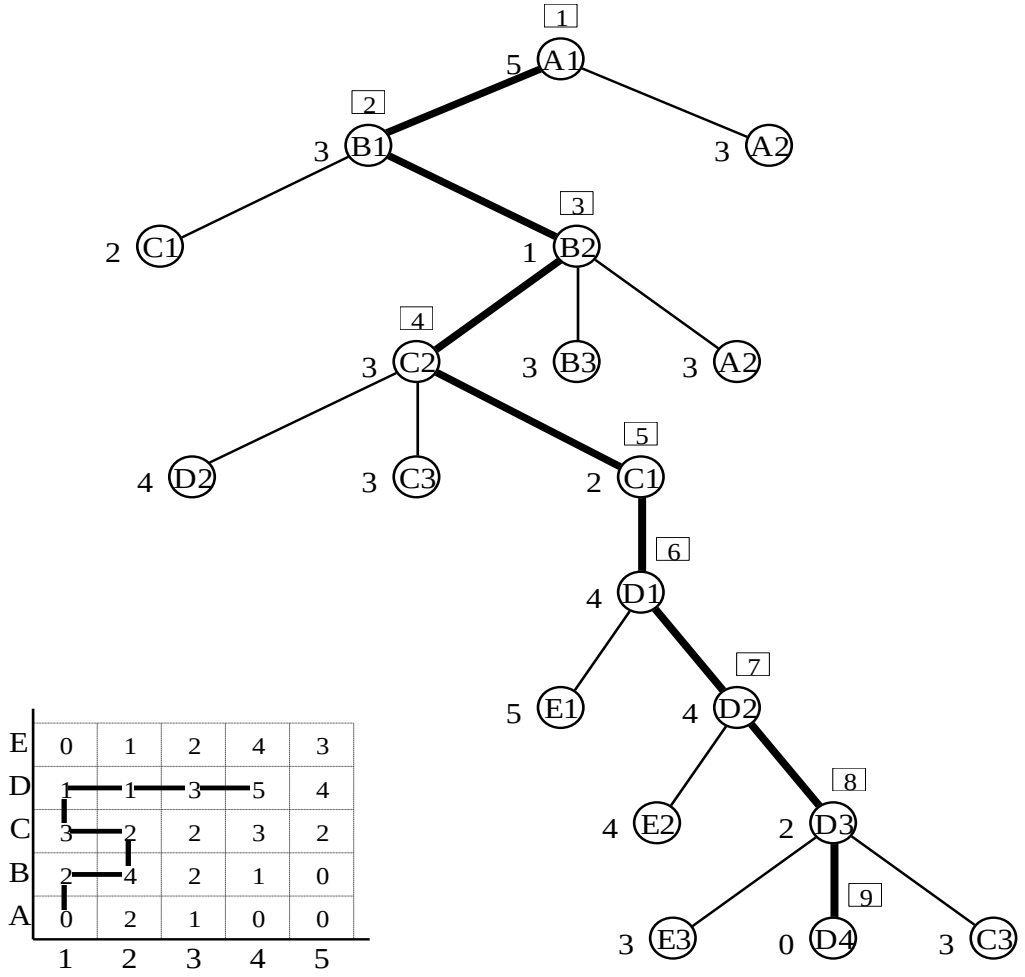
Najviša tačka terena je D4, te ona prema tome predstavlja ciljnu tačku. Da bi se ispunio zahtev iz tačke a) postavke da prioritet pri pretrazi ima najviša od susednih tačaka, heurističku funkciju *h* za proizvoljnu tačku X definisaćemo kao razliku visina ciljne tačke D4 i tačke X:

$$h = 5 - H_x$$

gde H_X predstavlja visinu tačke X. Treba primetiti da se pravilima navedenim u postavci dodaje i pravilo da se planinar ne vraća u tačku u kojoj je bio da bi se izbeglo da se planinar vrti u krug (planinar može sebi da crta mapu kretanja).

a) Stablo pretrage i mapa kretanja planinara prema algoritmu 3 iz dodatka 1 prikazani su na slici 22. Uz čvorove stabla pretrage (levo od čvorova) navedene su odgovarajuće vrednosti heurističke funkcije. Najpre se ekspanduje startni čvor A1, tj. u stablo pretrage dodaju se čvorovi B1 i A2, pri čemu je vrednost heurističke funkcije oba čvora jednaka 3. Kako je postavkom navedeno da prioritet najpre dobija sever, zatim istok, jug i na kraju zapad, sledeći za ekspandovanje odabira se čvor B1. Prateći dalje algoritam, jednostavno je konstruisati ostatak stabla pretrage.

U ovom slučaju pretraga sve vreme ide po ciljnoj putanji (brojevi u kvadratima prikazuju redosled ekspanzije čvorova), ali nađeno rešenje nije optimalno u smislu minimalnog broja koraka.



Slika 22

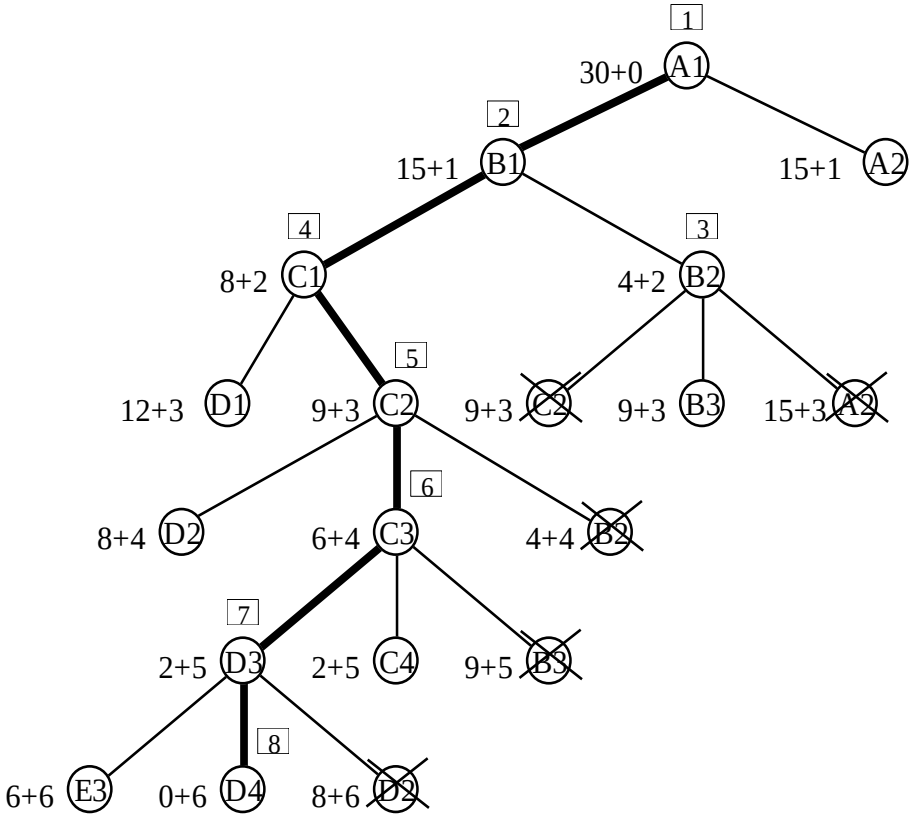
b) Stablo pretrage prikazano je na slici 23. Razmotrimo računanje vrednosti heurističke funkcije i cene na primeru: cena putanje A1 B1 B2 A2 je 3 jer planinar načini tri koraka da bi iz tačke A1 dospeo u tačku A2. Heuristička funkcija za ovu putanju je $h = 15$ jer je menhetn rastojanje MD tačke A2 od ciljne tačke D4 jednako apsolutnoj vrednosti razlike x i y koordinata ove dve tačke, to jest, MD = (4 - 2) + (4 - 1) = 5, a razlika visina tačaka A2 i D4 je $\Delta h = 5 - 2 = 3$. Na osnovu toga, funkcija procene f = h + c za ovu putanju jednaka je f = 15 + 3 = 18.

Treba primetiti da su, u slučaju b), na slici 23 neke putanje eliminisane iz daljeg razmatranja - ovim putanjama odgovaraju precrtani čvorovi u stablu pretrage. Eliminacija je sprovedena po principu *dinamičkog programiranja* koji je deo algoritma A* a sastoji se u tome da se iz pretrage eliminišu sve parcijalne putanje koje dovode do istog stanja osim putanje sa najmanjom cenom. Na primer, putanja A1 B1 B2 A2 eliminisana je iz razmatranja jer je već ranije nađena putanja A1 A2 do stanja A2 koja je kraća od prethodno navedene.

Kao rezultat pretrage metodom A* nađena je jedna od minimalnih putanja. Radi se o putanji:

A1-B1-C1-C2-C3-D3-D4.

Koliko još ima različitih minimalnih putanja?



Slika 23

Zadatak 12: Putna mreža (razni algoritmi pretrage)

Na slici 24 je prikazana mreža puteva sa označenim dužinama puteva u kilometrima. Tabela 5 prikazuje vazdušna rastojanja od pojedinih gradova do grada G u kilometrima. Prikazati stablo pretrage i navesti redosled obilaženja čvorova pri pretrazi za nalaženje puta između gradova S i G ako se koristi :

a) pretraga po dubini (*depth-first*)

b) pretraga po širini (*breadth-first*)

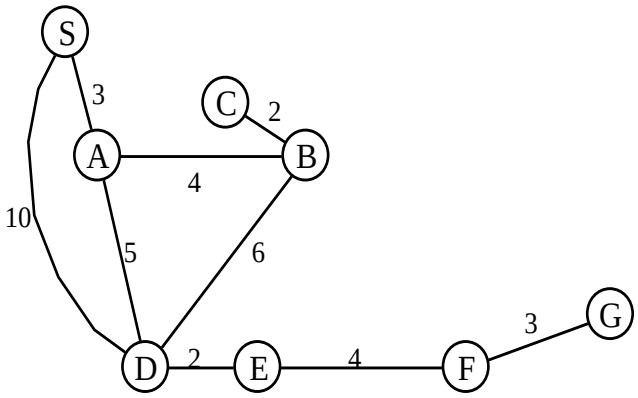
c) planinarenje (*hill-climbing*)

d) prvo najbolji (*best first*)

e) grananje i ograničavanje (*branch and bound*)

f) A*

Napomena: definisati na pogodan način heurističku funkciju i cenu rešenja za metode kojima su ove veličine potrebne.



Slika 24

Grad	S	A	B	C	D	E	F
Rastojanje do G	11.5	10.4	6.7	7.0	8.9	6.9	3.0

Tabela 5

Rešenje

Za metode c), d) i f) koji koriste heurističku funkciju vazdušno rastojanje od tekućeg grada do grada G može se koristiti kao procena udaljenosti do cilja (radi se o potcenjenoj veličini). U metodima e) i f) funkcija cene pri prelasku iz jednog grada u drugi odgovara dužini puta između ova dva grada naznačenoj na mapi.

a) Stablo pretrage po dubini prikazano je na slici 25. Redosled obilaženja čvorova daju brojevi u kvadratima; iz čvora C kontrola je vraćena na drugog naslednika čvora B. Nađeni put

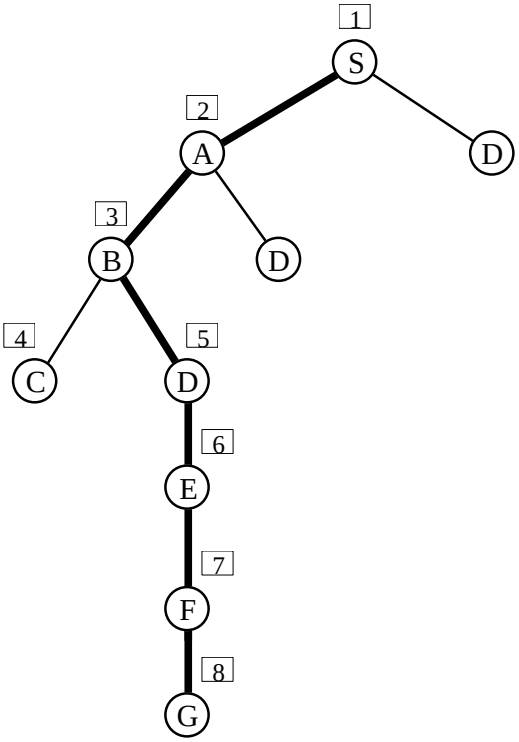
S-A-B-D-E-F-G

ima dužinu 22 km, što nije najkraći put između gradova S i G.

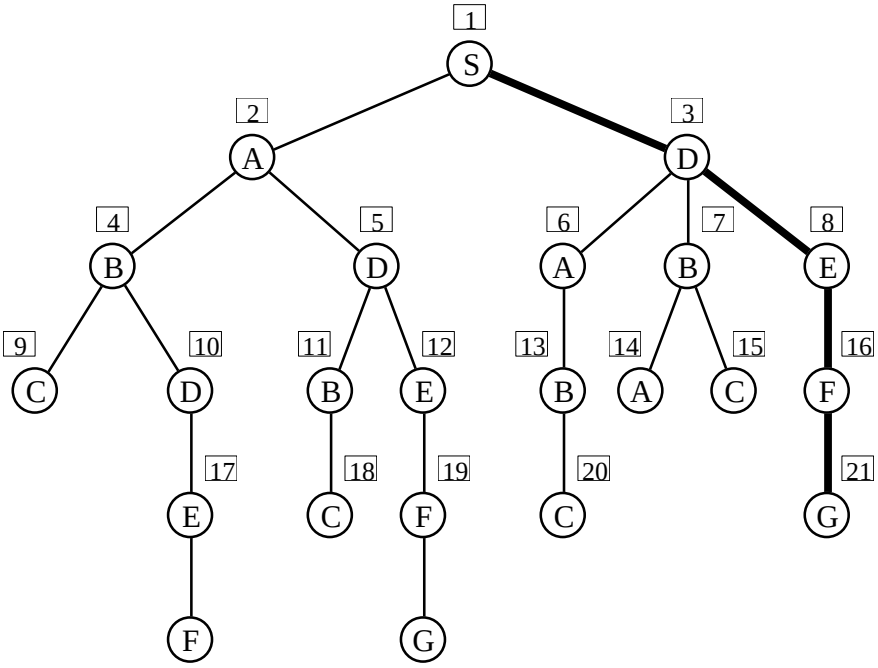
b) Stablo pretrage po širini prikazano je na slici 26. Stablo pretrage samo je malo manje od kompletnog stabla pretrage za ovaj problem - nije otkriven samo čvor G, naslednik čvora F na putanji S-A-B-D-E-F. Nađeni put dužine 19 km

S-D-E-F-G

prolazi kroz najmanji broj gradova, ali nije najkraći mogući.



Slika 25

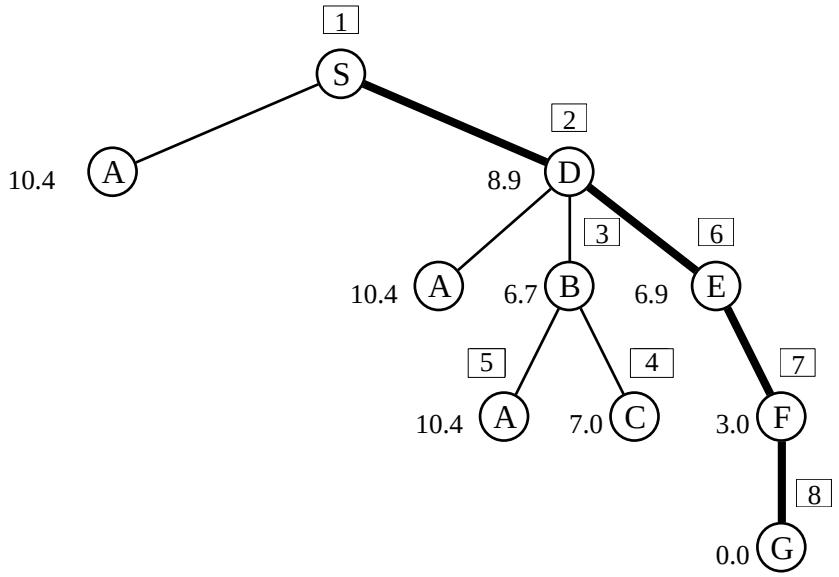


Slika 26

c) U slučaju pretrage metodom planinarenja (slika 27) nalazi se ista putanja kao pri pretrazi po širini. Ovaj primer lepo ilustruje nedostatak metode izbora najboljeg (po vrednosti heurističke funkcije procene) čvora lokalno, to jest, među sinovima tekućeg čvora:

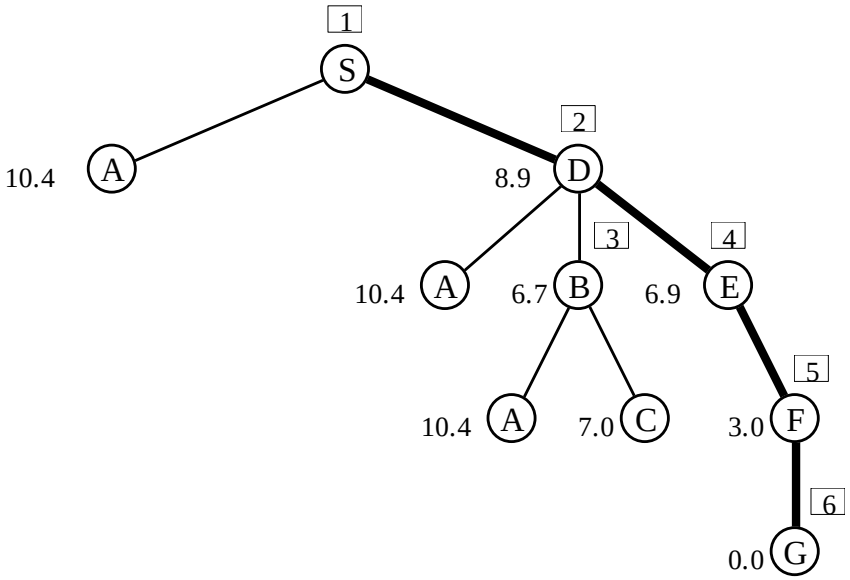
Po razvijanju čvora S, bira se čvor D u skladu sa heurističkom funkcijom čije su vrednosti navedene pored čvorova. Po razvijanju čvora D, bira se B kao najbolji. Razvijanjem čvora B,

dobijaju se čvorovi A i C. Oba ova čvora su lošiji od već otkrivenog čvora E, sina čvora D, ali čvor E neće biti izabran dok se ne obiđu i A i C jer je izbor čvorova lokalan. Primetiti da čvor A na putanji S-D-B-A nije dalje razvijan u stablu pretrage, jer bi to dovelo do stvaranja zatvorenih putanja.



Slika 27

d) U slučaju primene metoda 'prvo najbolji' (slika 28) nađeno rešenje je isto kao u prethodnom slučaju. S obzirom da se kod ovog metoda najbolji čvor bira globalno, među svim otkrivenim a neobiđenim čvorovima, izbegava se obilaženje sinova A i C čvora B, to jest, manje se skreće sa ciljne putanje nego u prethodnom slučaju.



Slika 28

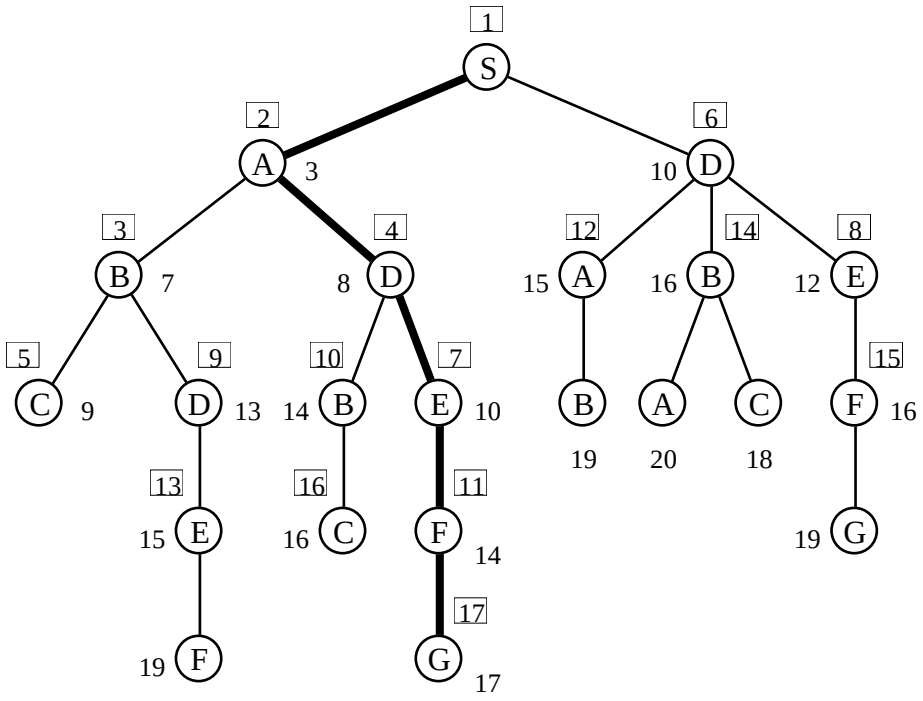
e) Na slici 29 prikazano je stablo pretrage po metodu *grananja i ograničavanja* (algoritam 5 u dodatku 1). U ovom slučaju nađeno je optimalno rešenje problema:

S-A-D-E-F-G

Ova putanja je dužine 17 km.

Stablo pretrage obuhvata, kao i u slučaju pretrage po širini, gotovo kompletno stablo pretrage. U ovom slučaju bira se parcijalna putanja najmanje dužine. Dužine pojedinih putanja prikazane su uz čvorove stabla pretrage (ne zaokruženi brojevi). Zaokruženim brojevima predstavljen je redosled obilaženja čvorova pri pretrazi.

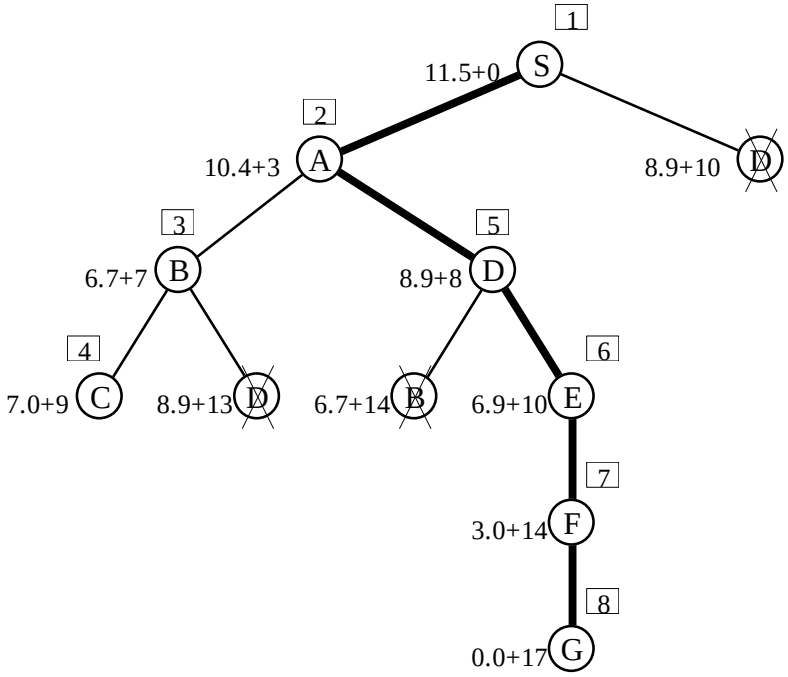
Interesantno je primetiti da se pretraga u ovom slučaju ne završava kada se ekspanduje čvor F na putanji S-A-D-E-F i pri tome otkrije ciljni čvor G. Pre završetka pretrage potrebno je bilo produžiti sve otkrivene parcijalne putanje kraće od 17, koliko iznosi dužina nađene putanje do G, do njihovog završetka ili do čvorova gde dužina putanje prelazi 17. Moguće je da se ovakvim produžavanjem nađe nova, kraća putanja do čvora G što se u ovom slučaju nije desilo.



Slika 29

f) Pretraživanjem metodom A* (slika 30) dobija se ista minimalna putanja kao i u prethodnom slučaju. Za razliku od prethodnog slučaja, odstupanje od ciljne putanje pri pretrazi je relativno malo zahvaljujući korišćenju heurističke funkcije. Precrtani čvorovi na slici 30 reprezentuju parcijalne putanje uklonjene po principu dinamičkog programiranja iz daljeg razmatranja.

Ovaj zadatak dobro ilustruje komparativne prednosti i nedostatke pojedinih metoda, kako onih u klasi traženja bilo kog rešenja - metodi a), b), c) i d), tako i onih u klasi traženja optimalnog rešenja - metodi e) i f). Za određeni problem, generalno su bolji oni metodi koji za isti graf pretrage i uz korišćenje istih funkcija procene stanja i cene operatora promene stanja, dolaze do rešenja obilaženjem manjeg broja čvorova stabla pretrage.



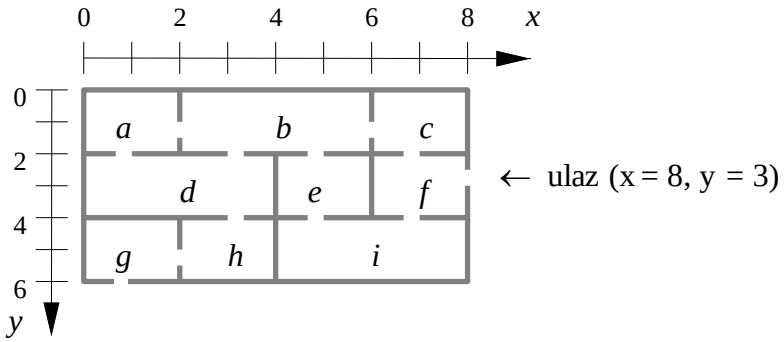
Slika 30

Zadatak 13: Džems Bond

Na slici 31 prikazan je tlocrt jedne kuće, u kojoj se, u prostoriji *g*, nalazi garaža u kojoj je auto Aston Martin (koordinate: x=1, y=5). U kuću utrčava Džems Bond, 8 sekundi ispred grupe loših momaka koji ga jure, a svi se kreću brzinom od 1 *m/s*. Dž.B. poseduje senzor koji mu javlja udaljenost od auta a loši momci pretražuju kuću deleći se u manje grupe po potrebi, tako da svaku prostoriju zaposedaju za najkraće moguće vreme.

a) Koji metod pretrage koristi Dž.B., a koji loši momci?

b) Pokazati kako će se kretati Dž.B., a kako njegovi gonioci, i odrediti hoće li Dž.B. uspeti da stigne do auta pre loših momaka.



Slika 31

Rešenje

a) Tlocrt kuće predstavlja graf pretrage, a Dž.B. i loši momci svojim prolaskom kroz kuću otkrivaju dva različita stabla pretrage. Džems Bond, kao pojedinac, može pri pretrazi da vrši

izbor samo lokalno najboljeg naslednika tekućeg čvora stabla pretrage u čemu mu pomaže senzor za određivanje rastojanja do automobila. Prema tome, Dž. B. primenjuje metod *planinarenja*.

Loši momci ne poseduju senzor za određivanje udaljenosti od kola, što odgovara pretrazi bez korišćenja heurističke funkcije. S obzirom da nevaljalci zaposedaju svaku od prostorija za najkraće vreme, radi se o pretrazi koja pronalazi optimalne putanje do svih soba, to jest, svih čvorova u grafu pretrage. Ove uslove zadovoljava pretraga metodom *grananja i ograničavanja*. Uz dodatnu pretpostavku da neko od loših momaka uvek ostaje u svakoj od soba koje zaposednu, loši momci ne moraju ponovo da zaposedaju sobe do kojih su ranije došli po alternativnoj putanji. U pretrazi se, pod ovom pretpostavkom, može primeniti princip *dinamičkog programiranja*, da se iz daljeg razmatranja isključe sve putanje koje vode do određene sobe osim najkraće.

b) Dž.B. će uspešno umaći ako dođe do automobila a da se pri prolasku kroz kuću ne sretne ni u jednoj od soba kroz koje prolazi sa goniocima. Za rešenje problema potrebno je, stoga, odrediti redosled soba kroz koje Džems Bond prolazi i odgovarajuće vremenske trenutke kada se nalazi u svakoj od njih i uporediti ih sa vremenskim trenucima kada sobe bivaju 'okupirane' od strane gonilaca.

Sledeće pretpostavke uprošćavaju računanje pređenih puteva i vremena potrebnih za njihovo prelaženje:

- Kretanje Džems Bonda i njegovih gonilaca je isključivo u pravcu x ili y ose (nema dijagonalnog kretanja).

- Pri prelaženju iz sobe u sobu uvek se ide iz centra jedne sobe u centar druge.

Stablo pretrage za pretraživanje metodom planinarenja prikazano je na slici 32.

Scenario kretanja Džemsa Bonda, prema tome, je putanjom prikazanom na mapi na istoj slici. Uzimajući u obzir brzinu kretanja, kao i navedene pretpostavke, zaključuje se sledeće:

- u sobi *f* Dž. B. se nalazi od 0. do 2. sekunde,

- u sobi *i* Dž. B. se nalazi od 2. do 6. sekunde,

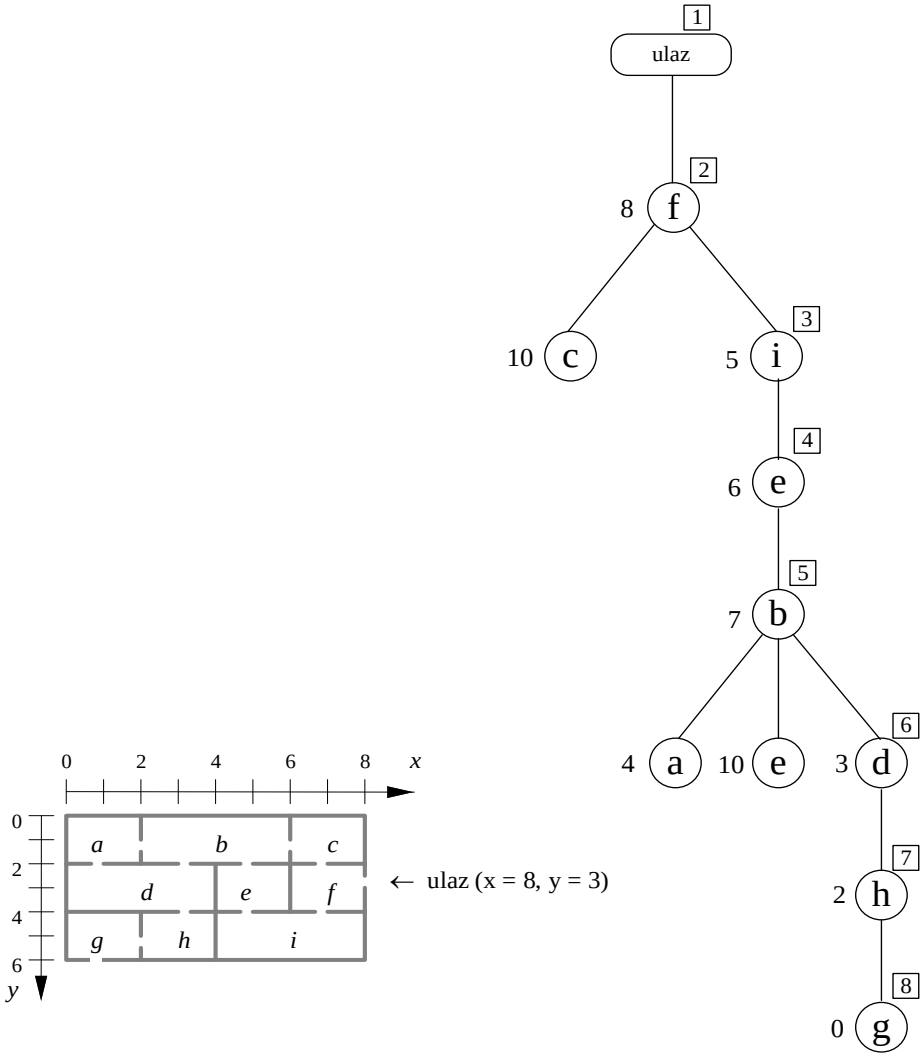
- u sobi *e* Dž. B. se nalazi od 6 do 8. sekunde,

- u sobi *b* Dž. B. se nalazi od 8 do 12. sekunde,

- u sobi *d* Dž. B. se nalazi od 12. do 16. sekunde (s obzirom na pretpostavku da se uvek dođe do centra sobe, dakle ne ide se pravolinijski od vrata do vrata),

- u sobi *h* Dž. B. se nalazi od 16. do 18. sekunde

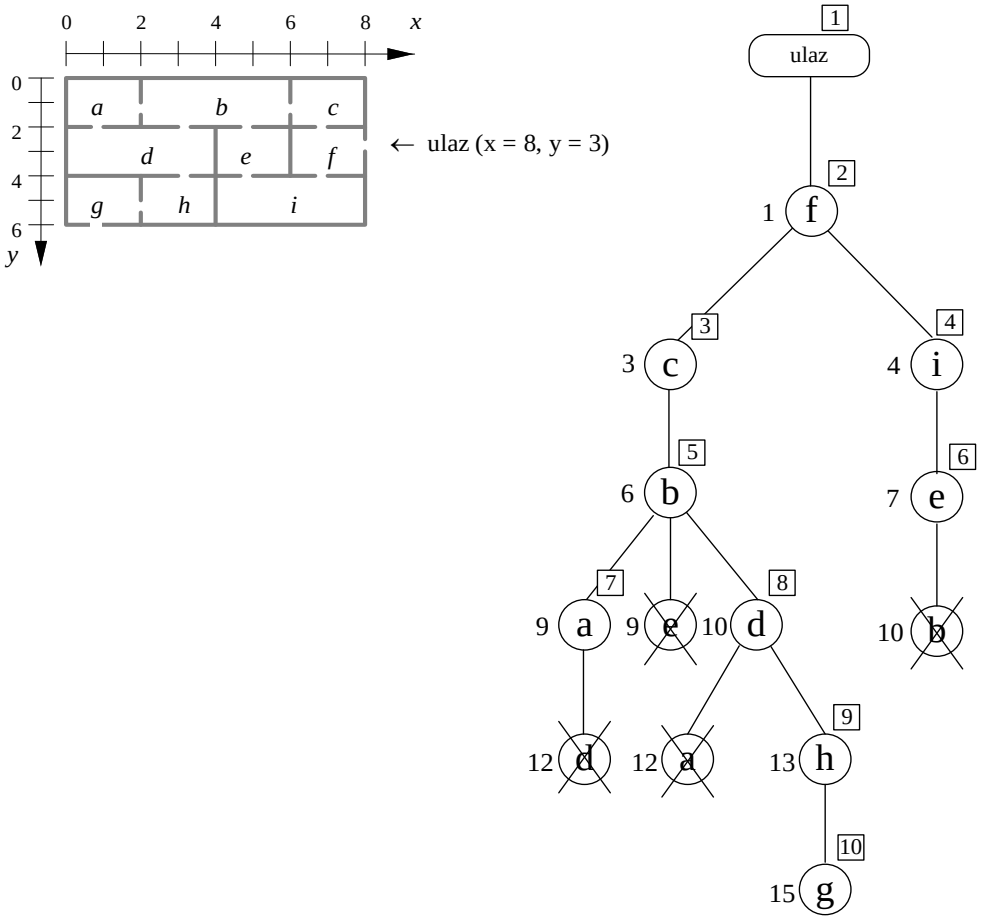
- u sobi *g* Dž. B. se nalazi od 18. do 19. sekunde, kada ulazi u kola i velikom brzinom izvozi ih iz kuće.



Slika 32

Gornja analiza važi naravno pod pretpostavkom da gonioci nisu presreli agenta 007 u kući. Kretanje loših momaka prikazano je stablom pretrage prikazanom na slici 33. Uz pojedine čvorove (levo od njih) naznačene su dužine putanja od ulaza do centara odgovarajućih prostorija, a uokvireni brojevi predstavljaju redosled obilaska čvorova. Precrtani čvorovi odgovaraju putanjama koje su eliminisane primenjujući princip dinamičkog programiranja, tako da se svi neprecrtani čvorovi nalaze na najkraćim putanjama od ulaza.

Na tlocrtu kuće na slici 33 prikazano je kretanje loših momaka po putanjama iz stabla pretrage. Dužina putanje do ulaza određene prostorije odgovara vremenu zaposedanja prostorije u sekundama od trenutka ulaska loših momaka u kuću (s obzirom da je brzina kretanja 1 m/s). Za prostorije *a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*, i *i* to su redom 8., 4., 2., 8., 6., 0., 12., 14. i 2. sekunda od trenutka ulaska, odnosno 16., 12., 10., 16., 14., 8., 20., 22. i 10. sekunda od trenutka ulaska Džemsa Bonda u kuću. Prema tome, Džems Bond uspešno izmiče lošim momcima koji su mu 'za petama' u prostorijama *b* i *d*.



Slika 33

Zadatak 14: Premeštanje terminala

U zgradi postoji nekoliko terminala koje treba premestiti:

- terminal A, sa trećeg sprata na drugi sprat,
- terminal B, sa drugog sprata na prvi sprat,
- terminal C i D, sa prvog sprata na treći sprat, i
- terminal E, sa prvog sprata na drugi sprat.

U polaznom stanju, lift (koji je u stanju da primi najviše dva terminala) nalazi se na prvom spratu. Cilj je da se svi terminali nađu na svojim odredištima, pri čemu se udaljenost do cilja meri zbirom spratova razlike za sve terminale. Cena vožnje liftom je

$$c = (0.8 + 0.2 * broj\_spratova) * (1 + 0.5 * broj\_terminala)$$

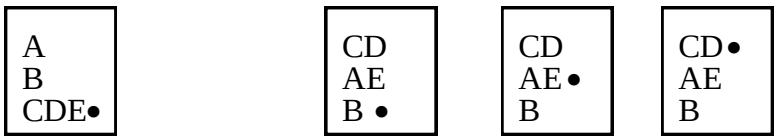
Pretpostavlja se da se lift po dolasku na određeni sprat svaki put prazni. Potrebno je odrediti redosled primene operatora i prikazati graf pretrage za metode pretraživanja:

a) 'prvo najbolji' i

b) A*.

Rešenje

Stanja ćemo predstaviti simbolički na način na koji je početno stanje prikazano na slici 34*a*, a tri moguća završna stanja na slici 34*b*. Tačkom je označen položaj lifta.



(a) (b)

Slika 34

Faktor grananja korena stabla pretrage je veliki (14 čvorova naslednika). U narednim nivoima stabla pretrage faktor grananja može se znatno smanjiti poštujući princip da se ne razmatra pomeranje terminala koji se već nalaze na svom mestu. Lako je zaključiti da rešenje koje bi obuhvatalo takva pomeranja ne može biti optimalno.

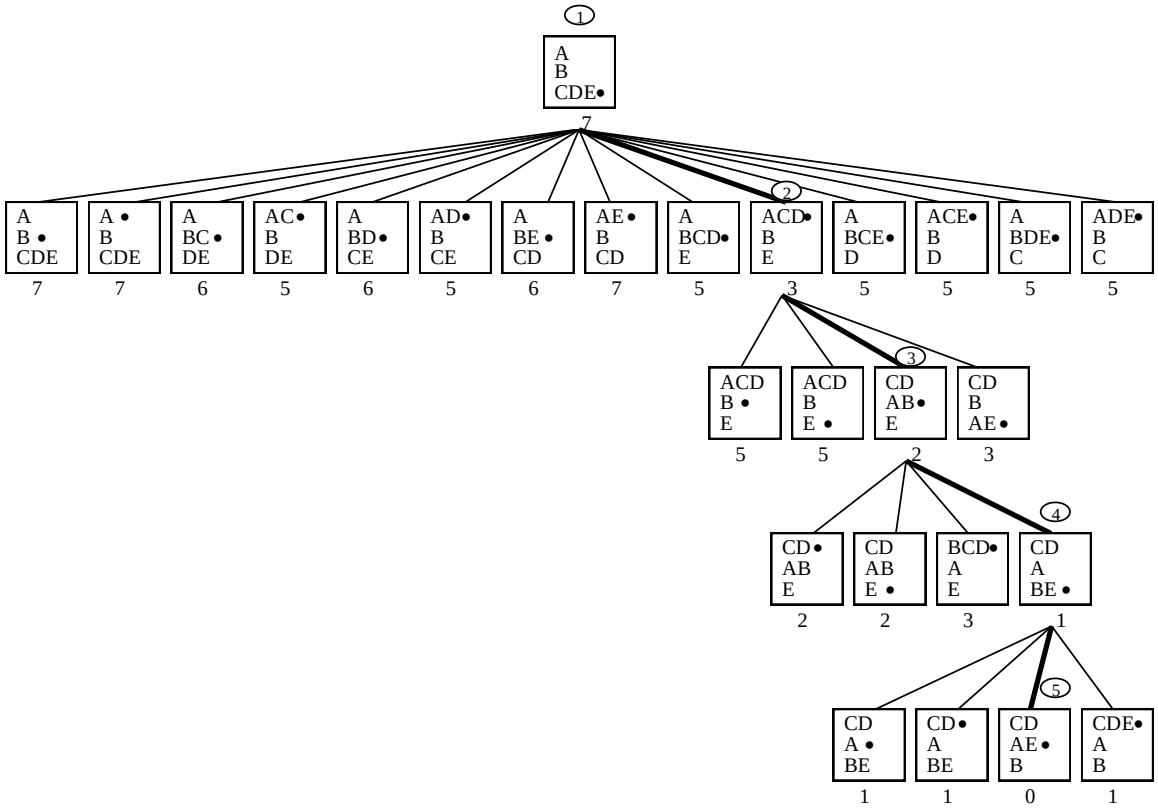
a) Stablo pretrage za metod 'prvo najbolji' prikazano je na slici 35. Ispod pojedinih čvorova navedene su vrednosti heurističke funkcije, a uokvireni brojevi daju redosled obilaženja čvorova. Rešenje dobijeno metodom 'prvo najbolji' se sastoji u sledećim akcijama:

- terminale C i D sa prvog sprata prebaciti na treći
- terminal A prebaciti sa trećeg sprata na drugi
- terminal B prebaciti sa drugog sprata na prvi i
- terminal E sa prvog sprata prebaciti na drugi sprat.

Može se pokazati (videti deo b) da je dobijeno optimalno rešenje, mada to nije garantovano u opštem slučaju za metod 'prvo najbolji'. Heuristička funkcija je dobro izabrana, pa pretraga sve vreme sledi ciljnu putanju.

b) Stablo pretrage i redosled pretraživanja čvorova za metod A* prikazani su na slici 36. Uz svaki čvor naveden je zbir heurističke funkcije (prvi sabirak) i kumulativne cene po parcijalnoj putanji od korena stabla do tog čvora (drugi sabirak). Pronađeno je isto (optimalno) rešenje kao i u slučaju a).

U slučajevima a) i b) prve četiri ekspanzije stanja pri pretrazi slede istu putanju koja je ujedno i ciljna putanja. Dok se u slučaju pod a) za petu ekspanziju bira ciljni čvor i time pretraga okončava, u slučaju b) pretraga skreće sa ciljne putanje i vrše se dve ekspanzije više pre nego što se obiđe ciljni čvor. Ovakvo ponašanje A* algoritma je posledica činjenice da ovaj algoritam (pod određenim uslovima) garantuje optimalnost rešenja. U trenutku unošenja ciljnog čvora u stablo pretrage, računa se funkcija procene f_c ovog čvora. S obzirom da se radi o ciljnom čvoru, heuristička funkcija za ovaj čvor je nula, pa je f_c jednako ceni putanje od korena do ciljnog čvora, to jest ceni rešenja. Posle unosa ciljnog čvora u stablo pretrage, a pre njegovog obilaska, to jest, završetka pretrage, kod A* algoritma potrebno je ispitati sve parcijalne putanje za koje je funkcija procene f manja od funkcije f_c ciljne putanje. Ove parcijalne putanje se produžuju



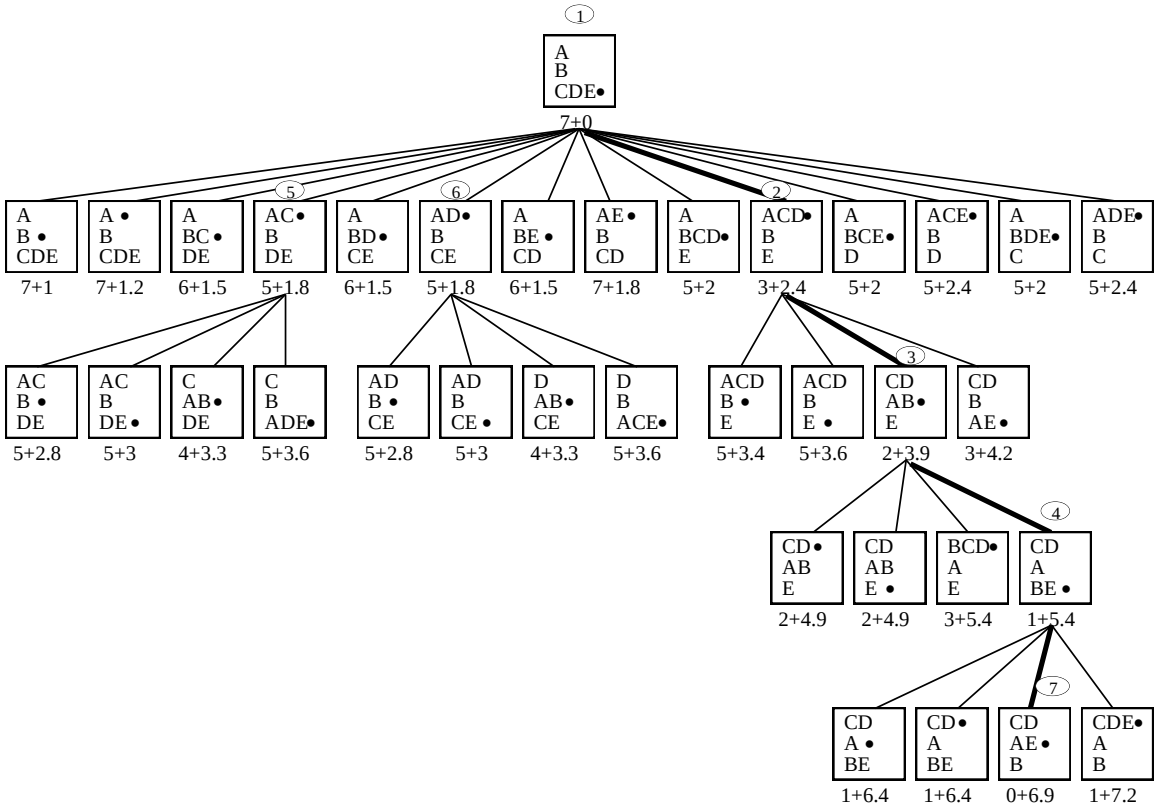
Slika 35

- do svoga kraja ili
- do tačke u kojoj je funkcija f veća od f_c,

jer postoji mogućnost da se nađe nova, kraća putanja do cilja. U konkretnom slučaju, po otkrivanju ciljnog čvora posle četvrte ekspanzije ekspandovana su još dva čvora van ciljne putanje pre nego što je pretraga okončana jer su njihove funkcije procene bile manje od 6.9 koliko iznosi cena rešenja.

Postavlja se pitanje da li završetak pretrage pod navedenim uslovima obezbeđuje uvek dobijanje optimalnog rešenja. Funkcija f za neku parcijalnu putanju može se shvatiti kao procena cene rešenja koje bi se dobilo produžavanjem te putanje, s obzirom da c predstavlja već plaćenu cenu, a h procenu potrebnih troškova da se iz datog čvora stigne do cilja. Nađeno rešenje je garantovano optimalno jedino ako je h tačna procena ili potcenjena vrednost, tako da je f takođe ili tačna ili potcenjena vrednost. Tada možemo sa sigurnošću zanemariti putanje sa vrednošću f većom od f_c, jer bi eventualno rešenje po toj putanji imalo veću cenu od cene nađenog rešenja, to jest, vrednosti f_c (jer je u ciljnoj tački h=0). U suprotnom slučaju, kada je f precenjena vrednost, odbacivanjem putanja kod kojih je $f > f_c$ može se desiti da zanemarimo putanju koja vodi do boljeg rešenja.

U konkretnom slučaju nađena je optimalna putanja. Međutim, funkcija h u nekim tačkama precenjuje vrednost do cilja. Na primer, u korenom čvoru h=7 dok je cena rešenja jednaka 6.9. Pored nađenog, postoji još jedno optimalno rešenje sa istom cenom 6.9:



Slika 36

- premestiti terminal E sa prvog sprata na drugi (cena 1.5)
- premestiti terminal B sa drugog sprata na prvi sprat (cena 1.5)
- premestiti terminal C i D sa prvog sprata na treći (cena 2.4) i
- premestiti terminal A sa trećeg sprata na drugi (cena 1.5)

Ciljna putanja za ovo alternativno rešenje obuhvata sedmog naslednika korenog čvora u stablu pretrage sa slike 36. Za pomenutog naslednika vrednost funkcije procene f iznosi 7.5, pa je u toj tački cena ovog rešenja precenjena za čitavih 0.6. Da je, kojim slučajem, alternativno rešenje bilo jedino optimalno rešenje, moglo bi se desiti da se pri pretrazi prvo pronađe rešenje sa cenom npr. 7.4 i ovo rešenje proglasi za najbolje, a da se putanja optimalnog rešenja ne pokušava produžiti jer ima veću vrednost funkcije procene (7.5).

Drugim rečima, za neki drugi raspored terminala upotrebljena heuristička funkcija ne garantuje optimalnost dobijenog rešenja.

Zadatak 15: Igra pomeranja blokova

Posmatrajmo igru pomeranja blokova. Neka je početna konfiguracija:

C	C	B	B	P

pri čemu je C - oznaka za crni blok, B - oznaka za beli blok, a P - oznaka za praznu ćeliju.

Pravila igre su sledeća:

- Blok se može pomeriti u susednu praznu ćeliju po jediničnoj ceni.
- Blok može da preskoči jedan blok da bi bio premešten u praznu ćeliju, po ceni 2.

Cilj igre je da se svi beli blokovi smeste levo od svih crnih blokova (bez obzira na poziciju prazne ćelije).

Naći heurističku funkciju *h* i dati rešenje problema primenom algoritma A*. Da li izabrana heuristička funkcija h zadovoljava monotonu restrikciju $h(n_i) \leq h(n_j) + c(n_i, n_j)$, za svaki par pozicija n_i i n_j gde n_j predstavlja sledbenika n_i? U prethodnoj formuli $c(n_i, n_j)$ predstavlja cenu poteza koji iz n_i vodi u n_j.

Rešenje

Definišimo heurističku funkciju *h* na sledeći način:

- za svaki od belih blokova odredimo broj crnih blokova cb_i koji se nalaze levo od njega u tekućem stanju.
- heuristička funkcija *h* je zbir brojeva određenih u prethodnoj tački pomnožen sa 2:

$$h = 2\sum_{i=1}^{2} cb_i$$

Na primer, u početnom stanju za oba bela bloka važi da se levo od njih nalaze po dva crna bloka pa je $h = 2 * (2+2) = 8$.

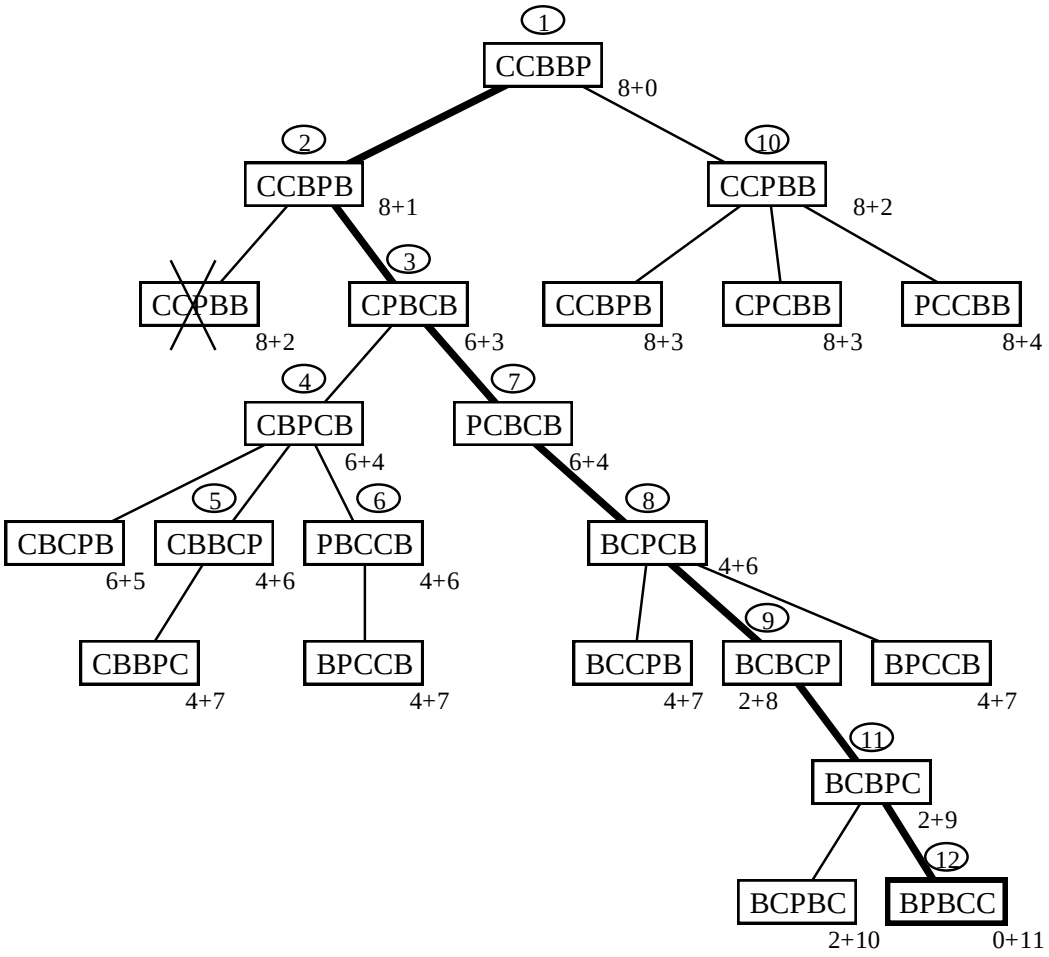
Koristeći funkciju *h* i inkrementalne cene operatora date u postavci, moguće je primeniti algoritam A*. Na slici 37 dato je stablo pretrage pri rešavanju ovoga problema. Zaokruženi brojevi označavaju redosled ekspanzije čvorova. Funkcija procene *f* navedena je u obliku zbira heurističke funkcije *h* (prvi sabirak) i cene parcijalne putanje *c* (drugi sabirak). U slučaju jednakih funkcija procene za dva čvora, pri pretrazi se najpre ekspanduje onaj sa manjom vrednošću heurističke funkcije *h.*

Zadovoljenost uslova monotone restrikcije za proizvoljnu heurističku funkciju *h* obezbeđuje da ova funkcija uvek potcenjuje udaljenost do cilja, pa samim tim garantuje optimalnost dobijenog rešenja. Heuristička funkcija *h* koju smo definisali za ovaj problem zadovoljava monotonu restrikciju u šta nas uverava sledeće rezonovanje:

Neka n_i označava proizvoljnu poziciju do koje je stigla pretraga, a H vrednost heurističke funkcije $h(n_i)$ za ovu poziciju. Mogući potezi koji dovode u novu poziciju n_j mogu se podeliti u sledeće klase:

1. Pomeranje jednog (belog ili crnog) bloka u susednu praznu ćeliju ili prebacivanje jednog bloka preko bloka iste boje. Ovi potezi ne menjaju međusobni položaj belih i crnih blokova pa je vrednost heurističke funkcije za novu poziciju $h(n_j)$ jednaka vrednosti H. S obzirom da je cena poteza $c(n_i, n_j)$ veća od nule (iznosi 1 ili 2), nejednakost $h(n_i) \leq h(n_j) + c(n_i, n_j)$ u ovom slučaju je ispunjena:

$$H \leq H + c(n_i, n_j)$$



Slika 37

2. Prebacivanje crnog bloka na desnu stranu preko belog bloka ili prebacivanje belog bloka na levu stranu preko crnog bloka. Ovi potezi, za jedan od belih blokova, smanjuju broj crnih blokova na levoj strani za 1. Prema tome, za novo stanje će biti:

   $$h(n_j) = H - 2$$

   Međutim, cena ovih poteza je 2 pa je nejednakost ispunjena:

   $$H \leq (H - 2) + 2$$

3. Prebacivanje crnog bloka na levu stranu preko belog bloka ili prebacivanje belog bloka na desnu stranu preko crnog bloka. Ovi potezi, za jedan od belih blokova, povećavaju broj crnih blokova na levoj strani za 1. Prema tome, za novo stanje će biti:

   $$h(n_j) = H + 2$$

   Cena ovih poteza je 2. Može se utvrditi da je i u ovom slučaju nejednakost ispunjena:

   $$H \leq (H + 2) + 2$$

Problem se može generalizovati, na primer, na sledeći način:

- u početnom stanju svih *n* crnih blokova smešteno je levo od svih *n* crnih blokova, a prazna ćelija je u krajnje desnoj poziciji.

- u svakom potezu jedan blok se može pomeriti u praznu ćeliju, pod uslovom da pri tome ne preskoči više od *k* drugih blokova. Cena poteza je broj preskočenih blokova +1.
- cilj igre ostaje nepromenjen: premestiti sve bele blokove levo od svih crnih blokova.

Funkcija *h*, definisana u rešenju zadatka, ne zadovoljava monotonu restrikciju u generalizovanoj igri; na primer, ako beli blok preskoči dva crna, funkcija *h* se smanjuje za 4, a operator promene stanja košta samo 3. Kako izgleda heuristička funkcija koja zadovoljava monotonu restrikciju u ovom slučaju?

Zadatak 16: Viktorija (dvosmerna pretraga)

Na slici 38 zadate su početna i krajnja pozicija u igri Viktorija na polju 3 x 3. Pravila ove igre definisana su u zadatku 9.

2	8	3
1	6	4
7		5

početna pozicija

1	2	3
8		4
7	6	5

krajnja pozicija

Slika 38

a) Rešiti problem pretraživanjem unazad uz korišćenje algoritma A*. Za heurističku funkciju uzeti broj pločica koje nisu na svom mestu u tekućoj poziciji u odnosu na ciljnu poziciju. Inkrementalna cena svakog poteza je 1.

b) Za dati problem utvrditi gde se pretraživanje unapred i unazad susreću.

Rešenje

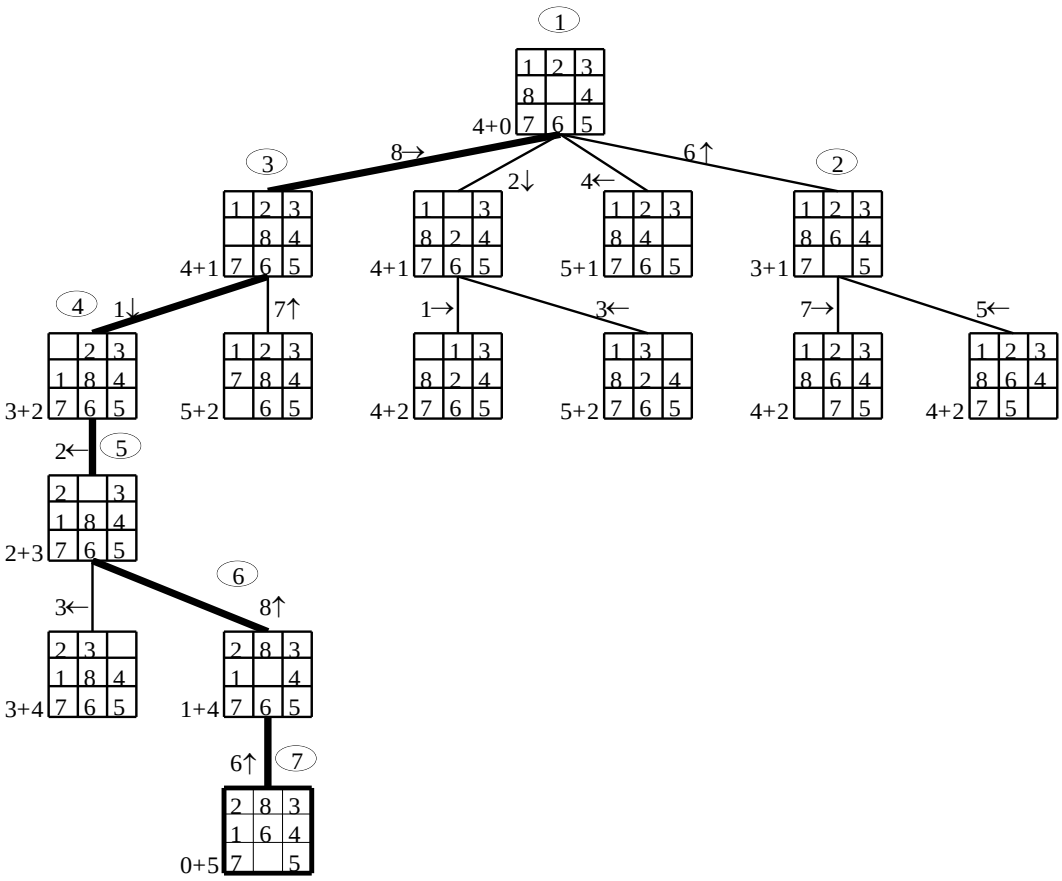
Problem u tački a) treba rešiti pretraživanjem unazad pa se za startno stanje pretrage uzima krajnja pozicija. Cilj pretrage je zadovoljen kada se dođe do početne pozicije koja u ovom slučaju predstavlja ciljno stanje pretrage. Operatori promene stanja primenljivi su u oba smera pretraživanja (i pri pretrazi unapred i pri pretrazi unazad).

Heuristička funkcija za startno stanje iznosi 4 jer pločice sa brojevima 1, 2, 6 i 8 nisu na svom mestu u odnosu na ciljno stanje. Mada je zadata heuristička funkcija jednostavna za računanje po definiciji, za nova stanja u pretrazi heurističku funkciju je moguće izračunati još lakše na osnovu heurističke funkcije stanja prethodnika u stablu pretrage. U cilju dobijanja vrednosti heurističke funkcije za novo stanje S' koje se dobija iz stanja-prethodnika S pomeranjem pločice sa brojem N na prazno mesto, heurističkoj funkciji stanja S treba oduzeti 1, dodati 1 ili je ostaviti nepromenjenu u zavisnosti od toga da li smo pomerili pločicu N u njenu ciljnu poziciju, udaljili pločicu sa ciljne pozicije, ili pomerena pločica nije u stanju S bila u ciljnoj poziciji niti se u novom stanju S' nalazi u ciljnoj poziciji.

U nekim slučajevima pretraživanja pogodno je sprovesti kombinovani metod pretraživanja po ideji da se istovremeno vrši i pretraga unapred (od početnog stanja ka završnom) i pretraga unazad (od završnog stanja ka početnom) dok se ne otkrije zajedničko stanje u stablima pretrage. U trenutku kada se dve pretrage susretnu, moguće je rekonstruisati kompletnu putanju rešenja od početnog do ciljnog čvora. U nemogućnosti da se pretrage unapred i

unazad vrše potpuno paralelno (kao na višeprocesorskoj mašini) ova ideja može se sprovesti tako da se naizmenično ekspanduje po jedno stanje u obe pretrage.

a) Stablo pretrage unazad uz korišćenje metoda A* prikazano je na slici 39. Rešenje predstavlja redosled poteza na putanji od ciljnog stanja ka korenu stabla pretrage.



Slika 39

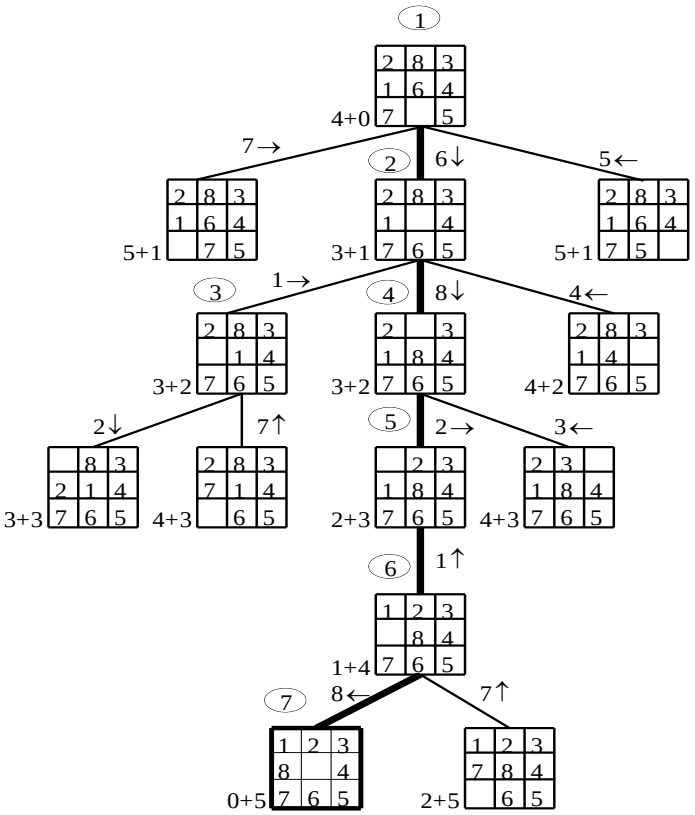
b) Kompletno stablo pretrage unapred za zadati problem, koristeći metod A* na osnovu zadate heurističke funkcije, prikazano je na slici 40.

Dobijeno je isto rešenje kao pri pretrazi unatrag. U opštem slučaju to ne mora da bude ispunjeno ako postoji više optimalnih rešenja (sa minimalnim brojem poteza).

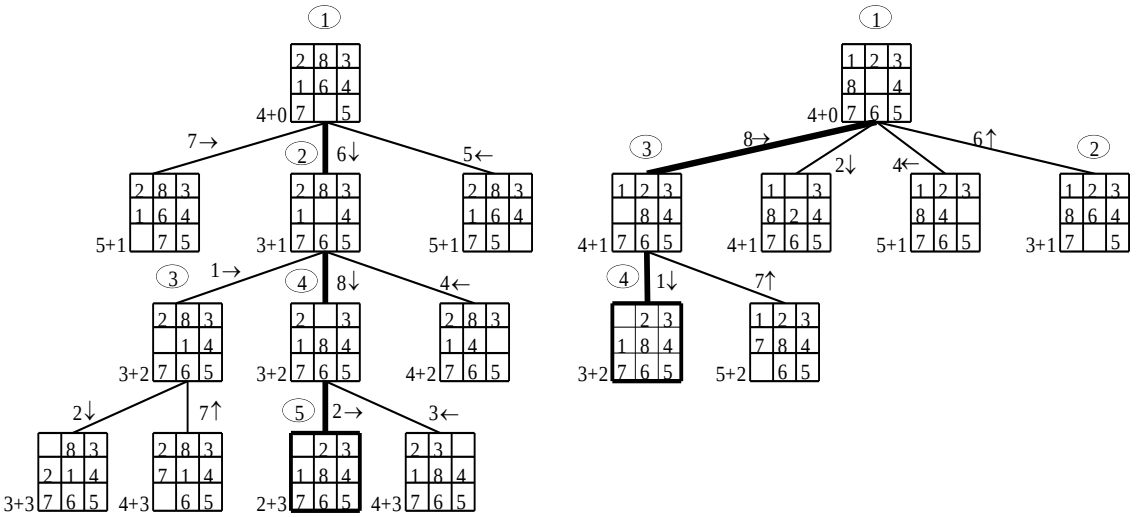
Pod pretpostavkom da kombinovanu pretragu sprovodimo naizmenično ekspandujući stanja prvo u pretrazi unapred, pa potom i unazad, pretrage će se susresti posle obilaska pet stanja u pretrazi unapred (u pretrazi unatrag to tog trenutka obiđena su četiri stanja). Na slici 41 prikazana su stabla pretrage unapred i unatrag u trenutku susreta. Debljom linijom označeno je zajedničko stanje u stablima pretrage. Rešenje problema predstavlja niz poteza na putanji u stablu pretrage unapred od korena stabla do zajedničkog stanja na koji se nadovezuje niz poteza na putanji u stablu pretrage unazad od zajedničkog stanja do korena. Dobijeno rešenje isto je kao u prethodnim slučajevima.

U konkretnom slučaju kombinovana pretraga iz tačke b) nije omogućila uštedu u pretraživanju. Pri pretrazi unapred ukupno je obiđeno 7 čvorova, pri pretrazi unazad takođe 7, dok je pri kombinovanoj pretrazi ekspandovano ukupno 9 čvorova. Uštede nije bilo jer je heuristička funkcija dobro usmeravala pretragu. Veće uštede pri primeni kombinovane

pretrage mogu se očekivati kod metoda gde pretraga u većoj meri odstupa od ciljne putanje, na primer pri pretrazi po širini kada je ciljna putanja relativno dugačka.



Slika 40

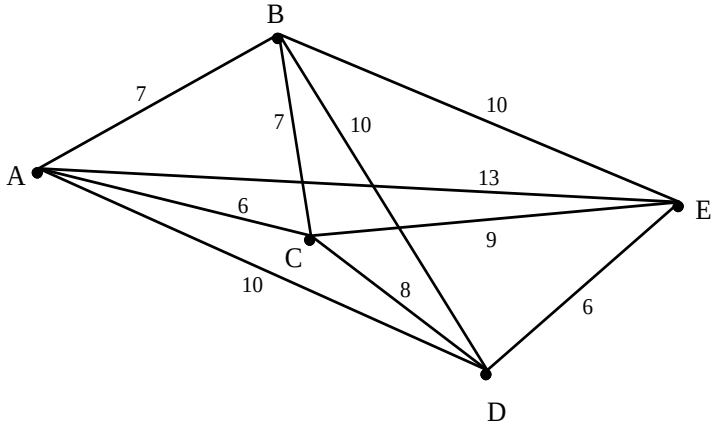


Stablo pretrage unapred Stablo pretrage unazad

Slika 41

Zadatak 17: Problem trgovačkog putnika

Trgovački putnik mora da poseti svaki od pet gradova prikazanih na slici 42. Između svakog para gradova postoji put, dužine naznačene na slici. Polazeći od grada A, naći minimalan put koji obezbeđuje posetu svakom gradu samo jedanput i povratak u A. Predložiti dve različite heurističke funkcije. Za svaku od funkcija primenom nekog od algoritama pretraživanja naći rešenje problema. Koja od predloženih funkcija daje bolje rešenje?



Slika 42

Rešenje

Najjednostavnija je heuristika da u svakom koraku pretraživanja zadatog grafa prioritet damo jednom od neobiđenih gradova koji je najbliži tekućem gradu (i do koga postoji put iz tekućeg grada). Funkcija koja odgovara ovoj heuristici definiše se na sledeći način:

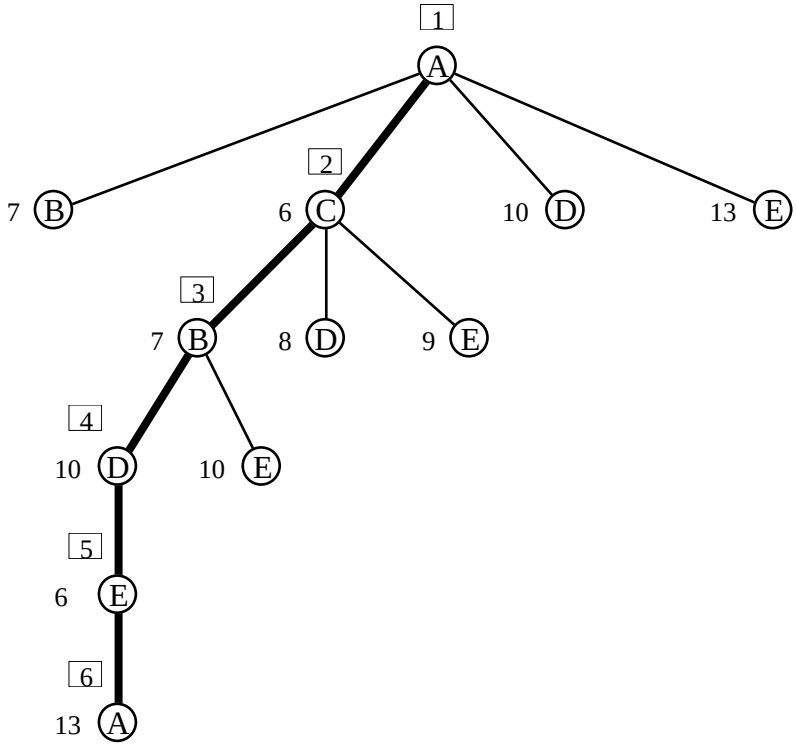
Vrednost heurističke funkcije h_1 čvora Y - naslednika tekućeg čvora X - jednaka je dužini puta između gradova X i Y.

$$h_1(Y) = rastojanje(X,Y)$$

Pošto nam heuristička funkcija služi da ocenimo čvor lokalno među sledbenicima tekućeg čvora, prirodno joj odgovara algoritam planinarenja. Stablo pretrage za dati problem koje se dobija primenom planinarenja prikazano je na slici 43.

Put A-C-B-D-E-A koji se dobija kao rešenje je dužine 42 i nije minimalan, jer planinarenje i ne garantuje nalaženje minimalnog rešenja. Minimalan put za dati problem je A-C-D-E-B-A (ili ista ruta u suprotnom smeru) koji ima dužinu 37.

Da bismo dobili garantovano minimalno rešenje, moramo ga tražiti primenom algoritma A*. Heuristička funkcija h_1 nije pogodna u ovom slučaju jer je vrednost funkcije h_1 za proizvoljan čvor stabla pretrage već uračunata u cenu parcijalne putanje do tog čvora, tako da bi se algoritam A* sveo na algoritam 'granaj i ograniči' što bi imalo za posledicu otkrivanje gotovo kompletnog stabla pretrage za dati problem. Kompletno stablo pretrage prikazano je na slici 44.

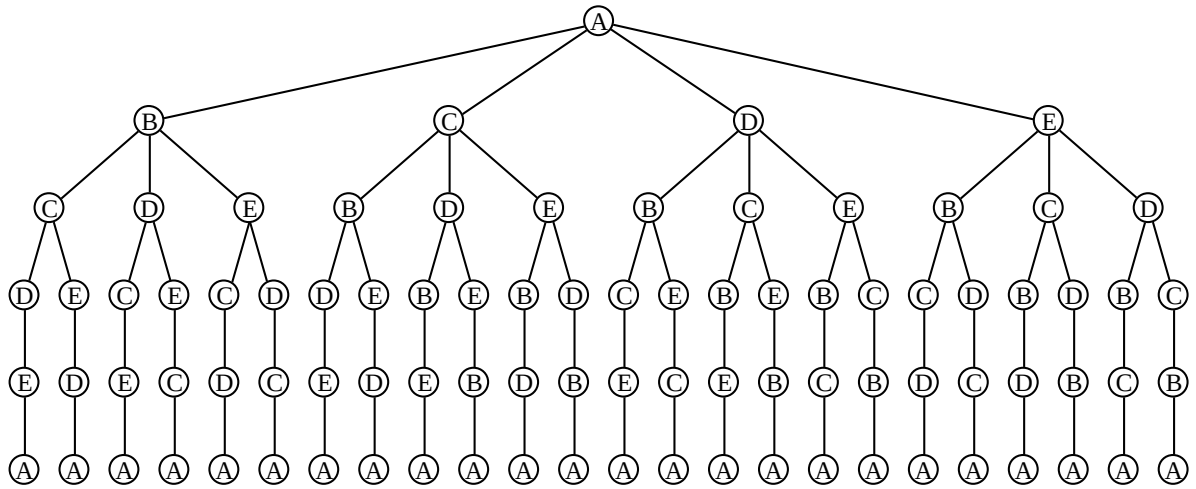


Slika 43

Razmotrimo heurističku funkciju h_2:

Vrednost heurističke funkcije h_2 čvora Y na parcijalnoj putanji P od korena stabla pretrage jednaka je dužini takozvanog *minimalnog razapinjućeg stabla* (MRS) koje obuhvata sve čvorove grafa G koji se ne nalaze na putanji P osim čvorova Y i A:

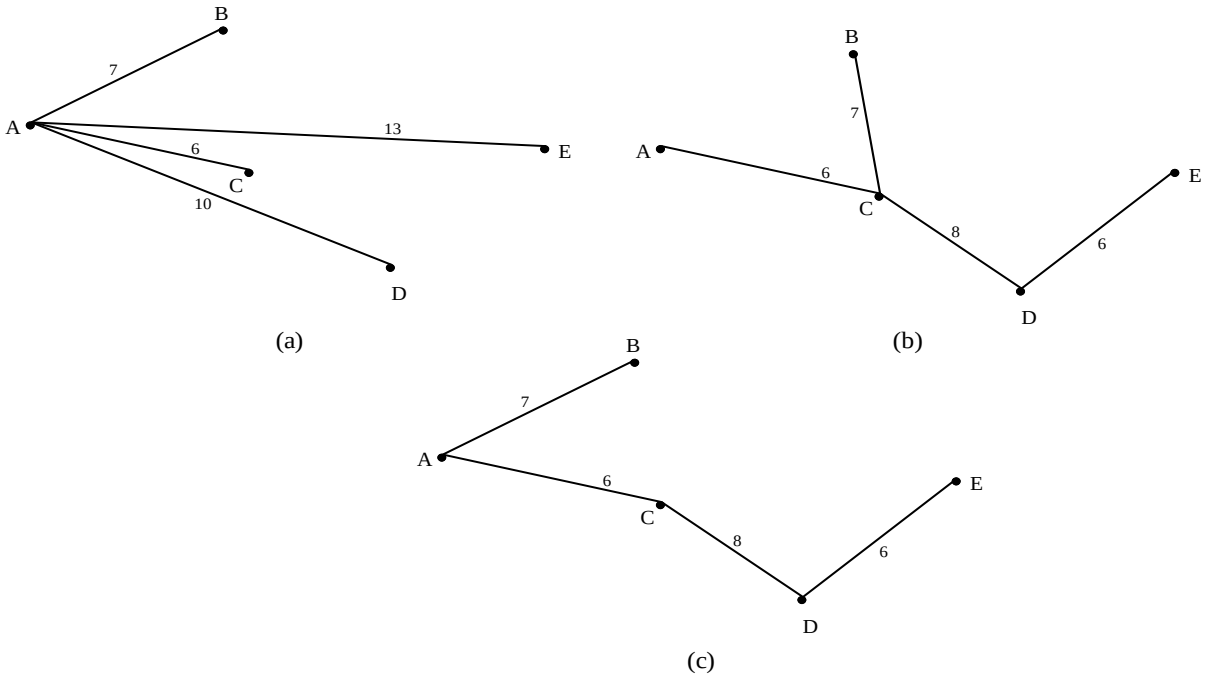
$h_2(Y)$ = dužina MRS((G\P)∪{A,Y})



Slika 44

Razapinjuće stablo u nekom grafu je skup grana grafa koje povezuju sve čvorove grafa tako da nije formirana nijedna zatvorena petlja. Dužina razapinjućeg stabla je zbir dužina svih grana koje ga sačinjavaju. Minimalno razapinjuće stablo u grafu ima najmanju dužinu od svih razapinjućih stabala tog grafa.

Slika 45 prikazuje tri različita razapinjuća stabla koja odgovaraju kompletnom grafu pretrage sa slike 42. Razapinjuća stabla sa slike 45*b* i 45*c* ujedno predstavljaju dva minimalna razapinjuća stabla za ovaj graf.



Slika 45

Dužina minimalnog razapinjućeg stabla za dati graf može poslužiti kao procena dužine puta koji mora preći trgovački putnik. Pokazuje se da je u pitanju uvek potcenjena vrednost. U našem slučaju dužina minimalnog razapinjućeg stabla je 27, dok je dužina minimalnog puta za trgovačkog putnika jednaka 37. Funkcija h_2 koristi dužinu minimalnog razapinjućeg stabla neobiđenog dela grafa pretrage za procenu preostalog dela puta. S obzirom da funkcija h_2 uvek daje potcenjenu vrednost, postoji garancija da će algoritam A* uvek dati optimalno rešenje.

Za zadati graf G, minimalno razapinjuće stablo H određuje se na osnovu sledećeg algoritma:

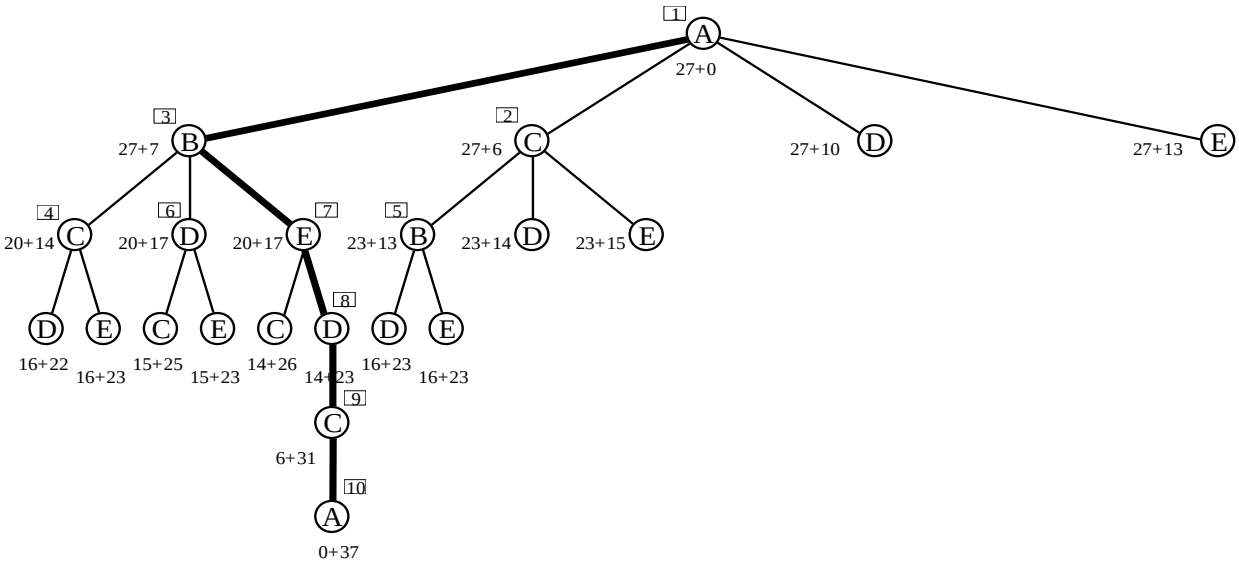
1. Inicijalno stablo H sadrži sve čvorove grafa G i nijednu granu.

2. U stablo H uključujemo najkraću granu grafa G koja nije u H, a koja sa granama iz H ne zatvara konturu.

3. Ponavljamo korak 2. dok se ne formira jedinstveno stablo H koje povezuje sve čvorove.

Za graf na slici 42, primenjujući ovaj algoritam biramo redom grane AC, DE, BC (posle toga ne može AB, jer bi se formirala kontura AB, BC, AC) i CD i tako dobijamo razapinjuće stablo sa slike 45*b*. Ukoliko posle grana AC i DE izaberemo granu AB, tada ne može BC već se bira AB tako da se dobija stablo sa slike 45*c*.

Stablo pretrage korišćenjem algoritma A* i heurističke funkcije h_2 prikazano je na slici 46. U pretrazi, od dva čvora sa jednakim funkcijama procene, prioritet je dat čvoru sa manjom vrednošću heurističke funkcije.

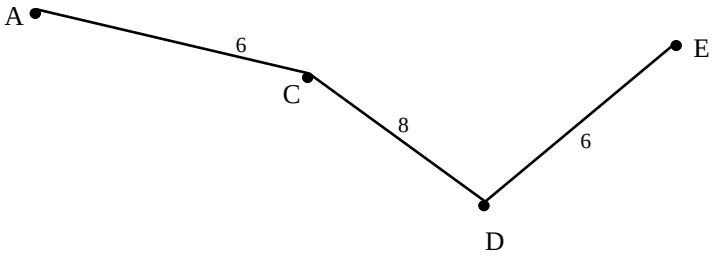


Slika 46

Vrednost funkcije procene f prikazana je uz svaki čvor stabla pretrage na uobičajen način kao zbir vrednosti heurističke funkcije (levi sabirak) i funkcije cene parcijalne putanje (u ovom slučaju dužina puta) od korena stabla pretrage do datog čvora (desni sabirak). Kao primer određivanja funkcije procene tokom pretrage razmotrimo vrednost funkcije f u čvoru E, na parcijalnoj putanji A-B-E:

- h_2(E)= dužina MRS({A,C,D,E})

  Primetimo da se dužina minimalnog razapinjućeg stabla određuje za deo polaznog grafa iz koga je udaljen čvor B (i grane koje ga povezuju sa drugim čvorovima) jer se ovaj čvor već nalazi na parcijalnoj putanji, dok su u grafu ostali čvorovi A i E s obzirom da moramo proceniti preostali put od čvora E do čvora A preko čvorova C i D. Minimalno razapinjuće stablo za traženi podgraf je prikazano na slici 47 i njegova dužina iznosi 20, koliko je na slici 47 i naznačeno.



Slika 47

- Dužina parcijalne putanje c(A-B_E) = rastojanje(A,B) + rastojanje(B,E) = 7 + 10 = 17 tako da je ukupna vrednost funkcije f za čvor E jednaka 37.

Pri rešavanju problema trgovačkog putnika algoritmom A* ne sme se koristiti princip dinamičkog programiranja koji nalaže da se od svih parcijalnih putanja do određenog grada razmatra samo najkraća od njih. Da je ovaj princip korišćen, u stablu pretrage sa slike 46 eliminisao bi se, na primer, čvor E na parcijalnoj putanji A-C-D-E jer je dužina te parcijalne putanje jednaka 14 i veća od dužine parcijalne putanje A-D koja je jednaka 10. Parcijalna putanja A-C-D se nalazi na ciljnoj putanji, tako da bi se eliminacijom te putanje eliminisalo i

minimalno rešenje. Kod problema trgovačkog putnika moraju se, prema tome, ravnopravno razmatrati sve parcijalne putanje do određenog čvora.

U literaturi se mogu naći i druge heuristike za rešavanje problema trgovačkog putnika koje nisu zasnovane na algoritmima pretraživanja i koje u praksi daju dobre rezultate.

Zadatak 18: Problem zamenjivanja brojeva

Data su sledeća pravila koja se mogu iskoristiti da zamene brojevi na levoj strani nizom brojeva na desnoj strani:

$6 \rightarrow 3,3$	$4 \rightarrow 2,2$	$3 \rightarrow 2,1$
$6 \rightarrow 4,2$	$4 \rightarrow 3,1$	$2 \rightarrow 1,1$

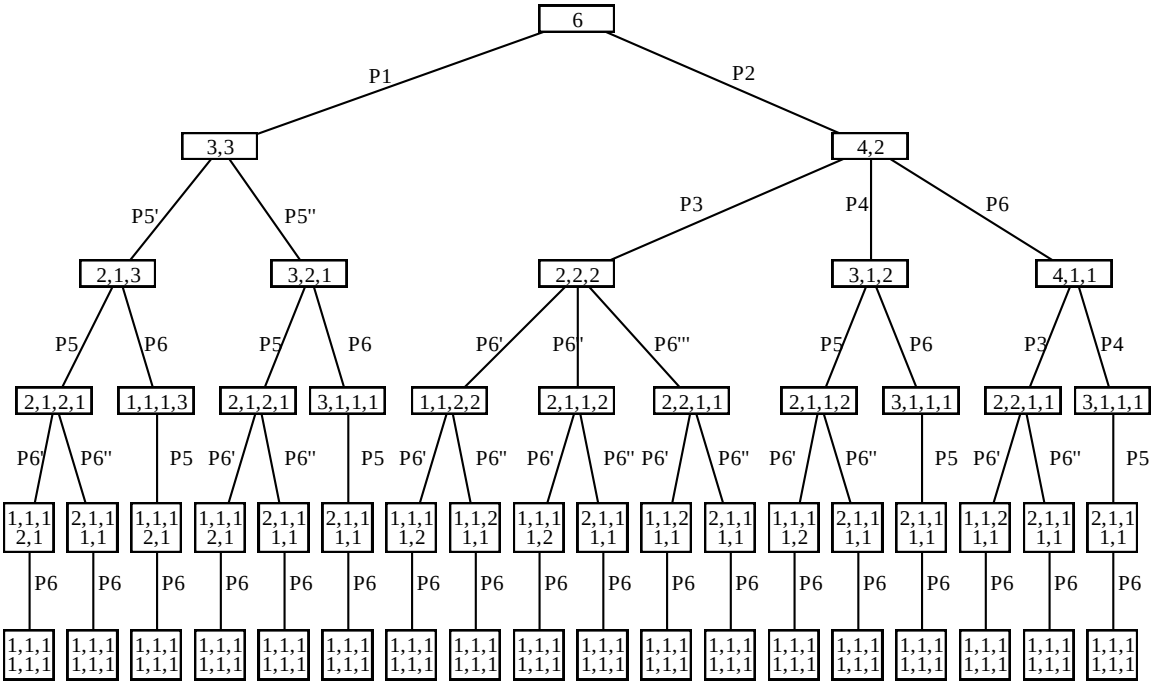
Kako se mogu iskoristiti ova pravila da se broj 6 transformiše u niz jedinica? Pokazati kako AO* algoritam obavlja ovu transformaciju. Usvojiti da je cena k-konektora k jedinica, a vrednost heurističke funkcije h u čvoru označenom brojem 1 je nula a čvora označenog sa n iznosi n.

Rešenje

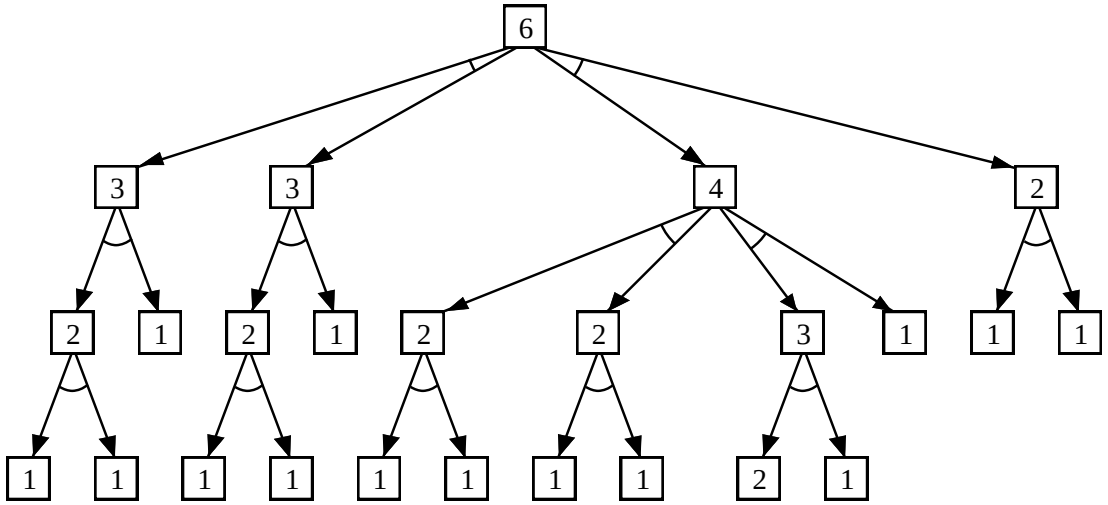
Rešenje postavljenog problema mogli bismo potražiti klasičnim algoritmima pretrage. Kompletno stablo pretrage u tom slučaju prikazano je na slici 48. Ovo stablo sadrži dosta redundanse. Na primer, za rešenje problema nije bitno da li u stanju koje je opisano listom 2,1,3 prvo zamenjujemo cifru 2 primenom pravila P6 pa onda cifru 3 primenom pravila P5 ili obrnuto, jer je u oba slučaja krajnji rezultat isti.

Zadati problem može se *razložiti* (dekomponovati) na niz potproblema tako da svaki problem rešavamo nezavisno. Način na koji zamenjujemo pojedinu cifru jedinicama možemo rešavati nezavisno od zamene ostalih cifara u istom stanju. Na primer, da bismo rešili problem prelaska iz stanja 4,2 u ciljno stanje 1,1,1,1,1,1 posebno ćemo posmatrati problem zamene cifre 4 jedinicama i problem zamene cifre 2 jedinicama.

Pogodna predstava razloženog problema zamene cifara jedinicama je uz upotrebu AND-OR stabla. Za zadati problem AND-OR stablo predstavljeno je na slici 49. Čvorovi stabla su pojedinačni potproblemi, u ovom slučaju pojedinačne cifre. Čvorovi su povezani takozvanim *k-konektorima*. Radi se o generalizovanim granama koje povezuju jedan čvor-roditelj sa k čvorova naslednika.



Slika 48



Slika 49

Konektor je predstavljen nizom grana koje čvor roditelj povezuju sa svakim od naslednika i koje su sve međusobno povezane lukom. Na primer, iz korenog čvora 6 polaze dva 2-konektora:

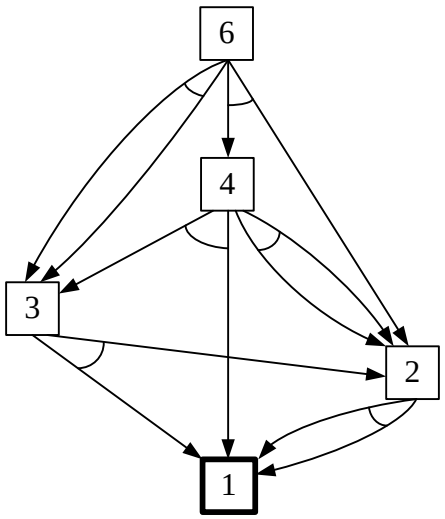
- Levi 2-konektor povezuje čvor 6 sa dva čvora, oba označena cifrom 3. Ovaj konektor izražava pravilo P1 da se cifra 6 može zameniti sa dve cifre 3.
- Desni 2-konektor povezuje čvor 6 sa čvorovima 4 i 2. Ovaj konektor izražava pravilo P2 da se cifra 6 može zameniti ciframa 4 i 2.

Prema tome, problem zamene cifre 6 jedinicama može se rešiti ILI zamenom 6 sa 3 I 3 ILI zamenom 6 sa 4 I 2. U opštem slučaju, potproblem je rešen ako je rešen primenom bilo kog od konektora iz odgovarajućeg čvora grafa, pri čemu po tom konektoru svi čvorovi naslednici

moraju biti rešeni. Konektor dakle, izražava I relaciju (engl. AND) dok postojanje više konektora iz istog čvora izražava ILI relaciju (engl. OR) pa su po tome AND-OR stabla i dobila ime.

Jedna terminološka napomena: U nekim problemima pretrage AND-OR stabla imaju osobinu da iz svakog čvora grafa ide ili jedan k-konektor (k>1) ili k 1-konektora. Čvorovi za koje važi prvo svojstvo nazivaju se tada AND čvorovi, a čvorovi sa drugim svojstvom OR čvorovi. Primer AND čvora u stablu sa slike 49 bio bi čvor 3. U istom stablu, međutim, koreni čvor 6 nije ni AND čvor ni OR čvor.

U zadatom problemu može se primetiti da će isto rešenje za zamenu određene cifre biti primenljivo nezavisno od stanja u kome se nalazi ta cifra. Na primer, rešenje koje dobijemo za cifru 4 biće primenljivo i u stanju 4,1,1 kao i u stanju 4,2. Koristeći ovo svojstvo možemo rešenje problema predstaviti još kompaktnije koristeći, umesto AND-OR stabla, AND-OR aciklički graf. AND-OR aciklički graf (u nastavku ćemo ga skraćeno nazivati AND-OR grafom) je vrsta acikličkog grafa kod koga su grane generalizovane k-konektorima. U literaturi se ovakvi grafovi ponekad nazivaju i hipergrafovima. Za zadati problem AND-OR graf predstavljen je na slici 50. Početni čvor pretrage za nalaženje rešenja je čvor 6, a ciljni čvor je 1 jer sve cifre treba zameniti jedinicama.

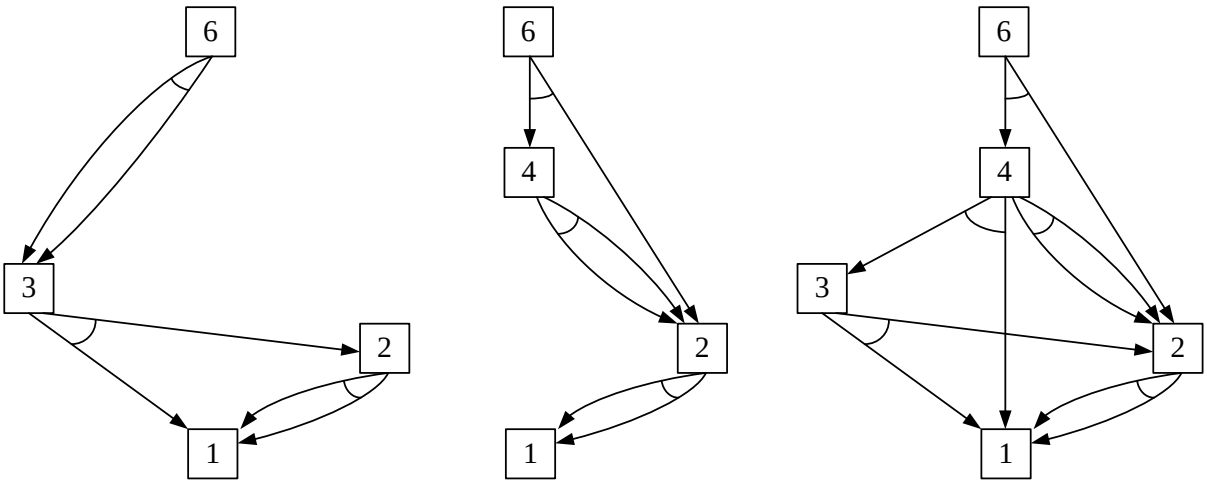


Slika 50

Kod 'klasičnih' metoda pretrage, rešenje je predstavljeno putanjom u grafu pretrage od početnog do nekog od ciljnih čvorova. Pri korišćenju AND/OR grafova, cilj se predstavlja (u opštem slučaju) skupom ciljnih čvorova *N*. Rešenje je predstavljeno podgrafom G' kompletnog grafa pretrage G. Rešenje se, ako postoji, dobija tako što se, polazeći od startnog čvora *n,* izabere jedan od konektora koji od čvora *n* vodi ka čvorovima-naslednicima n_1, n_2,..., n_k. Ukoliko svaki od čvorova naslednika predstavlja ciljni čvor (dakle jedan od čvorova iz skupa ciljnih čvorova *N*), rešenje je pronađeno i sastoji se od izabranih čvorova povezanih izabranim konektorom. U suprotnom slučaju, za svaki od čvorova naslednika koji nije ciljni čvor, potrebno je izabrati jedan od konektora i uključiti taj konektor i njegove čvorove-naslednike u rešenje. Procedura uključivanja novih konektora i čvorova u rešenje ponavlja se sve dok u podgrafu G' postoji čvor koji nije ciljni, a za koji nije izabran konektor.

S obzirom da procedura određivanja rešenja u AND/OR grafu uključuje proizvoljan izbor konektora, u opštem slučaju postoji više rešenja određenog problema pretrage. Razmotrimo

nalaženje rešenja za dati problem na osnovu AND/OR grafa sa slike 50. Za dati problem postoje tri različita rešenja prikazana na slici 51. Rešenje dobijamo polazeći od čvora 6. Moguće je izabrati jedan od dva konektora koji predstavljaju pravila P1 i P2. Ukoliko izaberemo levi konektor koji odgovara pravilu P1, u podgraf rešenja uključujemo startni čvor, izabrani konektor i čvor 3 koji predstavlja jedinog naslednika čvora 6 po konektoru P1. S obzirom da čvor 3 nije rešenje problema (da se podsetimo, jedini ciljni čvor je čvor 1), potrebno je izabrati izlazni konektor iz čvora 3. Radi se o jednom jedinom konektoru koji predstavlja pravilo P5. U podgraf rešenja uključujemo izabrani konektor P5 i čvorove-naslednike čvora 3 po konektoru P5, a to su čvorovi 2 i 1. Sada je potrebno za čvor 2 izabrati jedini izlazni konektor P6 i dodati taj konektor u rešenje čime se dobija kompletno rešenje prikazano na slici 51*a*. Ukoliko se u prvom koraku nalaženja rešenja izabere alternativni izlazni konektor čvora 6, moguće je dobiti druga dva moguća rešenja datog problema prikazana na slici 51*b* i *c*.



(a) (b) (c)

Slika 51

Konektorima u AND/OR grafu, mogu se pridružiti cene koje reprezentuju cene upotrebe određenih pravila pri rešavanju problema. Na osnovu ovih cena može se definisati cena određenog rešenja čime se ustanovljava kriterijum za poređenje različitih rešenja. Cena k(*n*,*N*) za neki podgraf G' grafa G od startnog čvora *n* do skupa ciljnih čvorova *N* definiše se sledećom rekurzivnom formulom:

- Ako je *n* element skupa *N*, onda je *k*(*n*, *N*) = 0.
- Inače, čvor *n* poseduje izlazni konektor prema skupu čvorova $n_1,n_2,...,n_i$. Neka je cena ovog konektora c_n. Tada je cena k(*n* , *N*) kompletnog rešenja jednaka zbiru cena izlaznog konektora čvora *n* i cena svih podgrafova od čvorova naslednika do ciljnih čvorova iz skupa *N*:

$$k(n,N) = c_n + k(n_1, N) + k(n_2, N) + ... + k(n_i, N)$$

Odredimo cenu rešenja sa slike 51*a* prema ovoj definiciji:

$$k(6,\{1\}) = k(P1) + k(3,\{1\}) + k(3,\{1\})$$

Kada se zamene cene podgrafova iz pojedinih međučvorova koje iznose

k(3,{1}) = k(P5) + k(2,{1}) + k(1,{1})

k(2,{1}) = k(P6) + k(1,{1}) + k(1,{1})

k(1,{1}) = 0

dobija se

k(6,{1}) = k(P1) + 2*k(P5) + 2*k(P6) = 10

s obzirom da je cena svakog od pravila jednaka 2.

Treba primetiti da se u ceni rešenja cene pojedinih konektora uračunavaju više puta. U opštem slučaju, ukoliko postoji *m* različitih putanja u grafu rešenja od startnog čvora do nekog čvora *n*, u cenu rešenja biće uračunata *m* puta cena izlaznog konektora čvora *n*. Ovo je logično, s obzirom da se pravilo predstavljeno tim konektorom mora primeniti *m* puta da bi se došlo do rešenja.

Radi ilustacije prethodne diskusije posmatrajmo ponovo rešenje sa slike 51*a*. U cenu ovog rešenja, uračunata je dva puta cena pravila P5. Da bismo dobili rešenje, moramo cifru šest zameniti sa dve cifre 3, a zatim *svaku od* trojki zameniti primenom pravila P5. Prema tome, tokom zamene se dva puta primenjuje pravilo P5 pa je i logično da se cena ovog pravila uračuna dvostruko u cenu kompletnog rešenja.

Analogija A* algoritmu u slučaju pretrage korišćenjem AND-OR acikličkih grafova je algoritam pretrage AO*, naveden u dodatku 1 (algoritam 10). Ovaj algoritam garantuje pronalaženje optimalnog rešenja kada se cena rešenja definiše na opisani način. Pretragu algoritmom AO* usmerava heuristička funkcija koja se definiše za svaki čvor AND/OR grafa. Heuristička funkcija za čvor *n* mora da ispunjava određeni uslov da bi pronađeno rešenje bilo optimalno, analogno slučaju kada se koristi algoritam A*. U slučaju algoritma AO* heuristička funkcija h(n) za proizvoljan čvor n AND/OR grafa mora da predstavlja potcenjenu cenu optimalnog podgrafa rešenja od čvora n do ciljnih čvorova.

Izvršavanje algoritma AO* sastoji se iz ponavljanja dve glavne faze:

- ekspanzije izabranog čvora grafa
- revizije funkcija procene čvorova grafa.

Prva faza je ekspanzija izabranog čvora AND/OR grafa i dodavanje njegovih izlaznih konektora i čvorova-naslednika u graf. Jedan od konektora, koji ima najbolju funkciju procene, pri tome biva obeležen kao rezultat druge glavne faze algoritma. U svakom trenutku pretrage *podgraf najboljeg parcijalnog rešenja* (analogno najboljoj parcijalnoj putanji kod A* algoritma) može se dobiti polazeći od startnog čvora grafa i prateći obeležene konektore. Sledeći čvor koji će biti ekspandovan uvek je jedan od čvorova koji pripadaju podgrafu najboljeg parcijalnog rešenja koji nije ciljni čvor. Rešenje je nađeno kada se startni čvor obeleži kao REŠEN (čvor *n* je rešen kada su ekspandovani svi čvorovi koji nisu ciljni u podgrafu koji polazi od čvora *n* i ide preko obeleženih konektora do ciljnih čvorova).

Druga glavna faza algoritma je revizija funkcija procene čvorova u grafu. Funkcija procene f(*n*) čvora *n* predstavlja procenu cene podgrafa optimalnog rešenja od čvora *n* do skupa ciljnih čvorova. Kada se neki čvor *n* unese u graf, a pre nego što se taj čvor ekspanduje, njegova funkcija procene f(*n*) je inicijalno jednaka vrednosti njegove heurističke funkcije h(*n*). Za razliku od algoritma A*, gde se funkcija procene računa samo jedanput za svaki čvor stabla pretrage, u AO* algoritmu postoji potreba za revizijom funkcija procene čvorova. Kada se čvor *n* ekspanduje, njegova funkcija procene se ažurira na osnovu cena izlaznih konektora iz

tog čvora i vrednosti funkcija procene čvorova naslednika. Ako iz čvora *n* ide više izlaznih konektora, uzima se najbolja (to jest najmanja) vrednost po nekom od konektora i taj konektor se obeležava čime se produžava najbolje parcijalno rešenje. Revidiranu vrednost funkcije procene čvora *n* potrebno je proslediti nagore u grafu. Pri tome se (eventualno) ažuriraju funkcije procene čvorova-prethodnika čvora *n* po *obeleženim* konektorima. Čvorove-prethodnike po neobeleženim konektorima nije potrebno razmatrati. Pošto je procena uvek potcenjena veličina, a ažuriranjem se dobijaju preciznije procene, vrednost funkcije procene ažuriranjem može samo da se poveća. Ažuriranje može da dovede do toga da za određeni čvor funkcija procene po nekom od konektora koji nije obeležen postane povoljnija od vrednosti po obeleženom konektoru (s obzirom da se ova poslednja ažuriranjem povećala) , pa je tada potrebno premestiti obeležje na neobeleženi konektor.

Primenimo AO* algoritam na zadati problem. U postavci problema definisane su kako heuristička funkcija za svaki čvor AND/OR grafa, to jest za svaku cifru, tako i cene pojedinih konektora.

Graf pretrage se inicijalno sastoji samo od startnog čvora 6 za koga je vrednost funkcije procene jednaka vrednosti njegove heurističke funkcije i iznosi 6. Razvijanjem čvora 6 u graf unosimo izlazne konektore čvora 6 i čvorove 2, 3 i 4 (slika 52*a*).

Vrednosti funkcije procene za ove čvorove jednake su vrednostima njihovih heurističkih funkcija (na slici 52*a* trenutne vrednosti funkcija procene napisane su pored svakog čvora). Sada se revidira funkcija procene čvora 6. Po levom konektoru, vrednost funkcije procene za čvor 6 jednaka je dvostrukoj vrednosti funkcije procene čvora 3 (jer je to jedini čvor naslednik po levom 2-konektoru) na koju se dodaje cena levog konektora, odnosno:

$f_{P1} = c_{P1} + f(3) + f(3) = 2 + 3 + 3 = 8$

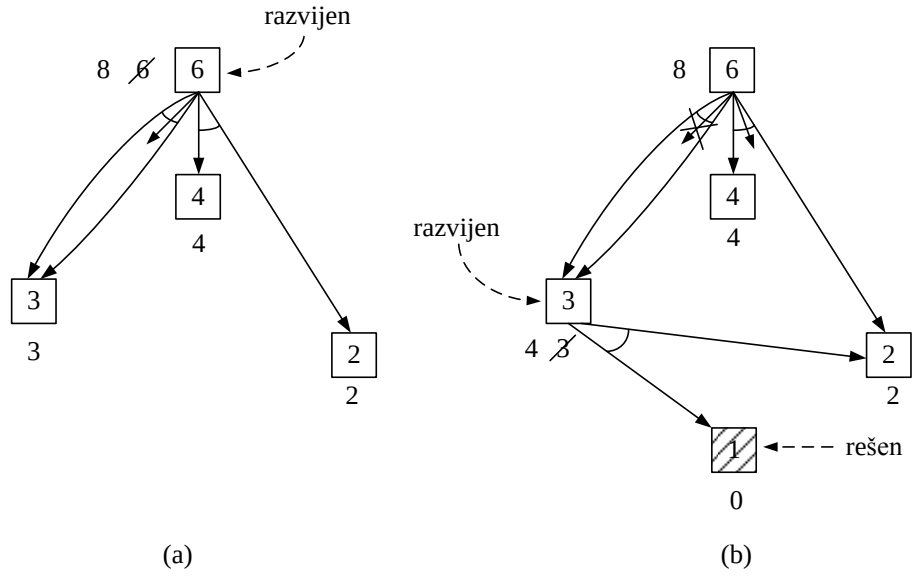
gde je sa P1 označen levi konektor na slici 52*a*. Na sličan način, vrednost funkcije procene po desnom konektoru je:

$f_{P2} = c_{P2} + f(4) + f(2) = 2 + 4 + 2 = 8$

Prema tome, ažurirana vrednost funkcije procene za čvor 6 je:

$f(6) = \min (f_{P1}, f_{P2}) = 8$

Potrebno je markirati jedan od konektora čvora 6 koji odgovara najboljem parcijalnom rešenju. S obzirom da su u konkretnom slučaju funkcije procene po oba konektora jednake proizvoljno je izabran levi konektor i obeležen strelicom. S obzirom da čvor 6 nema prethodnika u grafu, završena je faza ažuriranja funkcija procene čvorova.



Slika 52

U sledećoj iteraciji algoritma AO*, potrebno je za ekspanziju izabrati novi čvor. Taj čvor treba da se nalazi u podgrafu koji predstavlja najbolju (markiranu) parcijalnu putanju. U konkretnom slučaju radi se o čvoru 3. Razvijanjem ovoga čvora nastaje situacija prikazana na slici 52*b*. U graf je unesen izlazni konektor čvora 3 koji odgovara pravilu P5. Od novih čvorova u graf je unesen čvor 1. Pošto je ovo ciljni čvor, on je odmah (šrafiranjem) označen kao REŠEN. Čvor 3 ne možemo još označiti kao rešen, jer nije rešen čvor 2. Nova vrednost funkcije procene za čvor 3 jednaka je:

$f(3) = c_{P5} + f(2) + f(1) = 2 + 2 + 0 = 4.$

Pošto je promenjena vrednost funkcije procene za čvor 3 potrebno je, u skladu sa algoritmom, ažurirati procenu za čvor 6 kao prethodnika čvora 3 po označenom markeru P1. Nova vrednost funkcije procene po konektoru P1 iznosi:

$f_{P1} = c_{P1} + f(3) + f(3) = 2 + 4 + 4 = 10$

S obzirom da se funkcija procene po markeru P2 nije promenila i iznosi $f_{P2} = 8$, a pošto je

$f(6) = \min (f_{P1}, f_{P2}) = 8$

nema promene u vrednosti procene za čvor 6. Međutim, pošto se sada ova procena dobija po desnom konektoru, potrebno je ukloniti marker to jest strelicu sa konektora P1 i markirati konektor P2. Ovim je završena faza ažuriranja funkcija procene čvorova.

Pretraga se nastavlja razvijanjem jednog od nerazvijenih čvorova na markiranoj putanji. S obzirom da čvor 1 predstavlja ciljni čvor, on ne dolazi u obzir za razvijanje, tako da potrebno izabrati između čvorova 4 i 2.

Algoritam AO* ne specificira koji čvor treba izabrati pa je na proizvoljan način izabran čvor 4 (naravno, treba primetiti da je za dobijanje rešenja neophodno naknadno razviti i čvor 2 tako da ovaj izbor i nema preveliki uticaj na efikasnost pretrage). Slika 53*a* predstavlja graf pretrage posle razvoja čvora 4.

Funkcije procene za čvor 4 po konektorima P3 i P4 su:

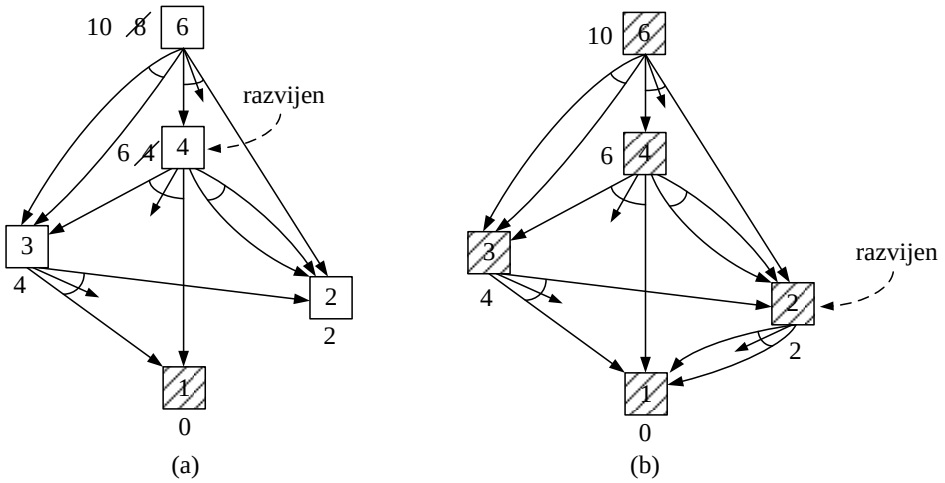
$f_{P3} = c_{P3} + f(2) + f(2) = 2 + 2 + 2 = 6$

$f_{P4} = c_{P4} + f(3) + f(1) = 2 + 4 + 0 = 6$

pa je nova vrednost funkcije procene za čvor 4:

$f(4) = \min (f_{P3}, f_{P4}) = 6$

Na proizvoljan način (s obzirom na jednakost procena) između konektora P3 i P4 biramo P4 i markiramo ga. Sledi ažuriranje funkcije procene čvora 6, prethodnika čvora 4 po obeleženom konektoru. Nova vrednost f(6) je 10 jer su sada procene i po konektoru P1 i po konektoru P2 jednake 10. Konektor P2 ostaje obeležen jer nije dobijena bolja procena.



Slika 53

Sada se razvija poslednji od nerazvijenih čvorova a to je čvor 2 (slika 53*b*). Vrednost funkcije procene čvora 2 se ne menja jer je sada:

$f(2) = c_{P6} + f(1) + f(1) = 2 + 0 + 0 = 2$

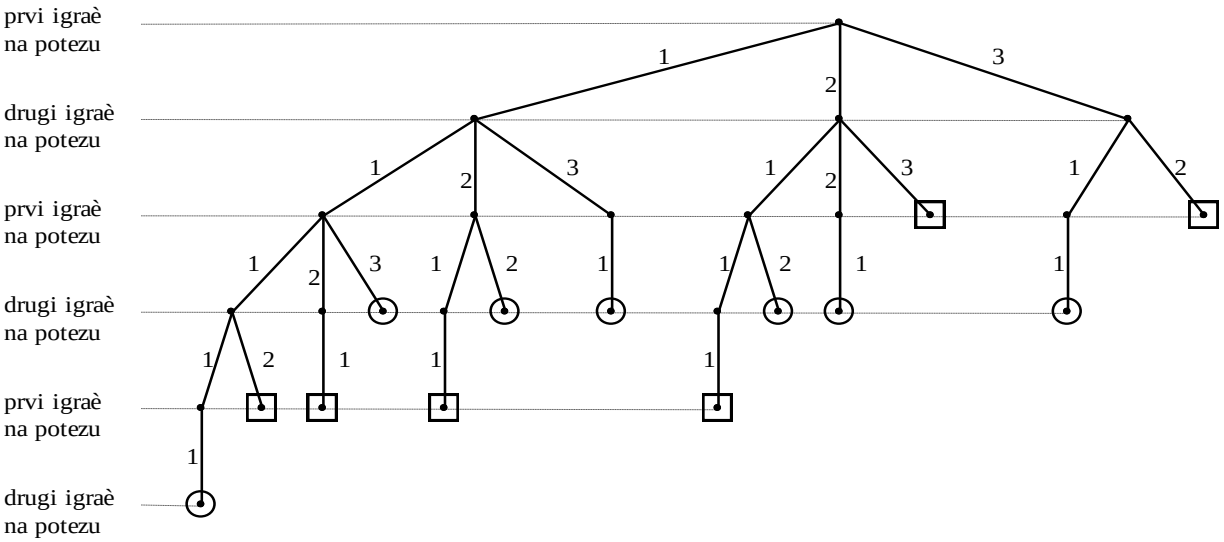
pa nema potrebe ažurirati procene ostalih čvorova u grafu. Čvor 2 je rešen, jer je ciljni čvor 1 njegov jedini naslednik po izlaznom konektoru P6. Potrebno je razmotriti rešenost čvorova - prethodnika čvora 2 po obeleženim konektorima. Čvor 3 je rešen jer su oba njegova naslednika po obeleženom konektoru P5 rešena. Iz istih razloga rešeni su i čvorovi 4 i 6. Pošto je startni čvor 6 rešen, pretraga se završava. Funkcija procene startnog čvora daje tačnu cenu rešenja koja iznosi 10 u ovom slučaju. Dobijeno je rešenje sa slike 51*c*. S obzirom da smo u pojedinim trenucima pretrage birali konektore na proizvoljan način, mogli mogli smo dobiti i jedno od druga dva rešenja jer su takođe optimalna kao i dobijeno rešenje.

Zadatak 19: Igra nim

Igra *nim* se izvodi na sledeći način: Dva igrača naizmenično uklanjaju jedan, dva ili tri metalna novčića sa steka koji sadrži na početku pet novčića. Igrač koji uzme poslednji novčić gubi. Pokazati da igrač koji je drugi na potezu može uvek da pobedi. Koja bi bila dobitnička strategija?

Rešenje

Na slici 54 prikazano je kompletno stablo pretrage ove igre. Čvorovi stabla su pozicije u igri. Uz svaku granu stoji broj novčića koji bivaju uklonjeni da bi se iz jedne pozicije prešlo u drugu.



Slika 54

Listovi stabla odgovaraju završnoj poziciji u kojoj više nema nijednog novčića na steku. Listovi obeleženi sa ○ odgovaraju pozicijama u kojima gubi igrač koji je prvi na potezu jer se do ovih listova stiže primenom neparnog broja operatora uzimanja novčića. Listovi obeleženi sa □ odgovaraju pozicijama u kojima gubi igrač koji je drugi na potezu.

Radi sistematičnog određivanja dobitničke strategije drugog igrača konstruisaćemo AND/OR graf igre nim na sledeći način:

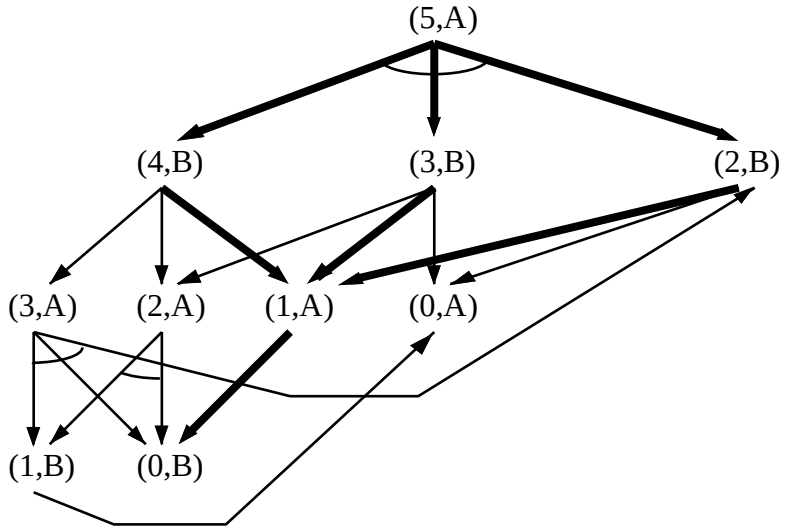
- Stanja ćemo predstaviti uređenim parom u kome prva komponenta označava broj novčića na steku (od 0 do 5), a druga igrača koji je na potezu (A ili B). Početno stanje kodiramo kao (5,A) jer je na steku svih pet novčića, a prvi igrač je na potezu.

- Ciljno stanje grafa je stanje (0,B) jer ono označava da je igrač A sa steka uklonio poslednji novčić, to jest da je igrač B pobedio.

- Operatori promene stanja su skidanje sa steka jednog, dva ili tri novčića i predstavljaće grane AND/OR grafa.

- Čvorove oblika (*x*,A), dakle stanja u kojima je prvi igrač na potezu definišemo kao AND čvorove, a čvorove oblika (*y*,B) to jest stanja u kojima je drugi igrač na potezu definišemo kao OR čvorove. Sa tačke gledišta igrača B mora postojati rešenje (putanja u grafu) za svaki mogući potez igrača A (odnosno za svakog naslednika AND čvora) i bar za jedan potez igrača B iz svake pozicije (odnosno bar za jednog naslednika OR čvora).

AND/OR graf igre nim konstruisan prema izloženim principima prikazan je na slici 55. Graf je acikličan, s obzirom da se u svakom potezu sa steka uklanja bar po jedan novčić tako da se ne može desiti povratak u neko od ranijih stanja u igri.



Slika 55

Debljim linijama prikazan je podgraf koji predstavlja rešenje datog problema. S obzirom da se radi o relativno jednostavnom grafu za nalaženje rešenja korišćen je rekurzivni algoritam definisan u zadatku 18. Rešenje je jedinstveno jer se mora uzeti u obzir da ono ne sme sadržati čvor (0,A) pošto taj čvor označava situaciju kada igrač A pobeđuje. To povlači da u rešenje ne smeju biti uključeni niti čvorovi (1,B), (3,A) i (2,A). Na ovaj način jednoznačno su određeni konektori koji se biraju u OR čvorovima grafa.

Dobijeno rešenje AND/OR grafa predstavlja matematički izraženu dobitničku strategiju za drugog igrača. Rečima opisana, dobitnička strategija bila bi da drugu igrač u svom prvom potezu sa steka skine sve novčiće osim jednog, što je uvek moguće bez obzira na potez prvog igrača. Na taj način će igra uvek biti završena u sledećem potezu prvog igrača, jer on mora da uzme novčić koji je ostao na steku.

Zadatak 20: Problem šest kraljica

Na šahovsku tablu dimenzije 6 x 6 potrebno je smestiti šest kraljica tako da nijedna od njih ne napada bilo koju drugu. Rešiti problem primenom neke od metoda pretraživanja po izboru.

Rešenje

Formulišimo najpre ovaj problem kao problem pretraživanja: startno stanje je prazna šahovska tabla. U svakom potezu smeštamo jednu kraljicu na tablu, tako da nije napadnuta od prethodno smeštenih kraljica. Problem je rešen kada sa na tablu smeste svih šest kraljica. Ovako definisan problem pretraživanja je *komutativan*, što znači da u rešenju predstavljeno nizom operatora možemo bez ograničenja permutovati redosled operatora.

Očigledno je da u istoj vrsti table ne mogu biti smeštene dve kraljice. Da bismo smanjili faktor grananja u stablu pretrage, usvojićemo sledeću taktiku: prvu kraljicu smeštamo u prvu vrstu table, drugu kraljicu u drugu vrstu, i tako dalje.. Na ovaj način, u svakom potezu treba se odlučiti za poziciju u okviru jedne vrste.

Heurističku funkciju moguće je definisati na sledeći način:

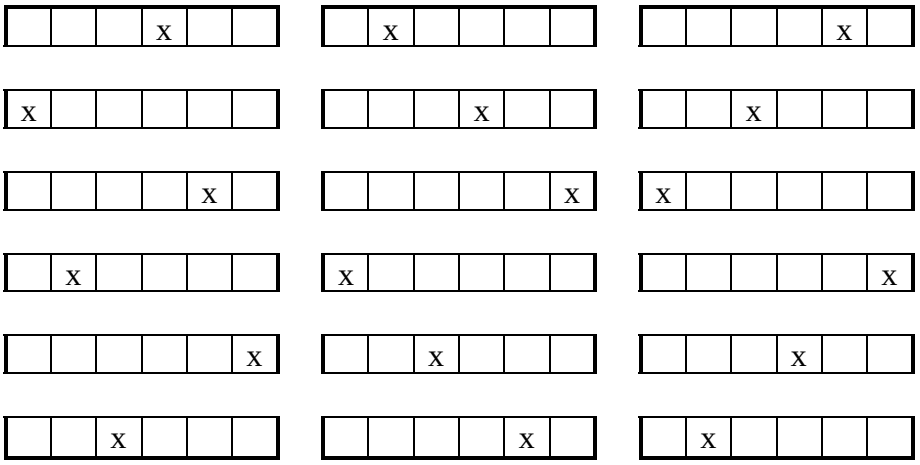
- Neka *diag*(*i*,*j*) označava dužinu najduže dijagonale koja prolazi kroz ćeliju tabele u vrsti *i* i koloni *j* (*i*,*j* ∈ {1,2,3,4,5,6}). Vrednost heurističke funkcije za smeštanje kraljice u vrstu *i* i kolonu *j* je:

$$h(i,j) = 6 * diag(i,j) + j$$

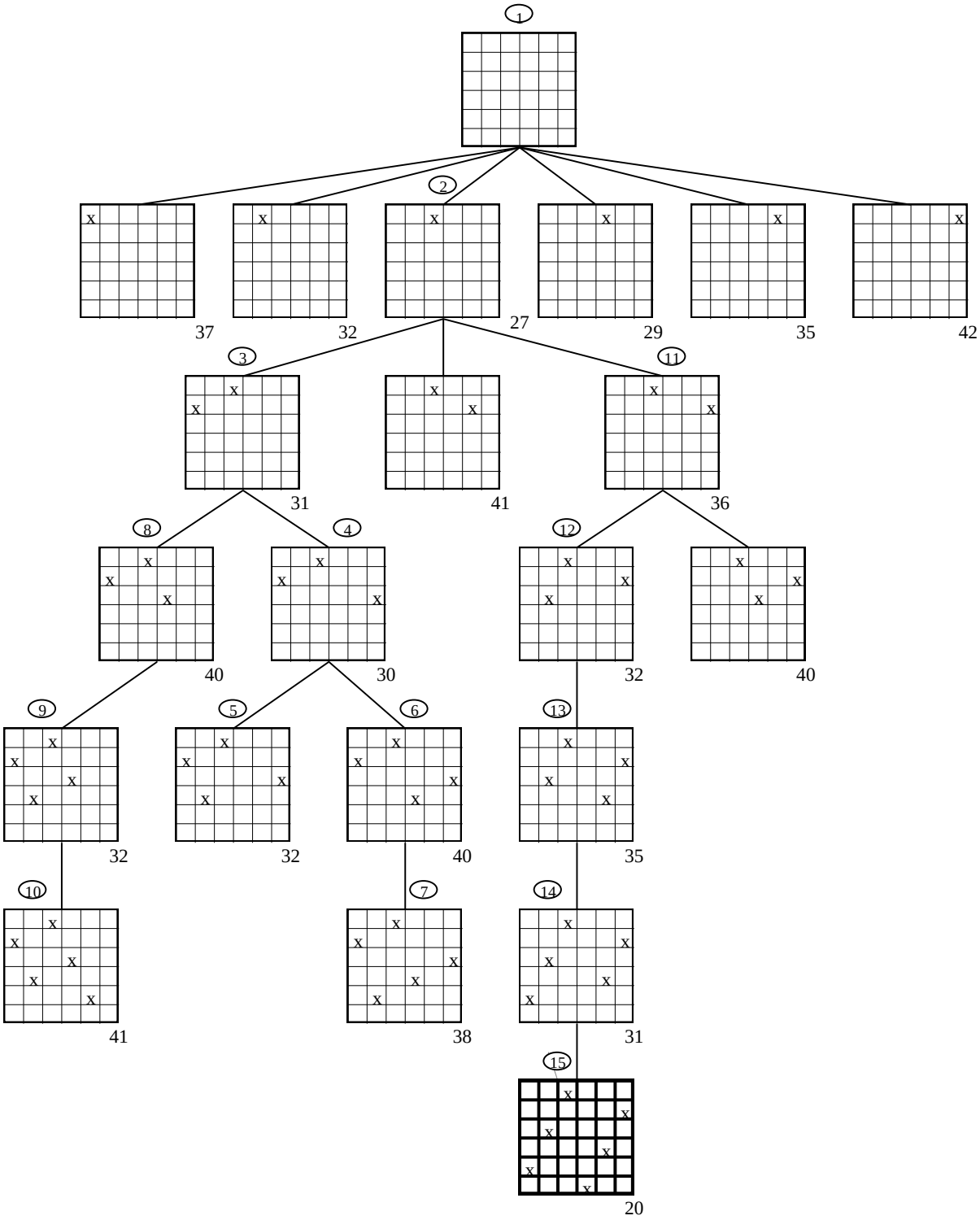
Drugim rečima, prednost dajemo pozicijama za manjom vrednošću *diag*. U slučaju kada u vrsti postoje dve pozicije sa istom vrednošću za *diag*, prednost ćemo dati levoj ćeliji. Treba primetiti da na vrednost heurističke funkcije ima uticaja jedino položaj poslednje postavljene kraljice. Drugim rečima, ova funkcija ne ocenjuje poziciju u globalu već samo lokalno među sledbenicima tekućeg čvora pretrage. Zbog toga će za pretragu biti primenjen metod planinarenja.

Pošto su definisani svi elementi potrebne za vršenje pretrage, moguće je primeniti algoritam planinarenja. Stablo pretrage prikazano je na slici 56. U toku procesa rešavanja obiđeno je ukupno 15 stanja u stablu pretrage i tri puta vršeno vraćanje unazad (*backtracking*) iz stanja koja nemaju naslednika, to jest, u kojima nije bilo moguće postaviti novu kraljicu tako da ne napada nijednu od već postavljenih kraljica.

Pored nađenog rešenja postoji još simetričnih rešenja problema koja se dobijaju na osnovu nađenog rešenja rotiranjem cele šahovske table:



Problem smeštanja 8 kraljica na tablu 8 x 8 moguće je rešiti na isti način kao i problem 6 x 6. Čitaocu se preporučuje da razmisli o heurističkoj funkciji koja bi ocenjivala tekuće stanje pretraživanja u globalu, tako da je moguće primeniti metod traženja 'prvo najbolji'.


1
2
37
32
27
29
35
42
3
11
31
41
36
8
4
12
40
30
32
40
9
5
6
13
32
32
40
35
10
7
14
41
38
31
15
20

Slika 56

Zadatak 21: Agenda

Pretpostavimo da se prostor pretraživanja može predstaviti kao potpuno (ispunjeno) binarno stablo od 31 stanja i četiri nivoa (ne računajući startno stanje kao nivo). Drugim rečima, svi čvorovi stabla sem listova poseduju po dva naslednika. Pretpostavimo takođe da postoji samo jedno ciljno stanje koje leži u prostoru pretraživanja.

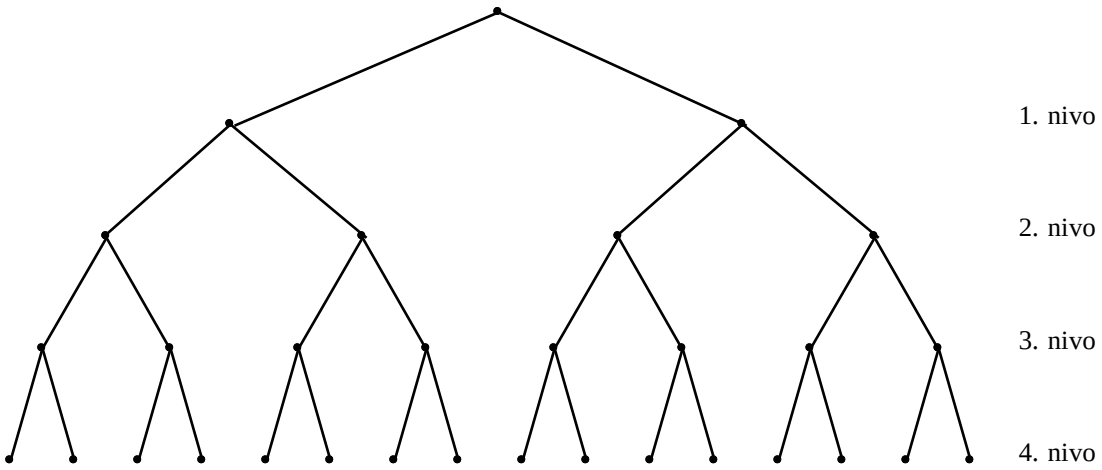
a) Kolika je maksimalna moguća veličina agende ako vršimo pretraživanje po širini?

b) Kolika je maksimalna moguća veličina agende ako vršimo pretraživanje 'prvo najbolji' ?

Rešenje

U toku pretrage skup čvorova koji su do tog trenutka uneti u stablo pretrage, a nisu ekspandovani predstavlja *agendu.* U algoritmima pretrage definisanim u dodatku 1 kao struktura podataka za pamćenje agende tokom pretrage koristi se lista. Procena maksimalne veličine agende za neki problem pretrage je od praktičnog značaja, da bi se mogao proceniti memorijski prostor potreban za tu pretragu. Maksimalna veličina agende tokom neke pretrage zavisi kako od upotrebljenog algoritma tako i od dimenzije problema (broja mogućih stanja) i konfiguracije grafa pretrage.

a) Stablo pretrage opisano u postavci zadatka prikazano je na slici 57. Pri pretrazi po širini, agenda u svakom trenutku pretrage sadrži čvorove sa istog nivoa stabla pretrage. Agenda se povećava tokom pretrage sa nivoom na stabla pretrage na kome se vrši ekspanzija stanja, tako da je agenda najveća u trenutku kada se ekspanduje poslednje stanje na trećem nivou stabla pretrage. Tada se u agendi nalaze svi čvorovi sa četvrtog nivoa kojih ima 16.

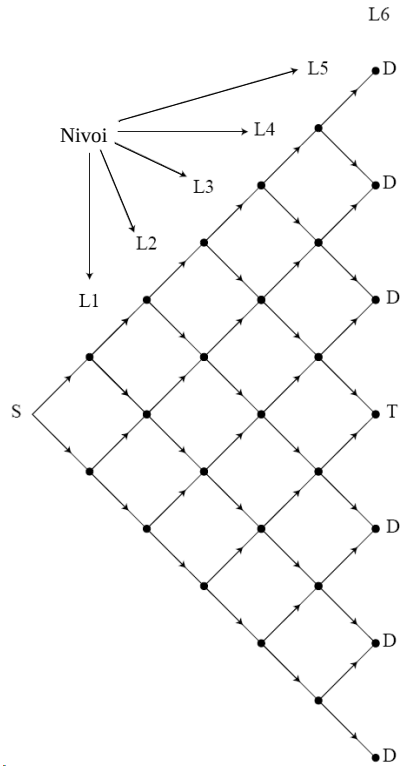


Slika 57

b) U ovom slučaju maksimalna moguća veličina agende je ista kao i u slučaju a) i iznosi 16, i to u trenutku pretrage kada ona sadrži sve čvorove četvrtog nivoa stabla pretrage. Kod pretrage 'prvo najbolji' agenda može sadržati stanja sa više različitih nivoa stabla pretrage. Ukoliko agenda sadrži (pored listova) i unutrašnje čvorove, sistematskom zamenom unutrašnjih čvorova njihovim sledbenicima povećava se veličina agende, da bi ona dostigla maksimalnu vrednost kada se u njoj nalaze samo listovi. Ovakva, za pretragu nepovoljna, situacija javlja se kada je heuristička funkcija loše definisana tako da se pretraga 'prvo najbolji' svodi na pretragu po širini.

Zadatak 22: Zalutala deca

Dvoje dece pobeglo je od roditelja i stoji u nepoznatom gradu koji je na mapi (slika 58) označen slovom S. Pošto su ogladneli žele da idu u restoran koji je lociran u čvoru T. Sve ulice su jednosmerne. Svaki put između čvorova dugačak je jednu jedinicu.



Slika 58

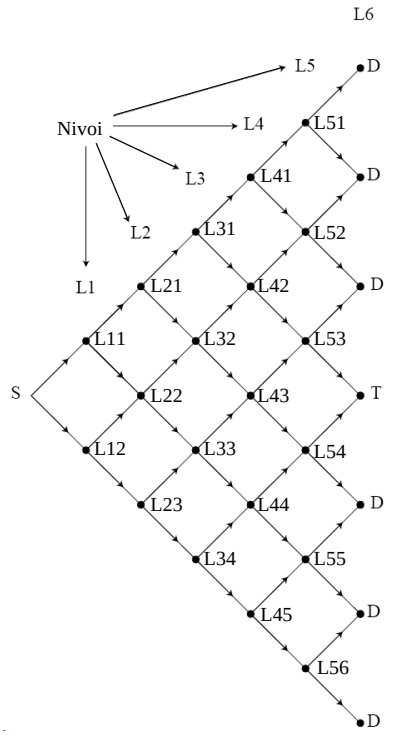
a) Teodora je odlučila da primeni metodu grananja i ograničavanja, bez upotrebe dinamičkog programiranja, kako bi pronašla najkraću putanju od S do T. Ukoliko se ikada dvoumi kojim putem da krene, ona bira put koji je najbliži vrhu strane. Pre nego što bude sigurna da je pronašla najkraći put, ona pronalazi n puteva od kojih svaki počinje u čvoru S i završava se u čvoru koji je obeležen bilo sa T bilo sa D i dodaje taj put u red puteva. Kolika je vrednost za n. Obrazložiti. Prikazati određeni broj koraka (više od 5) pretraživanja opisanom metodom.

b) Uroš misli da bi bilo bolje koristiti i procenu udaljenosti uz grananje i ograničavanje. Takođe on, za razliku od Teodore, primenjuje dinamičko programiranje. Procena udaljenosti za svaki čvor je prava linija do čvora T. Da li Uroš koristi korektnu heuristiku? Obrazložiti. Ukoliko se ikada dvoumi kojim putem da krene, on bira putanju koja je bliža vrhu strane. Pre nego što bude siguran da je pronašao najkraći put (do ciljnog čvora), on pronalazi n puteva od startnog čvora S do jednog od krajnjih čvorova T ili D i dodaje ih u listu puteva. Koliko je n u ovom slučaju? Prikazati postupak pretraživanja.

Rešenje

U cilju lakšeg razumevanja rešenja zadatka, slika 59 prikazuje detaljnije označenu sliku datu postavkom. Svaki čvor grafa nalazi se na nekom nivou u odnosu na startni čvor (L1 – prvi nivo udaljenost, L2 – drugi nivo udaljenosti, itd.). Dodatno svakom čvoru pridruženo je odgovarajuće ime L_{xy}, gde x označava nivo čvora, a y redni broj čvora na nivou x, pri čemu se

numeracija vrši od vrha ka dnu strane. Na primer, sa L_{42} označen je drugi čvor od vrha strane na četvrtom nivou.

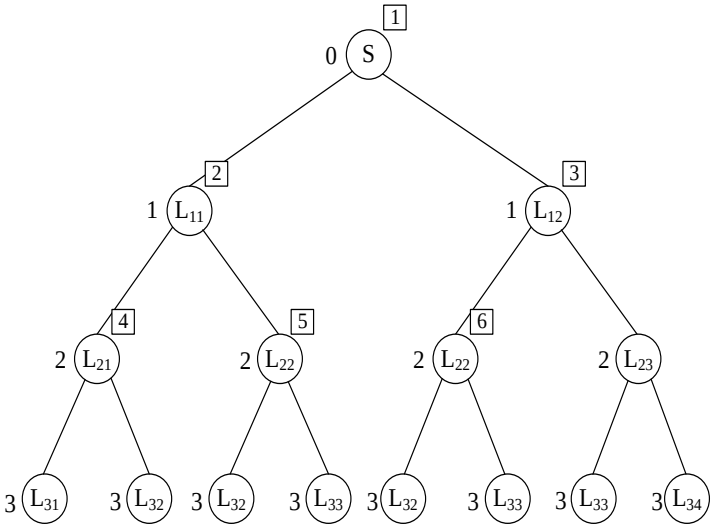


Slika 59

a) Kako je od ranije poznato, metod grananja i ograničavanja daje optimalno rešenje u odnosu na broj grana, od startnog do odredišnog čvora. Za detalje o algoritmu grananja i ograničavanja pogledati algoritam 5 u dodatku 1.

Razmotrimo kako Teodora obavlja obilazak stabla u skladu sa navedenim algoritmom. Startni čvor S ima dva naslednika, čvor L_{11} i L_{12}. Na osnovu toga kreiraju se dve parcijalne putanje S, L_{11} i S, L_{12}, čije su cene koštanja 1, i dodaju u listu putanja. Sledeće, potrebno je iz liste putanja izabrati onu sa najmanjom cenom koštanja. Kako obe putanje u listi imaju jednaku cenu koštanja potrebno je razmotriti izbor putanje definisan postavkom zadatka. Kako je definisano postavkom, potrebno je izabrati putanju čiji je poslednji čvor bliži vrhu strane. U našem slučaju u pitanju je putanja S, L_{11}. Kreiraju se dve nove parcijalne putanje produžavanjem izabrane i to S, L_{11}, L_{21} i S, L_{11}, L_{21}, koje se dodaju u listu putanja. Cena koštanja obe putanje je 2. Prvih šest ekspandovanih čvorova prikazano je na slici 60. Dalje se postupak sprovodi pravolinijski.

Bitno je uočiti šablon koji se javlja prilikom kreiranja i obilaska parcijalnih putanja. Iz startnog čvora kreirane su dve putanje, obe jedinične cene. Zatim je na osnovu svake od njih kreirano po dve putanje, sve četiri cene 2. Ovo dovodi do zaključka da će čvorovi na nivou n+1 biti obiđeni tek nakon obilaska svih čvorova na nivou n. Drugo, na osnovu opisanog, lako možemo odrediti, da bi broj putanja koje bi bile kreirane obilaskom kompletnog grafa, iznosio $2^6 = 64$, što je ujedno i broj putanja koje Teodora obilazi pre utvrđivanja rešenja.



Slika 60

b) Kako Uroš tokom obilaska pored cene putanje razmatra i procenu udaljenosti (heuristiku), može se zaključiti da primenjuje A* algoritam. S obzirom da kao heurističku funkciju razmatra vazdušno rastojanje, što je potcenjena vrednost stvarne udaljenost, može se zaključiti da je heuristika korektno izabrana, odnosno pronađena putanja biće optimalna.

Heuristička funkcija može se opisati narednom formulom:

$$h = \sqrt{\left(\frac{(6-x)\cdot\sqrt{2}}{2}\right)^2 + \left(\frac{d\cdot\sqrt{2}}{2}\right)^2} = \sqrt{\frac{(6-x)^2 + d^2}{2}}$$

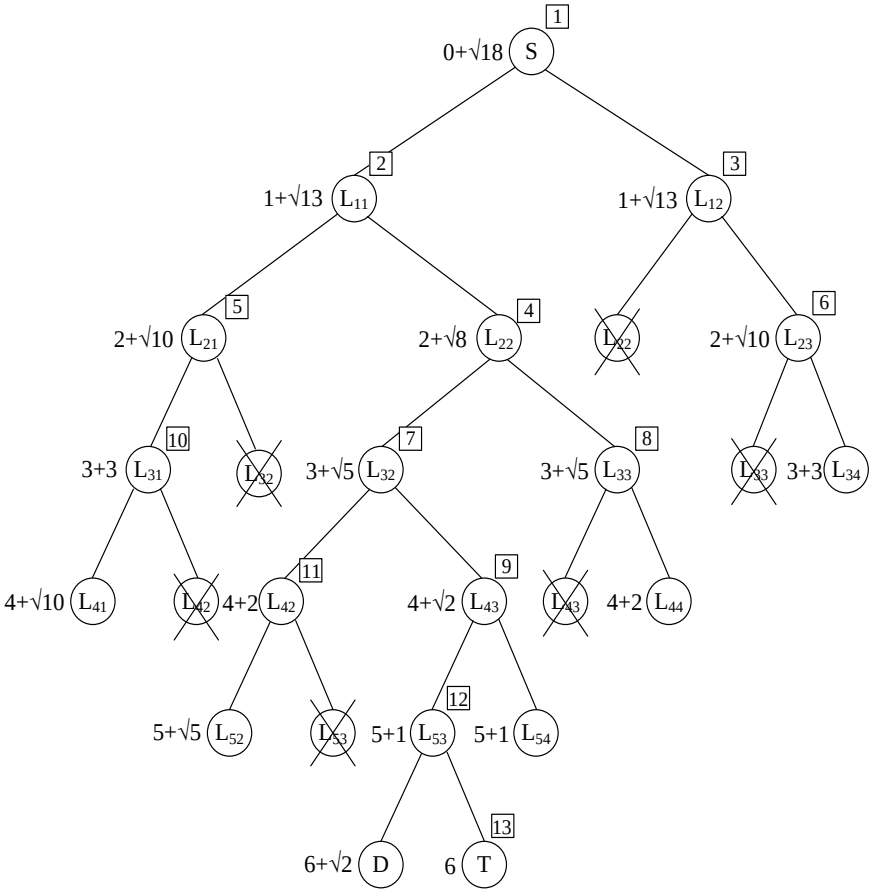
Sa *d* je obeleženo rastojanje čvora po vertikali od prave koja prolazi kroz tačke S i T. Na primer, d bi imalo vrednost 1 za čvorove L_{11}, L_{12}, L_{32}, L_{33}, L_{53}, L_{54}.

Cena koštanja nekog čvora zavisi od nivoa na kome se čvor nalazi. Za čvor na nivou x cena koštanja iznosi x.

Vrednost funkcije procene računa se na osnovu sledeće formule:

$$f = x + \sqrt{\left(\frac{(6-x)\cdot\sqrt{2}}{2}\right)^2 + \left(\frac{d\cdot\sqrt{2}}{2}\right)^2} = x + \sqrt{\frac{(6-x)^2 + d^2}{2}}$$

Obilazak stabla primenom A* algoritma prikazan je na slici 61. Pregledom stabla može se utvrditi da n iz postavke zadatka ima vrednost 2.



Slika 61

Zadatak 23: Odlazak na kafu

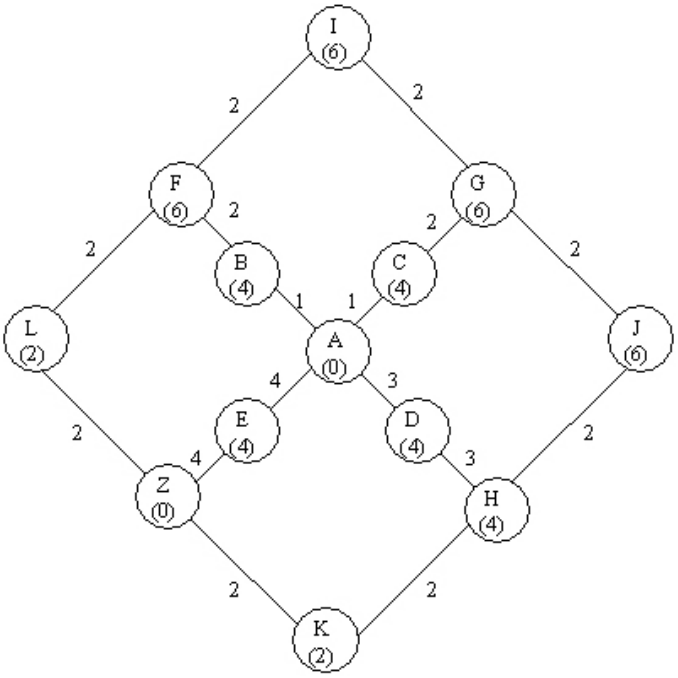
Grafom (slika 62) je opisano jedno malo, ali lepo mesto u Srbiji, Lipnički Šor. Raskrsnice su obeležene čvorovima grafa a putevi granama grafa. Dužina puteva naznačena je uz grane grafa. Na svakoj raskrsnici nalazi se po jedna kuća. Živka, koja živi u kući A, rešila je da jutarnju kafu popije sa Gordanom koja živi u kući Z. Živka nije sigurna kojim će putem najpre stići do kuće Z, ali zna koliki je napor potreban od svake raskrsnice do kuće Z. Napor potreban od svake raskrsnice naznačen je unutar svakog čvora.

(Kada je izbor raskrsnice dvosmislen potrebno je vršiti izbor po leksikografskom poretku (pr: ukoliko se za obilazak može izabrati čvor B ili C, potrebno je izabrati čvor B).

a) Izvršiti pretraživanje strategijom po širini. Usvojiti da je put do odredišnog čvora pronađen tek u trenutku njegove ekspanzije ne kada se ciljni čvor nađe u redu. Napisati kojim redom se vrši ekspanzija čvorova. Dati kompletnu putanju od startnog do ciljnog čvora.
b) Izvršiti pretraživanje strategijom po dubini. Usvojiti da je put do odredišnog čvora pronađen tek u trenutku njegove ekspanzije ne kada se ciljni čvor nađe u redu. Napisati kojim redom se vrši ekspanzija čvorova. Dati kompletnu putanju od startnog do ciljnog čvora.

c) Izvršiti pretraživanje metodom prvo-najbolji. Koristiti dinamičko programiranje tokom pretraživanja. Napisati kojim redom se vrši ekspanzija čvorova. Dati kompletnu putanju od startnog do ciljnog čvora.

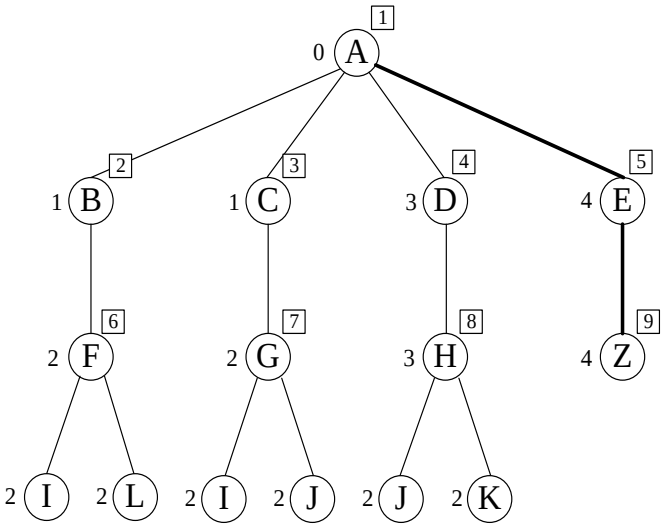
d) Izvršiti pretraživanje A* metodom. Prikazati promene liste parcijalnih putanja. Pretraga se završava kada se iz liste preuzme putanja u kojoj je Z poslednji čvor.



Slika 62

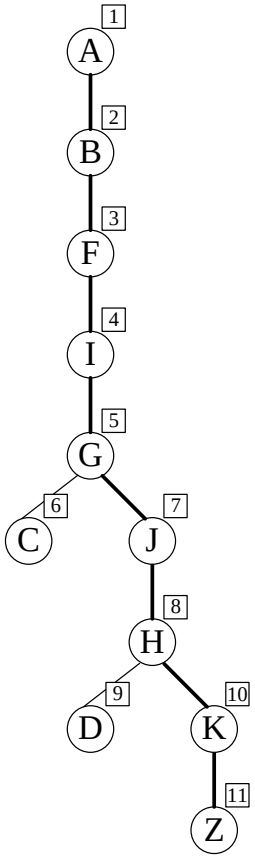
Rešenje

a) Na slici 63 prikazan je redosled ekspanzije (A, B, C, D, E, F, G, H, Z), kao i pronađena putanja (A, E, Z).



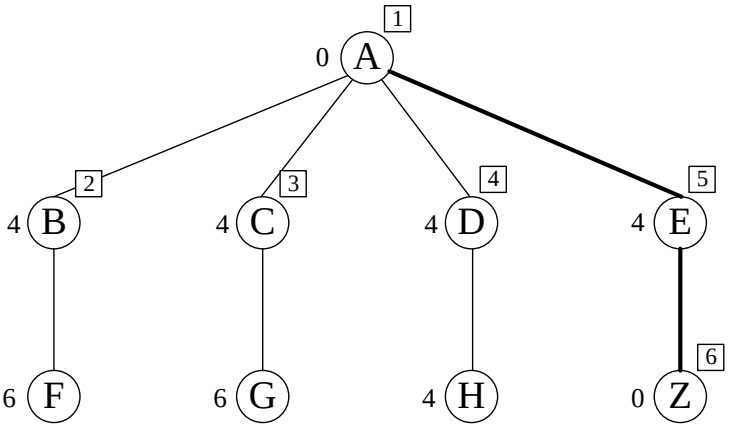
Slika 63

b) Primenom algoritma 2 iz dodatka 1 dobijeno je stablo pretrage prikazano na slici 64. Na slici je naznačen redosled ekspanzije čvorova (A, B, F, I, G, C, J, H, D, K, Z), kao i pronađena putanja (A, B, F, I, G, J, H, K, Z).



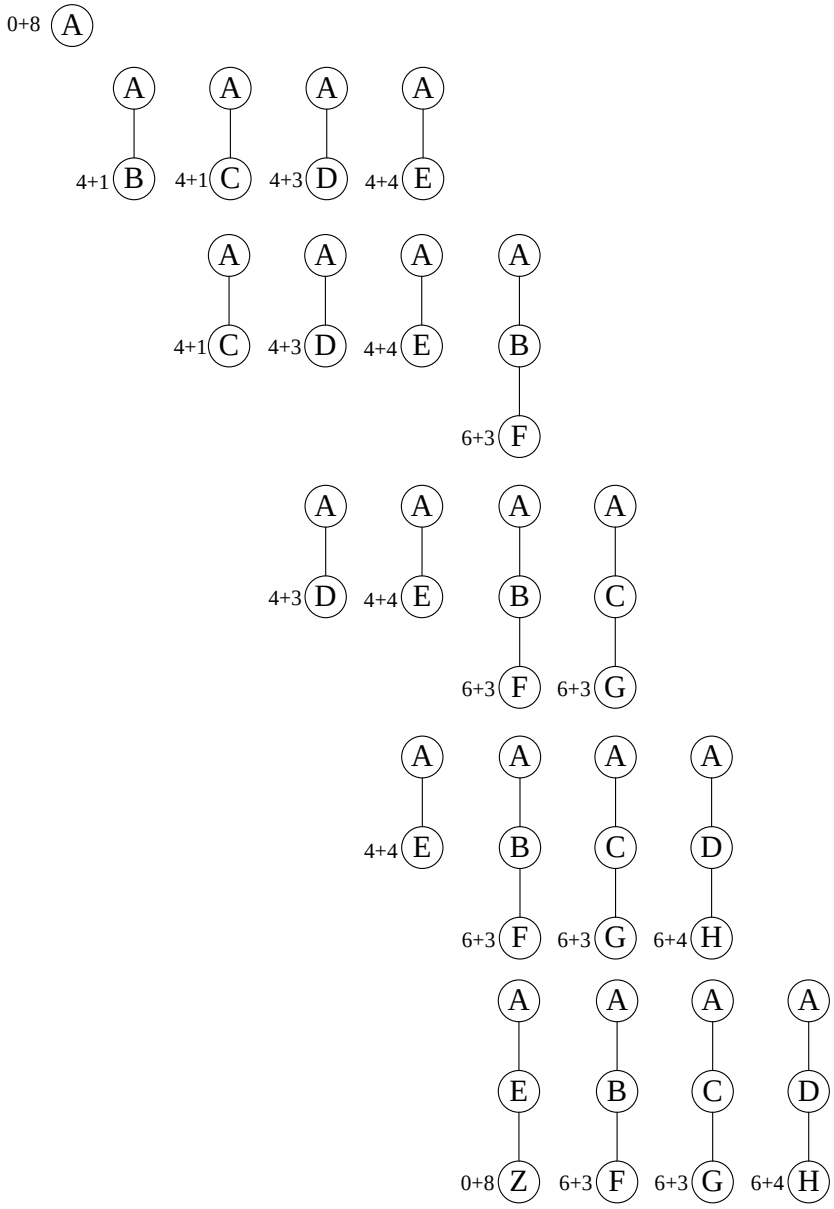
Slika 64

c) Na slici 65 prikazano je stablo pretrage primenom algoritma prvo-najbolji. Naznačen je redosled ekspanzije čvorova i pronađena putanja. Treba primetiti da je za ovu stavku naglašeno da je potrebno primeniti princip dinamičkog programiranja. Međutim, kako se može videti sa kreiranog stabla pretrage, nije postojala potreba za upotrebom ovog principa.



Slika 65

d) Slika 66 ilustruje promene liste parcijalnih putanja tokom obilaska stabla primenom A* metode. Prikazane liste putanja su sortirane. Kako se može videti sa slike, poslednje odabrana putanja je A, E, Z koja ujedno predstavlja i rešenje. Čitaocu se preporučuje da kreira stablo pretrage.



Slika 66

Zadatak 24: Utvrđivanje gradiva

Potrebno je pronaći putanju od startnog čvora S do odredišnog čvora G (slika 67), upotrebom različitih strategija pretraživanja.

Sve ulice su jednosmerne, sa leva na desno. Ukoliko se pojavi slučaj kada je izbor narednog čvora dvosmislen, potrebno je izabrati čvor bliži vrhu strane. Ukoliko se desi da nakon primene prethodnog pravila i dalje postoji više mogućih izbora onda je potrebno izabrati čvor bliži levom kraju strane.

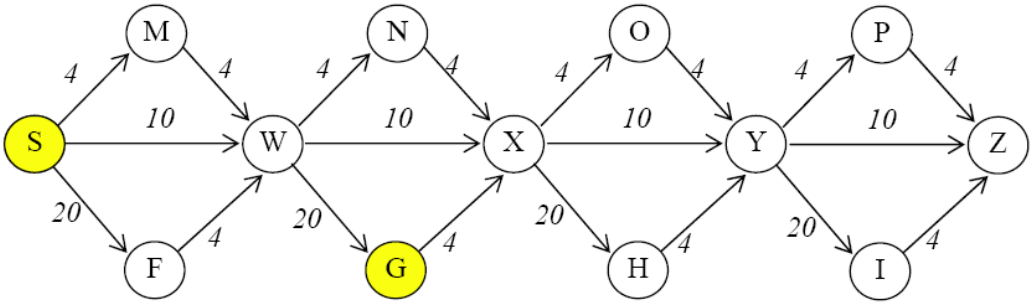
a) Primeniti strategiju pretraživanja po širini.

b) Primeniti strategiju pretraživanja po dubini. Prilikom pretrage primeniti princip dinamičkog programiranja.

c) Primeniti strategiju prvo-najbolji, u kombinaciji sa principom dinamičkog programiranja.

d) Primeniti strategiju grananja i ograničavanja, u kombinaciji sa principom dinamičkog programiranja.

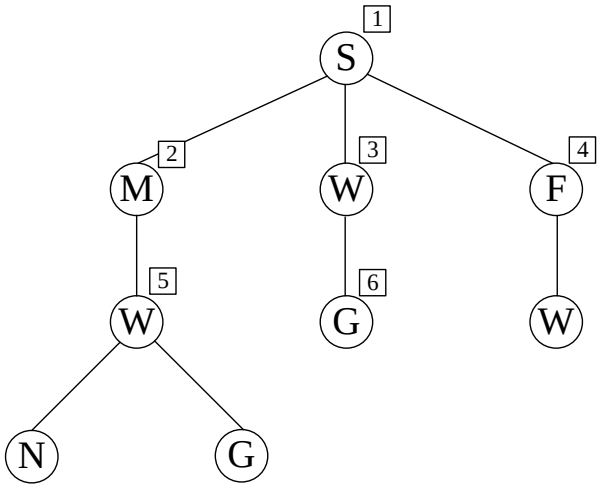
Cene koštanja prikazane su na slici.



Slika 67

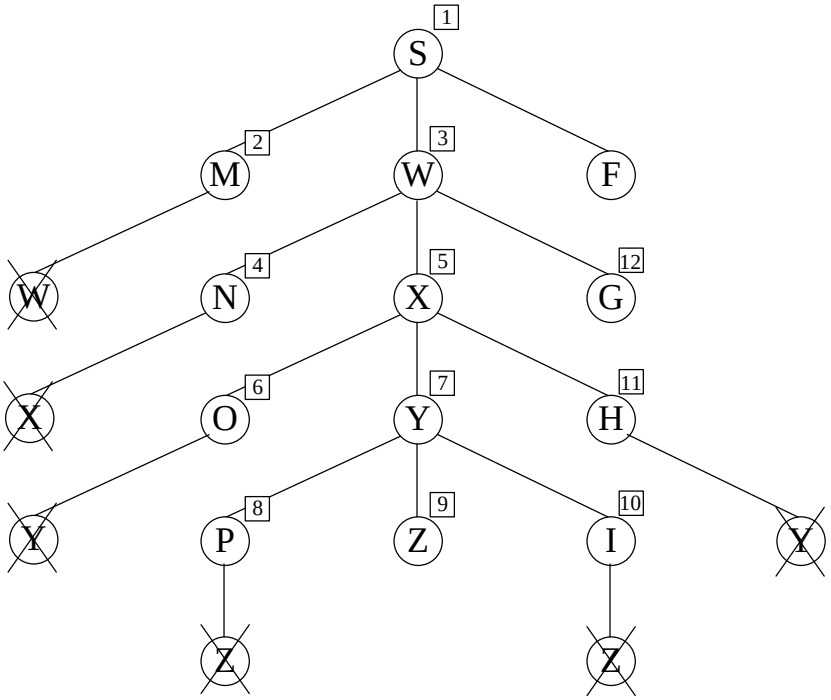
Rešenje

a) Primenom algoritma 1 iz dodatka 1 dobijeno je stablo pretrage prikazano na slici 68. Na stablu pretrage naznačen je redosled ekspanzije čvorova (S, M, W, F, W, G).



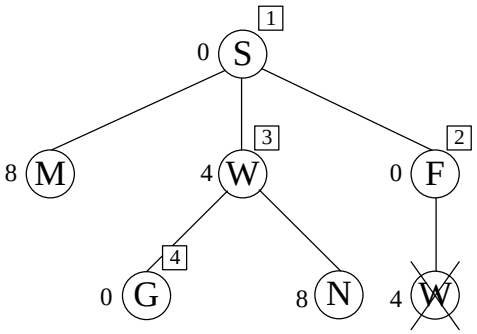
Slika 68

b) Uz standardni algoritam obilaska po dubini (algoritam 2, dodatak 1), primenjen je princip dinamičkog programiranja. Stablo pretrage prikazano je na slici 69. Precrtani čvorovi izuzeti su iz razmatranja na osnovu principa dinamičkog programiranja. Redosled ekspanzije čvorova naznačen je na stablu pretrage (S, M, W, N, X, O, Y, P, Z, I, H, G).



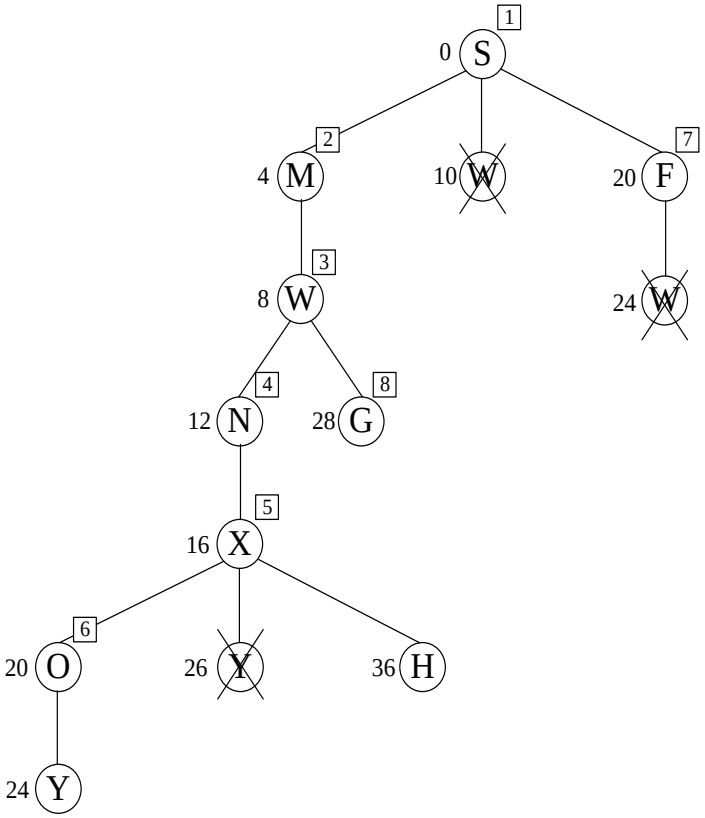
Slika 69

c) Uz algoritam 4 dat u dodatku 1 primenjen je princip dinamičkog programiranja. Rezultujuće stablo pretrage prikazano je na slici 70. Na stablu je naznačen redosled ekspanzije čvorova (S, F, W, G). Principom dinamičkog programiranja u ovom primeru iz razmatranja izuzet je samo jedan čvor, W (precrtan na slici).



Slika 70

d) Primenom metode grananja i ograničavanja dobijeno je stablo pretrage prikazano na slici 71.



Slika 71

Zadatak 25: Heuristika za A*

Pretpostavimo da za rešavanje nekog problema pretraživanja želimo da primenimo metodu A*. Neka je za tu primenu pronađena heuristička funkcija udaljenost do cilja najviše za K jedinica. Kako se može dobiti garantovano optimalno rešenje za A* pretraživanje?

Rešenje

Neka je d(S) stvarna, a h(S) procenjena udaljenost do cilja u proizvoljnom stanju pretrage S. Po uslovu zadatka je:

$$h(S) - d(S) \leq K$$

Preuređivanjem ove nejednakosti dobija se:

$$h(S) - K \leq d(S)$$

Poslednju nejednakost tumačimo na sledeći način: Veličina h(S) - K uvek predstavlja potcenjenu meru udaljenosti do cilja u proizvoljnom stanju pretrage S. Rešenje dobijeno primenom algoritma pretrage A* je garantovano optimalno ukoliko heuristička funkcija u svakom stanju pretrage predstavlja potcenjenu vrednost stvarne udaljenosti do cilja. Prema tome, modifikovanu heurističku funkciju h' koja garantuje optimalnost rešenja možemo definisati koristeći funkciju h na sledeći način:

$$h'(S) = \max [0 , h(S) - K]$$

pri čemu je još dodatno obezbeđena nenegativnost heurističke funkcije h'(S).

Zadatak 26: Projektovanje štampanog kola

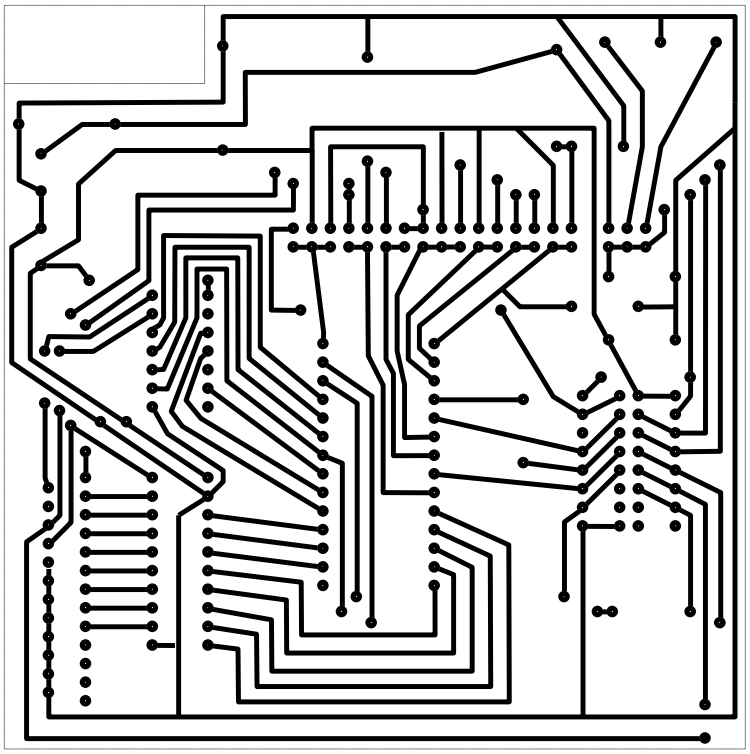
Posmatrajmo problem projektovanja električnih veza između tačaka na površini dvodimenzionalne elektronske štampane ploče (slika 72). Površinu posmatramo kao kvadratno polje sa kvadratnom rešetkom. Posmatrajmo ovo kao problem pretraživanja u kojem postoji samo jedan operator: naneti metal na mrežnu ćeliju [X,Y], gde su X i Y Dekartove koordinate. U početku, nijedna ćelija nije popunjena metalom. U ciljnom stanju, električna veza postoji između svakog para u zadatoj listi parova ćelija. Električna veza znači neprekidan niz susednih ćelija ispunjenih metalom koji povezuje par ćelija, a da pri tom nema vezu sa bilo kojim drugim ćelijskim parom; na primer, cilj može da bude da se povežu parovi [10,20] sa [25,29] i [3,9] sa [44,18] ali ne i [10,20] sa [3,9]. Potrebno je pronaći put uspostavljanjem ciljne veze sa najmanjim utroškom metala.

a) Odrediti prostor pretraživanja.

b) Da li je dvosmerno pretraživanje dobra ideja?

c) Kako se menja faktor grananja u toku direktnog pretraživanja?

d) Naći heuristiku koja ograničava direktno pretraživanje.



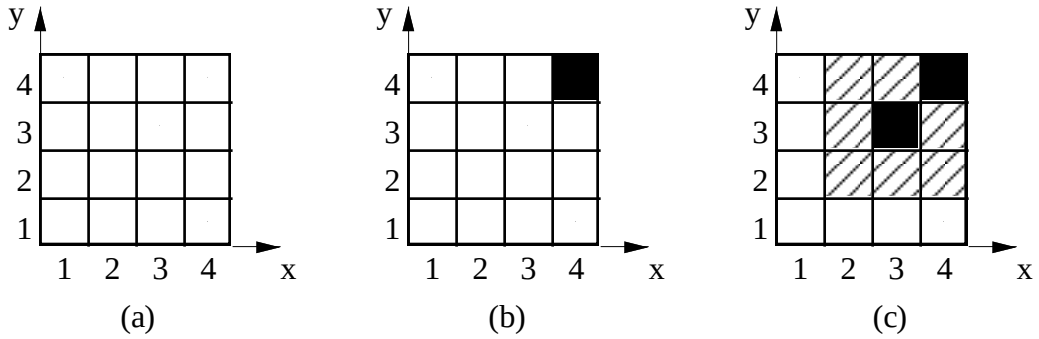
Slika 72

Rešenje

a) Prostor pretraživanja je skup svih mogućih izgleda površine štampanog kola. S obzirom da svaka ćelija može biti u jednom od dva stanja, i da ima N^2 ćelija gde je N dimenzija kvadratnog polja, postoji ukupno 2^{N^2} različitih rasporeda metala.

b) Dvosmerno pretraživanje nije moguće sprovesti, kao ni pretraživanje unazad jer postoji mnoštvo rasporeda metala koji zadovoljavaju ciljni uslov i ne postoji lak način da se ovi rasporedi pronađu. Kada bi bio poznat neki od ovih rasporeda, ne bi bilo potrebe za pretragom. U ovom slučaju cilj je upravo nalaženje tačnog opisa ciljnog stanja, za razliku od problema gde nam je opis ciljnog stanja poznat, a tražimo niz operatora koji iz startnog stanja prevode u ciljno stanje. Samo ova druga vrsta problema omogućava pretragu unazad.

c) Faktor grananja se smanjuje u svakom koraku. Svaka primena operatora uključuje prekrivanje jedne ćelije metalom, tako da se smanjuje broj ćelija koje treba razmatrati. Osim ćelije koja je upravo metalizovana, još neke ćelije mogu pri tom ispasti iz daljeg razmatranja - to su ćelije koje moraju ostati nepokrivene metalom jer bi u suprotnom došlo do formiranja pogrešnih električnih veza. Prema tome, faktor grananja može se smanjiti i za više od jedan po nivou stabla pretrage. Posmatrajmo, na primer, situaciju na slici 73 gde je potrebno povezati ćeliju [3,3] sa ćelijom [4,4], kao i ćeliju [1,4] sa ćelijom [4,1]. U početnom stanju (sve ćelije nepopunjene, slika 73*a*), faktor grananja je 16, jer je za popunjavanje moguće izabrati bilo koju ćeliju. Pretpostavimo da se u prvo koraku pretrage popuni ćelija [4,4] (slika 73*b*). Faktor grananja za drugi korak je 15, jer se može izabrati bilo koja od nepopunjenih ćelija. Neka je u drugom koraku izabrana ćelija [3,3]. Ovim je formirana veza ćelija [4,4] i [3,3] pa, s obzirom da ove ćelije ne učestvuju ni u jednoj drugoj vezi, ne sme se popunjavati nijedna od njima susednih ćelija (na slici 73*c* ove ćelije su osenčene). Prema tome, u trećem koraku, faktor grananja jednak je broju nepopunjenih ćelija koje su dozvoljene za popunjavanje, odnosno 7.



Slika 73

d) Mnoge heuristike se mogu osmisliti, na primer:

- Prioritet dati ćelijama koje su susedne poslednjoj pokrivenoj ćeliji (takozvano fokusiranje pažnje)
- Prioritet dati ćelijama koje leže na pravoj liniji između tačaka koje je potrebno spojiti.
- Čim se uspostavi električna veza između dve zadate tačke, ne metalizovati nijednu ćeliju koja je unutar određenog rastojanja (na primer 5 ćelija) od ovih tačaka.
- Prioritet dati ćelijama koje nastavljaju pravu liniju (na primer, metalizovati tačku [X,Y] ako su i [X-1,Y] i [X-2,Y] metalizovane).

Zadatak 27: Kombinovanje hemijskih jedinjenja

Kombinovanje hemijskih elemenata i jedinjenja da bi se dobila nova jedinjenja je neka vrsta pretraživanja. Poznate su sledeće reakcije:

1) AKO kombinujemo jedan mol Cl_2 i jedan mol H_2O
ONDA dobija se jedan mol HClO i jedan mol HCl

2) AKO kombinujemo jedan mol $CaCO_3$ sa dva mola HCl
ONDA dobija se jedan mol $CaCl_2$, jedan mol CO_2 i jedan mol H_2O

3) AKO kombinujemo dva mola H_2O_2 i jedan mol MnO_2
ONDA dobija se dva mola H_2O, jedan mol O_2 i jedan mol MnO_2

(dakle, ovo je katalizator)

4) AKO kombinujemo jedan mol H_2 i jedan mol Cl_2
ONDA dobija se dva mola HCl

5) AKO kombinujemo četiri mola HCl i jedan mol MnO_2
ONDA dobija se jedan mol $MnCl_2$, dva mola H_2O i jedan mol Cl_2.

a) Polazeći od startnog stanja koje se sastoji od dva mola Cl_2, jednog mola MnO_2, jednog mola $CaCO_3$ i dva mola H_2O_2, naći rešenje za dobijanje jednog mola $CaCl_2$ primenom pretraživanja po dubini.

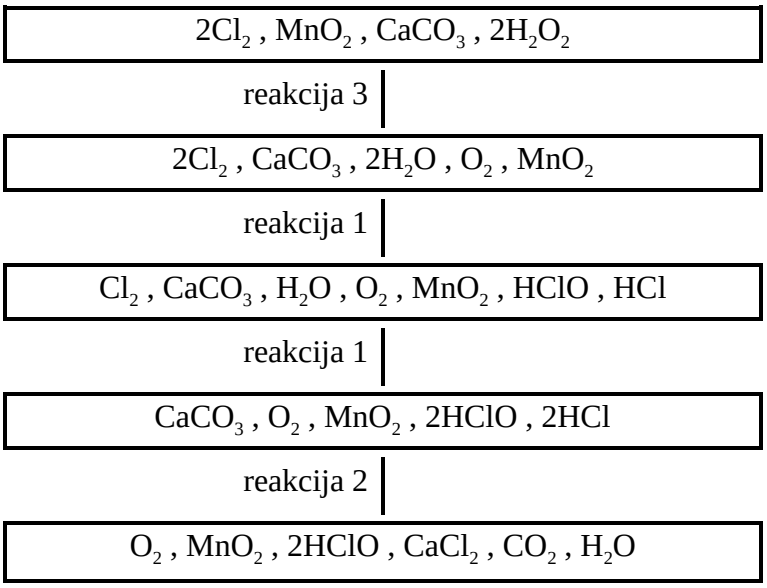
b) Formulisati heurističku funkciju i funkciju cene pogodnu za ovakve probleme.

Rešenje

a) Stanje pretrage će predstavljati spisak svih hemikalija raspoloživih u određenom trenutku (bilo da su inicijalno zadate ili proizvod reakcija) sa naznačenim količinama. Svaka od pet navedenih hemijskih reakcija definiše po jedan operator promene stanja. Preduslov za primenu određenog operatora je da se hemikalije potrebne za reakciju nalaze u tekućem stanju u količinama većim ili jednakim onim navedenim u formuli. Početno stanje opisano je u postavci. Ciljno stanje je bilo koje stanje koje poseduje bar 1 mol $CaCl_2$. Stablo pretrage za ovaj slučaj prikazano je na slici 74 i ne zavisi od algoritma pretrage, jer je u svakom od stanja primenljiv tačno jedan od operatora.

Treba primetiti da se veći faktor grananja mogao ostvariti da smo reakcije izvodili sa proporcionalno manjim količinama hemikalija. Tako bi se, na primer, u trećem stanju pretrage za reakciju po pravilu 5 mogao uzeti samo 1 mol HCl i 1/4 mola MnO_2 , što bi proizvelo novo stanje različito od onih prikazanih na slici 74. Međutim, uzimanje manjih količina hemikalija ne garantuje da bismo uvek dobili traženo jedinjenje u potrebnoj količini.

b) U opštem slučaju kod problema sinteze hemijskih jedinjenja, funkciju cene nekog operatora može predstavljati (tržišna) cena hemikalija potrebnih za određenu reakciju. Heuristička funkcija može biti broj neupotrebljenih hemikalija za koje se pretpostavlja da će biti upotrebljene, pomnoženo prosečnom cenom tih hemikalija. Ukoliko nam je potrebna heuristička funkcija koja daje lokalno najbolji sledbenik tekućeg čvora, operatore možemo poređati po prioritetu koji određujemo na osnovu toga šta je rezultat odgovarajućih hemijskih reakcija.



Slika 74

Alternativno se heuristička funkcija može bazirati na proceni opasnosti izvođenja pojedinih hemijskih reakcija, ako je ovaj faktor od značaja.

Usvojeni model pretraživanja pri kome se u svakom koraku bira jedna od reakcija za koje postoje potrebne komponente u dovoljnim količinama u tekućem stanju odgovara sledećoj situaciji: Svaka od hemikalija drži se odvojeno od svih drugih, za određenu reakciju mešaju se samo hemikalije potrebne za tu reakciju u željenim količinama a proizvodi reakcije se odvajaju jedni od drugih. U praksi su hemikalije često pomešane u jednoj reakcionoj posudi tako da se hemijske reakcije odvijaju simultano različitim brzinama i uspostavlja se neko ravnotežno stanje. Za takav slučaj morala bi se raditi složena analiza (to jest, simulacija istovremenih reakcija) za određivanje toga stanja. Operator promene stanja tada bi bilo dodavanje nove hemikalije u reakcionu posudu što bi dovelo do novog ravnotežnog stanja.

Zadatak 28: Simbolička integracija

Posmatrajmo problem simboličke integracije: Potrebno je izračunati zadati neodređeni integral nalaženjem primitivne funkcije za podintegralnu funkciju, to jest, za zadatu f(x) potrebno je naći funkciju F(x) tako da važi

$$F(x) + c = \int f(x)dx .$$

Funkcija *F(x)* naziva se u matematici *primitivnom funkcijom* funkcije *f(x)*.

a) Šta je prostor pretraživanja?

b) Kako izgleda startno stanje?

c) Kako izgleda ciljno stanje?

d) Koji su operatori promene stanja?

e) Da li se problem može razložiti na nekom međustanju?

Rešenje

a) Problem je generalno definisan u smislu da zadata funkcija f(x) može biti proizvoljna matematička funkcija jedne promenljive. U realizaciji automatske simboličke integracije nužno se nameće ograničenje da je funkcija f(x) zadata izrazom koji je sastavljen od uobičajenih algebarskih operacija i elementarnih matematičkih funkcija. U tom slučaju, prostor pretraživanja predstavlja skup svih mogućih izraza sastavljenih od ovih elementarnih funkcija. Pošto nema ograničenja u složenosti izraza, prostor pretraživanja je beskonačno veliki.

b) Startno stanje predstavljeno je zadatom funkcijom f(x) čiji neodređeni integral treba naći. U narednoj diskusiji simbolički ćemo stanja označavati neodređenim integralima funkcija, da bi se naglasilo da se rešava problem integracije.

c) Ciljno stanje je predstavljeno matematičkom funkcijom čija se primitivna funkcija trivijalno nalazi, to jest poznata je iz tablica (takozvani tablični integrali). Na primer,

1. $\int x^n dx = \frac{x^{n+1}}{n+1} + C, n \neq -1$
2. $\int \frac{dx}{x} = \ln|x| + C$
3. $\int \sin x dx = -\cos x + C$
4. $\int \cos x dx = \sin x + C$
5. $\int e^x dx = e^x + C$ i tako dalje..

i tako dalje

d) Problem se (u slučaju da rešenje postoji) može rešiti sistematskim transformisanjem podintegralne funkcije po određenim matematičkim pravilima dok se problem ne svede na jedan ili više tabličnih integrala. Pravila transformacije podintegralne funkcije igraju u ovom slučaju ulogu operatora promene stanja. Neka od pravila su:

1. Pravilo dekompozicije: $\int [f_1(x) \pm f_2(x)]dx = \int f_1(x)dx \pm \int f_2(x)dx$

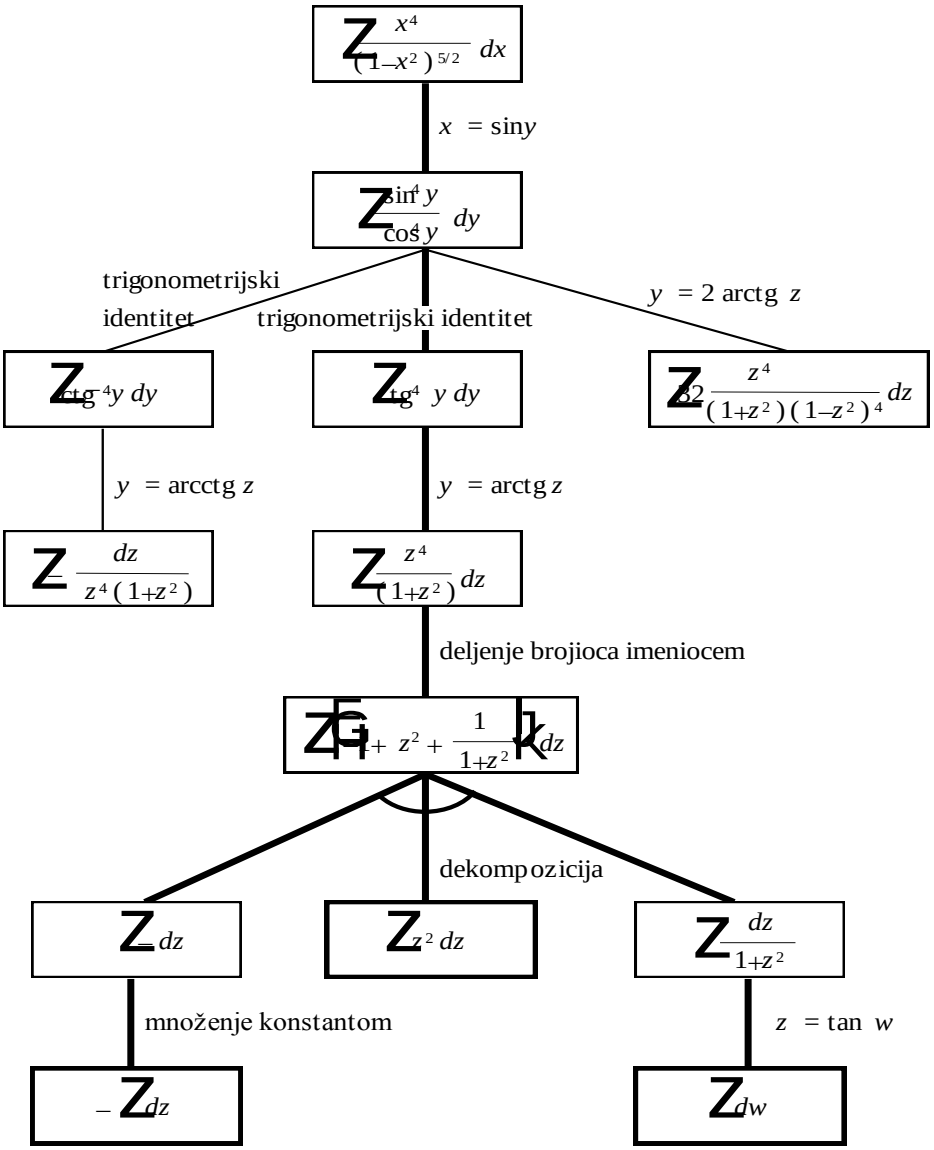
2. Pravilo množenja konstantom: $\int Af(x)dx = A\int f(x)dx, A \neq 0$ je proizvoljna konstanta.

3. Zamena promenljive: $\int f(x)dx = \int f[\varphi(t)]\varphi'(t)dt$ pri čemu je $x = \varphi(t)$ gde diferencijal promenljive x zamenjujemo diferencijalom funkcije promenljivom $\varphi(t)$ i pri tome se funkcija φ bira tako da nova podintegralna funkcija bude pogodnija od stare. Očigledno sistem koji primenjuje ovo pravilo mora imati realizovano i simboličko diferenciranje. Simboličko diferenciranje nije problem pretraživanja jer se za zadatu funkciju njen diferencijal jednoznačno određuje precizno definisanim pravilima.

4. Parcijalna integracija: $\int udv = uv - \int vdu$.

e) Pravilo dekompozicije omogućava nam dekompoziciju problema jer se problem nalaženja jednog (složenijeg) integrala svodi na nezavisno nalaženje više (prostijih) integrala. Proces pretraživanja se može sada izraziti koristeći AND/OR stablo. Na slici 75 prikazano je AND/OR stablo pretraga za izračunavanje integrala

$$\int \frac{x^4}{(1-x^2)^{5/2}} dx$$



Slika 75

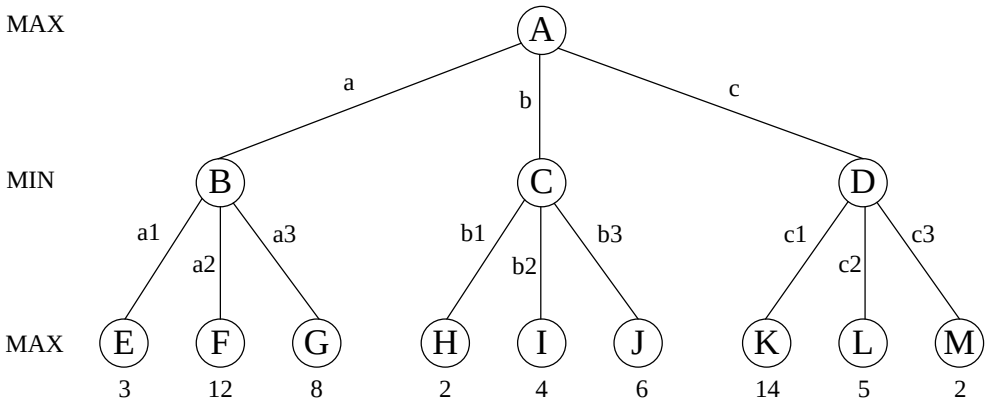
Ciljna stanja predstavljaju tablične integrale, tako da se na osnovu njih, kada se eliminišu promenljive uvedene u toku rešavanja problema dobija rešenje:

$$\int \frac{x^4}{(1-x^2)^{5/2}}dx = -\operatorname{tg}(\arcsin x) + \frac{1}{3}\operatorname{tg}^3(\arcsin x) + \arcsin x$$

1.3. Pretraživanje u Igrama

Zadatak 29: Minimax Metoda

Upotrebom minimax algoritma, za dato stablo igre (slika 76), pronaći naredni potez koji će biti odigran.



Slika 76

Rešenje

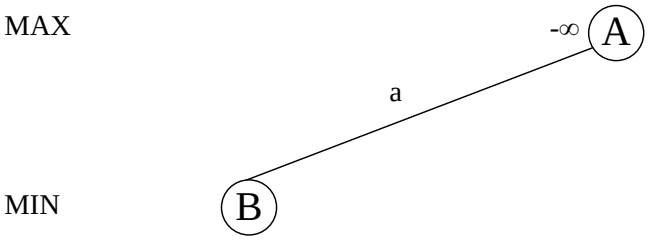
Svaka potezna igra se može predstaviti *stablom igre*. Svaki čvor stabla predstavlja jednu *poziciju* u igri a grane moguće poteze. Potezi preslikavaju jednu poziciju na tabli u drugu. Svakom igraču odgovara neki nivo stabla. Naime, ukoliko je prvom igraču odgovarao nivo N, onda će drugom igraču odgovarati nivo N+1.

Neke od pozicija nemaju moguće poteze. Ove pozicije nazivaju se *terminalne pozicije*. U ovim pozicijama svaki igrač dobija određeni rezultat (na primer, za iks-oks igru, terminalna stanja su ona u kojima su sva polja popunjena ili je jedan od igrača spojio tri simbola).

Broj grana od svakog čvora u stablu jednak je broju mogućih poteza u tom stanju i naziva se *faktor grananja*. Faktor grananja dobar je indikator koliko će sama igra biti komplikovana kompjuteru za igranje.

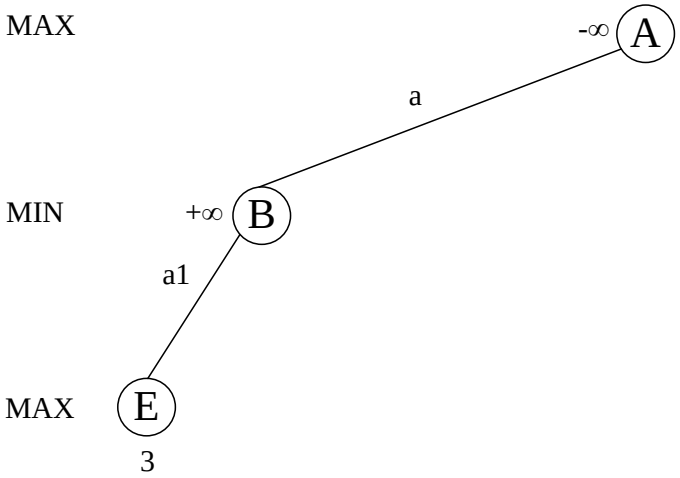
Kako bismo pronašli naredni potez koji će odigrati MAX, za stablo iz postavke zadatka, sprovešćemo, *minimax algoritam* opisan u dodatku 1 (algoritam 7). Za trenutno stanje u algoritam prosleđuje se koreni čvor stabla. Maksimalnu dubina stabla neće biti ograničena.

S obzirom da trenutni, koreni, čvor stabla nije terminalni njemu se dodeljuje beskonačna vrednost. Sledeće je potrebno razmotriti sve moguće operacije (*a*, *b* i *c*) iz trenutnog čvora i sačuvati najveću dobijenu vrednost za igrača (jer je u pitanju MAX igrač). Prvo se razmatra operacija a i pronalazi stanje do koga dovodi. U pitanju je stanje B. Za novodobijeno stanje rekurzivno se vrši poziv minimax algoritma. Trenutni izgled stabla prikazan je na slici 77.



Slika 77

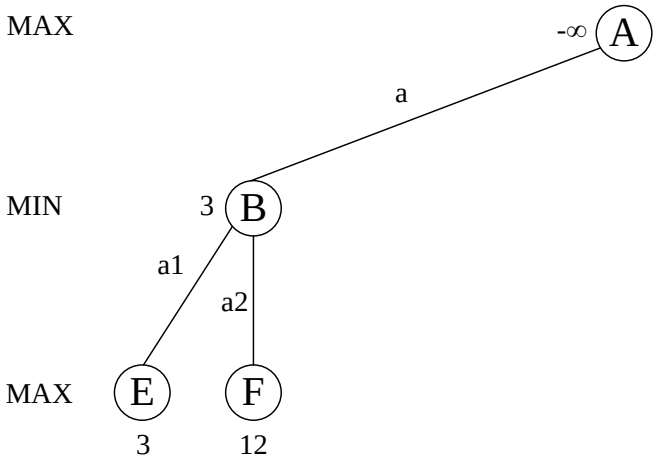
Kako čvor B takođe nije terminalni čvor potrebno je odrediti njegovu najbolju vrednost što postižemo razmatranjem stanja u koja možemo preći pod dejstvom jednog od operatora: *a1*, *a2*, *a3*, i izborom najmanje vrednost (primetiti da čvor B pripada nivou kada je MIN na potezu). Početna najbolja vrednost čvora B je pozitivna beskonačnost. Prva od operacija čvora B, a1, dovodi do prelaska u čvor E nad kojim vršimo novi rekurzivni poziv minmax algoritma. Čvor E jeste terminalni čvor i kao rezultat poslednjeg poziva minimax algoritma vraća se vrednost statičke funkcije procene čvora E. Vrednosti statičkih funkcija date su na slici u postavci zadatka, i za čvor E vrednost ove funkcije je 3. Izgled stabla igre dat je na slici 78.



Slika 78

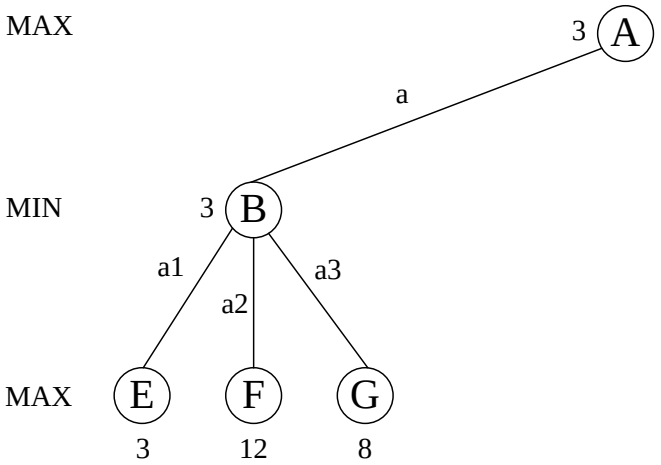
Nakon prvog povratak iz minimax funkcije čvor B dobija novu najbolju vrednost (min{+∞, 3}).

Kako postoje operacije koje nisu razmotrene nad čvorom B, bira se jedna od preostalih, najpre *a2*. Izabrana operacija vodi ka čvoru F, nad kojim se poziva algoritam. Kako je u pitanju terminalni čvor funkcija minimax se završava, a povratna vrednost je vrednost 12. Vrednost čvora B neće biti promenjena jer je vrednost dobijena preko čvora F veća od trenutno najbolje vrednosti, dobijene preko čvora E. Trenutni izgled stabla prikazan je na slici 79.



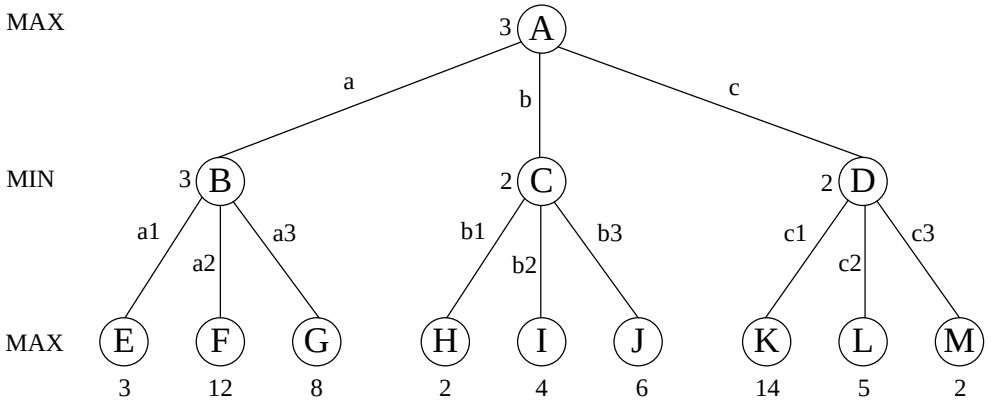
Slika 79

Preostala je još jedna operacija koju je potrebno razmotriti nad čvorom B. U pitanju je operacija *a3*, koja vodi u čvor G. Čvor G je terminalni čvor za koji statička funkcija procene vraća vrednost 8, kako je naznačeno u postavci zadatka. Trenutna vrednost čvora B bolja je od trenutno dobijene preko čvora G. To je ujedno i konačna vrednost čvora B koja predstavlja povratnu vrednost funkcije minmax. Ovog trenutka potrebno je ažurirati vrednost čvora A na 3 (max{-∞, 3}). Trenutni izgled stabla igre dat je na slici 80.



Slika 80

Nadalje je potrebno razmotriti ostale operacije, b i c, čvora A na analogni način. Ove operacije dovode do čvorova C i D, preko kojih se dobija vrednost 2. Konačni izgled stabla igre prikazan je na slici 81. Kako je trenutna vrednost čvora A, 3, veća od vrednosti dobijene preko ovih čvorova, neće doći do njene promene.

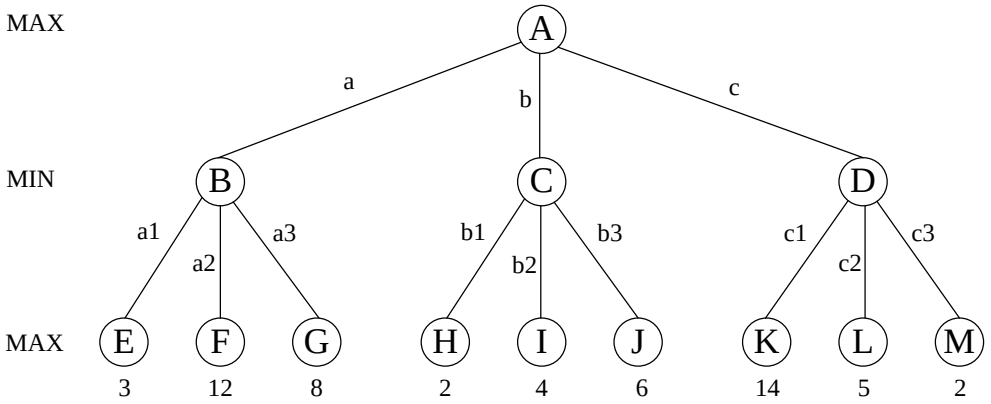


Slika 81

Konačna vrednost korenog čvora upućuje na potez koji je optimalan za igrača (dovodi ga u najbolju moguću poziciju). Ovaj potez naziva se i *minimax odluka*. Kako je vrednost korenog čvora u primeru 3, koja je dobijena preko grane *a*, to je ujedno i optimalni potez za MAX igrača u trenutnom stanju igre. Ukoliko pretpostavimo da MIN takodje vrši optimalan izbor poteza, on bi u sledećem koraku izabrao potez a1.

Zadatak 30: Alfa-Beta Odsecanje

Upotrebom minimax algoritma, uz primenu alfa-beta odsecanja, za dato stablo igre, na slici 82, pronaći naredni potez koji će biti odigran. Naznačiti koji čvorovi stabla neće biti obiđeni.



Slika 82

Rešenje

Problem minimax algoritma je eksponencijalni rast broja čvorova sa dubinom stabla. Metode za eliminisanje eksponencijalne zavisnosti još uvek nisu pronađene ali je otkrivena metoda koja u najboljem slučaju može prepoloviti broj čvorova koje je potrebno obići. Tehnika koja sprovodi ovu optimizaciju u literaturi je poznata pod nazivom *alfa-beta odsecanje*.

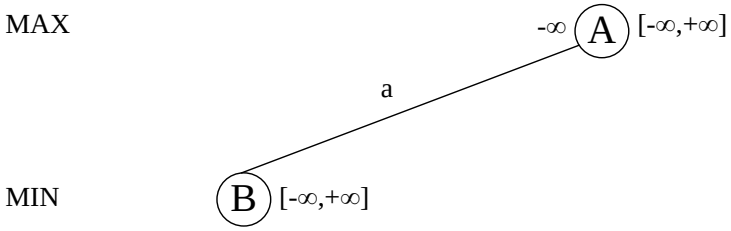
Prilikom *alfa odsecanja* potrebno je voditi evidenciju o najboljem potezu koji može biti odigran. Alfa granica je donja granica rezultata koja može biti postignuta. Pretragom se može pronaći bolja sekvenca poteza ali se lošija sekvenca neće prihvatiti. Vođenjem evidencije o

alfa vrednosti izbegava se razmatranje svih poteza u kojima bi protivnik imao mogućnost boljeg poteza.

Sa druge strane *beta odsecanje* vodi evidenciju o gornjoj granici rezultata koja se može postići. Ova vrednost se ažurira u trenucima kada nas protivnik natera na neku sekvencu poteza. Tada je poznata gornja granica koja se može postići pri čemu dalji potezi protivnika mogu ovu granicu dodatno suziti. Ukoliko se pronađe sekvenca poteza sa procenom većom od beta granice, taj deo se može izuzeti iz razmatranja jer protivnik neće dozvoliti te poteze.

Kako bismo pronašli traženi potez koji će odigrati MAX, za stablo iz postavke zadatka, sprovešćemo, *minimax algoritam sa alfa-beta* odsecanjem opisan u dodatku 1 (algoritam 8). Za trenutno stanje u algoritam prosleđuje se koreni čvor stabla. Maksimalnu dubina stabla neće biti ograničena, a vrednosti za alfa i beta su -∞ odnosno +∞.

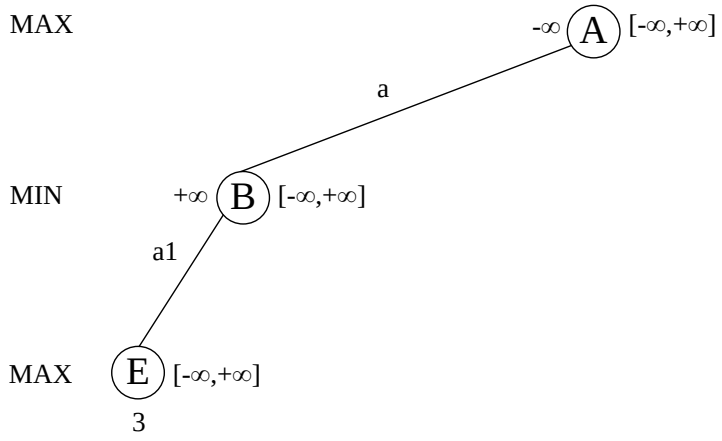
S obzirom da trenutni, koreni, čvor stabla nije terminalni njemu se dodeljuje beskonačna vrednost. Sledeće je potrebno razmotriti sve moguće operacije (*a, b* i *c*) iz trenutnog čvora i sačuvati najveću dobijenu vrednost za igrača (jer je u pitanju MAX igrač). Prvo se razmatra operacija *a* i pronalazi stanje do koga dovodi. U pitanju je stanje B. Za novodobijeno stanje rekurzivno se vrši poziv minimax algoritma. Trenutni izgled stabla prikazan je na slici 83.



Slika 83

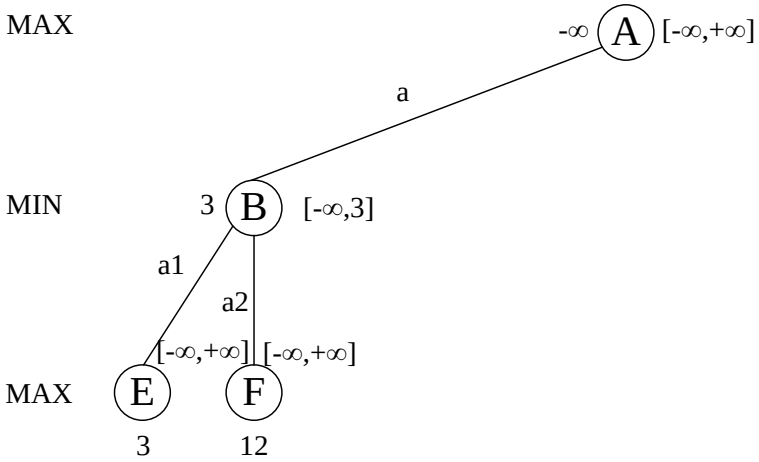
Kako čvor B takođe nije terminalni čvor potrebno je odrediti njegovu najbolju vrednost što postižemo razmatranjem stanja u koja možemo preći pod dejstvom jednog od operatora: *a1*, *a2*, *a3*, i izborom najmanje vrednost (primetiti da čvor B pripada nivou kada je MIN na potezu). Početna najbolja vrednost čvora B je pozitivna beskonačnost. Prva od operacija čvora B, *a1*, dovodi do prelaska u čvor E nad kojim vršimo novi rekurzivni poziv minmax algoritma. Čvor E jeste terminalni čvor i kao rezultat poslednjeg poziva minimax algoritma vraća se vrednost statičke funkcije procene čvora E. Vrednosti statičkih funkcija date su na slici u postavci zadatka, i za čvor E vrednost ove funkcije je 3. Izgled stabla dat je na slici 84.

Nakon prvog povratak iz minimax funkcije čvor B dobija novu najbolju vrednost (min{+∞, 3}). Takođe vrši se ažuriranje beta vrednosti u okviru čvora B koja sada iznosi takođe 3 (min{+∞, 3}).



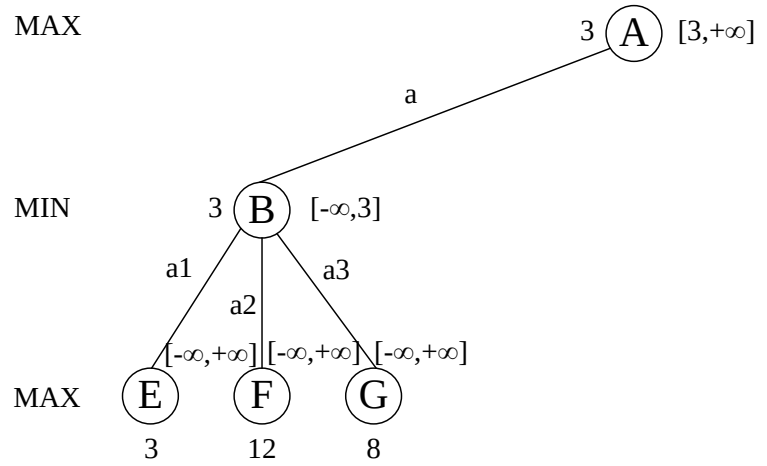
Slika 84

Kako postoje operacije koje nisu razmotrene nad čvorom B, bira se jedna od preostalih, najpre *a2*. Izabrana operacija vodi ka čvoru F, nad kojim se poziva algoritam. Kako je u pitanju terminalni čvor funkcija minimax se završava, a povratna vrednost je vrednost 12. Vrednost čvora B neće biti zamenjena jer je vrednost dobijena preko čvora F veća od trenutno najbolje vrednosti, dobijene preko čvora E. Takođe neće doći do promene beta vrednosti. Trenutni izgled stabla prikazan je na slici 85.



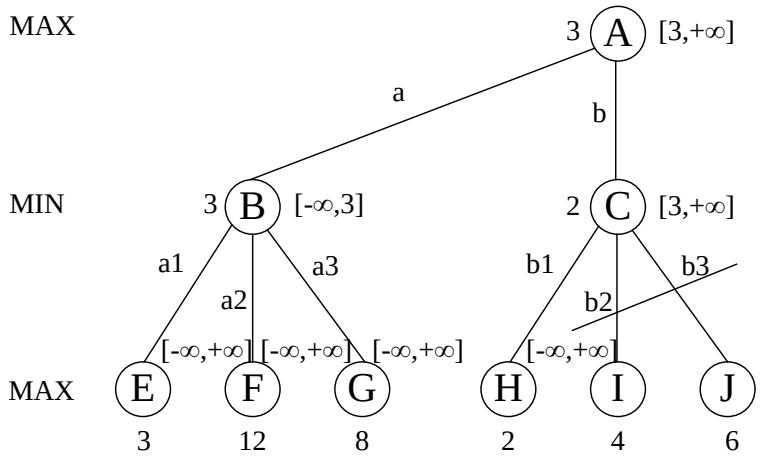
Slika 85

Preostala je još jedna operacija koju je potrebno razmotriti nad čvorom B. U pitanju je operacija *a3*, koja vodi u čvor G. Čvor G je terminalni čvor za koji statička funkcija procene vraća vrednost 8, kako je naznačeno u postavci zadatka. Trenutna vrednost čvora B bolja je od trenutno dobijene preko čvora G. To je ujedno i konačna vrednost čvora B koja predstavlja povratnu vrednost funkcije minmax. Ovog trenutka potrebno je ažurirati vrednost čvora A na 3 (max{-∞, 3}), kao i vrednost alfa na 3 (max{-∞,3}). Trenutni izgled stabla igre dat je na slici 86.



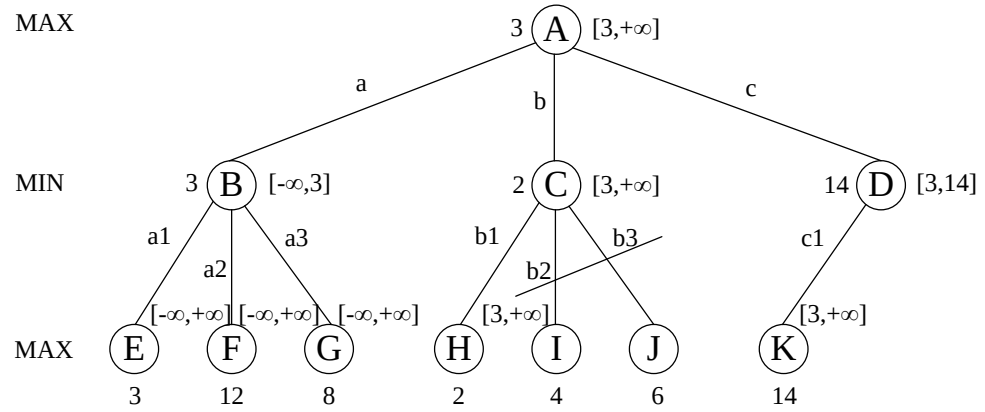
Slika 86

Nadalje je potrebno razmotriti ostale operacije, *b* i *c*, čvora A na analogni način. Operacija *b* vodi ka čvoru C za koji je potrebno odrediti najbolju vrednost. Treba razmotriti sve operacije koje je moguće izvršiti nad ovim čvorom. U pitanju su operacije *b1*, *b2* i *b3*. Operacija *b1* vodi ka čvoru H koji je terminalni čvor i čija je vrednost statičke funkcije procene jednaka 2. Ovo je ujedno i najbolja vrednost čvora C. Ovog trenutka vrši se provera da li je najbolja vrednost manja ili jednaka alfa vrednosti, što je u ovom slučaju ispunjeno i vrši se odsecanje dela stabla. Drugim rečima operacije *b2* i *b3* neće biti razmatrane i trenutno najbolja vrednost čini povratnu vrednost funkcije. Alfa vrednost A čvora se neće promeniti. Trenutni izgled stabla igre prikazano je na slici 87. Iako naznačeni na slici čvorovi I i J neće biti obiđeni.



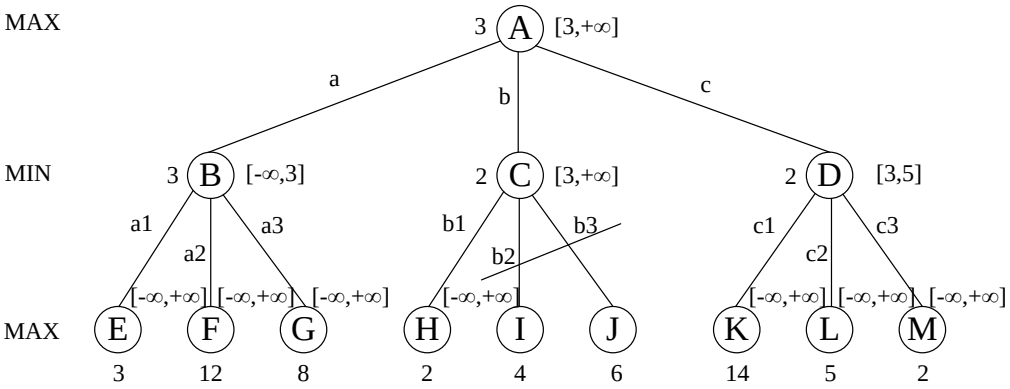
Slika 87

Na isti način potrebno je razmotriti preostalu operaciju čvora A, *c*. Operacija *c* dovodi do čvora D nad kojim je moguće izvršiti tri operacije *c1*, *c2* i *c3*. Operacija *c1* vodi ka čvoru L koji je terminalni i čija je vrednost statičke funkcije procene 14. To će ujedno biti i najbolja trenutna vrednost čvora D kao i nova vrednost beta parametra. Slika 88 Prikazuje ovaj trenutak.



Slika 88

Čvor L daje novu najbolju vrednost, 5, za čvor D ujedno postavljajući istu vrednost za parametar beta, čvora D. Povratna vrednost čvora M izlazi van alfa-beta granica u kom trenutku se trenutno najbolja vrednost čvora D propagira ka čvoru A. Konačni izgled granicu za beta na vrednost 5 dok čvor M izlazi van granice. Stoga se konačna vrednost čvora D jednaka 2. Konačni izgled stabla prikazan je na slici 89. Lako se zaključuje da je konačna minimax vrednost čvora A jednaka 3.

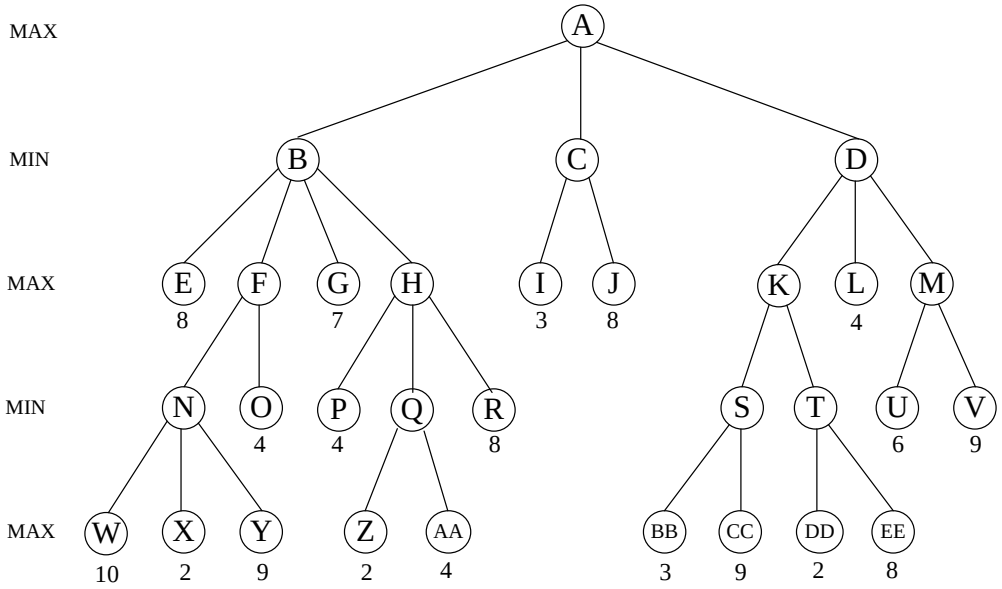


Slika 89

Minimax odluka u ovom zadatku svakako je jednaka odluci u prethodnom zadatku. Uloga alfa-beta odsecanja jeste optimizacija pretrage. Čvorovi I i J nisu obiđeni u ovom zadatku.

Zadatak 31: Online Partner

Igrate mice protiv online partnera na Facebook-u. Igra je skoro završena u trenutku kada vaš partner zahteva pauzu kako bi otišao do toaleta. Vi se slažete, ali dok čekate odlučujete da skicirate stablo igre preostalih poteza. Stablo koje ste skicirali prikazano je na slici 90.



Slika 90

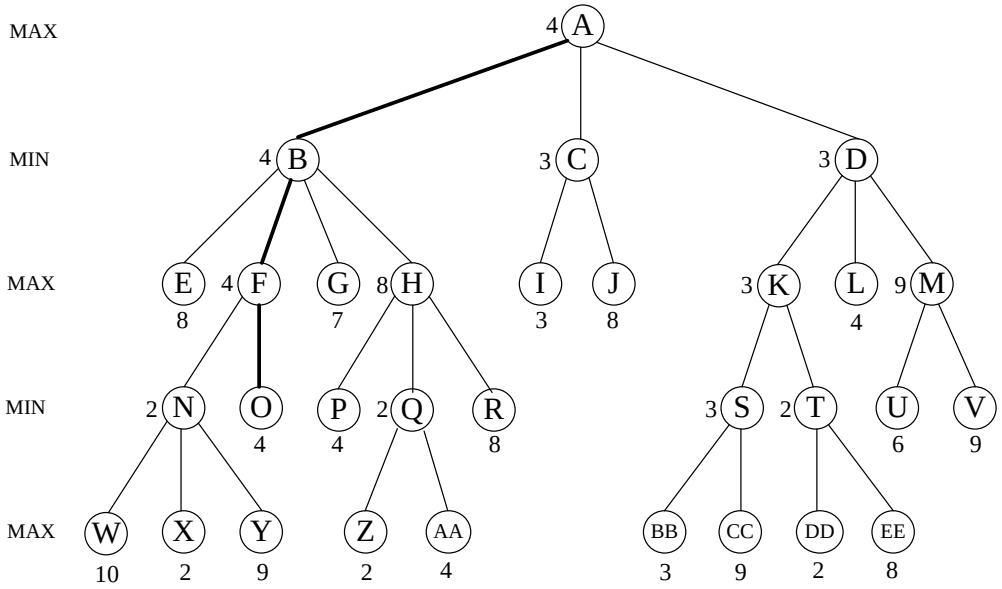
a) Nakon posmatranja stabla pretrage, odlučujete da pronađete koji je najbolji potez. Upotrebljavate minimax metod za određivanje optimalne putanje kroz stablo pretrage. Takođe, prikazujete minimax vrednost za svaki čvor na dijagramu, a zatim čvorove koji se nalaze na najboljoj putanji.

b) Vaš partner se i dalje nije vratio iz toaleta, stoga vi nastavljate da posmatrate stablo pretrage i razmišljate koji čvorovi bi mogli biti isečeni primenom alfa-beta odsecanja. Možete pretpostaviti da ste ranije pretražili levi deo stabla i pronašli da je minimax vrednost za čvor B jednaka 4. Nastavljate sa procesiranjem stabla na desno, i zaokružujete sve čvorove čija će statička vrednost biti izračunata ukoliko se primeni minimax algoritam sa alfa-beta odsecanjem. Takodje, precrtavate sve čvorove za koje vrednost neće biti izračunata.

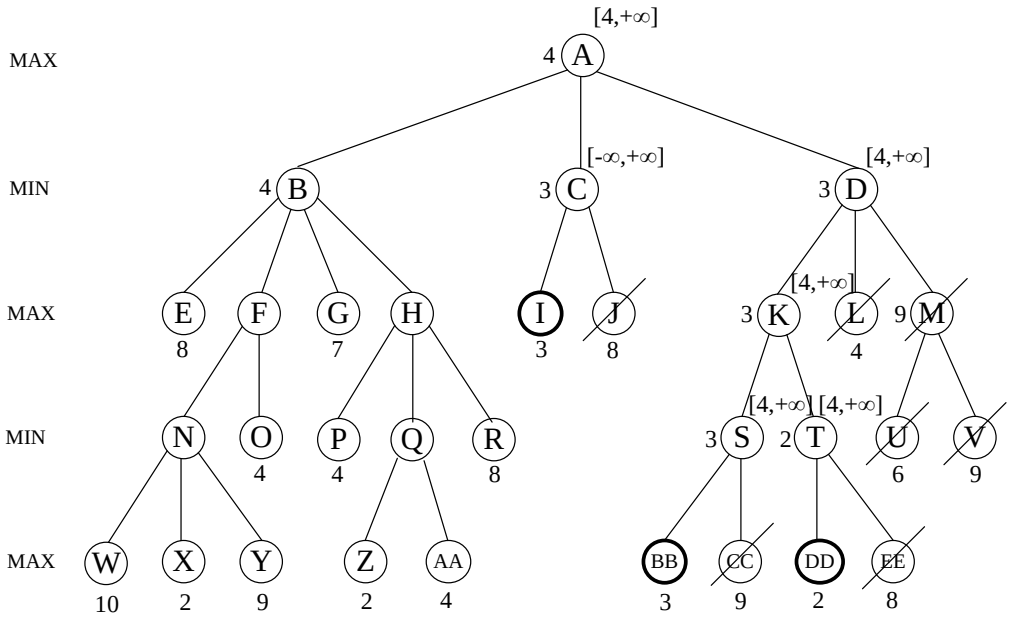
Rešenje

a) Primenom minimax algoritma dobijene su minimax vrednosti označene na slici 91. Optimalna putanja takođe je naznačena na slici i prolazi kroz čvorove A, B, F i O.

b) Primenom minimax algoritma sa alfa-beta odsecanjem dobijeno je stablo igre prikazano na slici 92. Precrtani čvorovi su ne obiđeni čvorovi stabla, dok su čvorovi za koje je izračunata statička vrednost funkcije označeni podebljano.



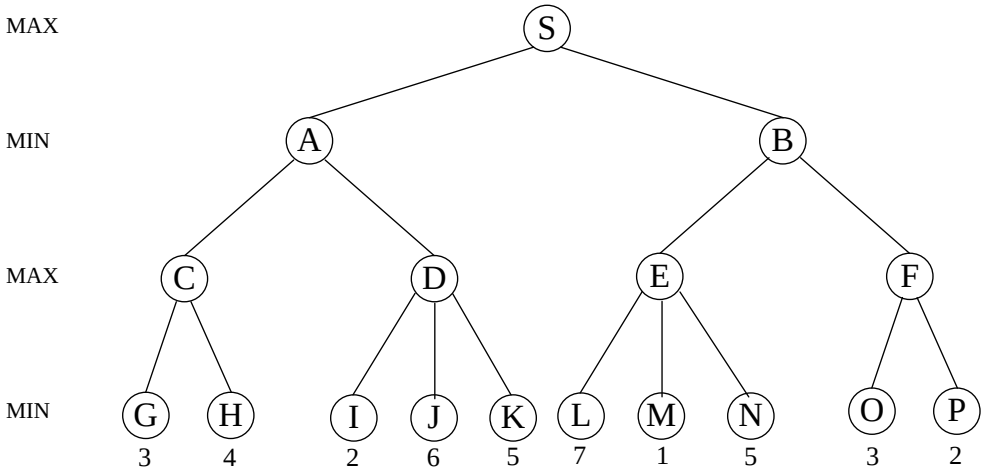
Slika 91



Slika 92

Zadatak 32: Progresivno Produbljivanje

Razmotrimo stablo igre prikazano na slici 93. Svaki čvor stabla obeležen je jednim slovom (slova od A do P) i za svaki list je naznačena statička vrednost. Uobičajeno, što je statička vrednost veća, to je bolja situacija iz perspektive MAX igrača.



Slika 93

a) Sprovesti minimax pretraživanje stabla, pretpostaviti da koreni čvor pokušava da maksimizuje uspeh. U kom listu će MAX završiti? Koliko statičkih vrednosti se izračunava prilikom pretraživanja?

b) Ukoliko se upotrebljava metoda progresivnog produbljivanja, koliko statičkih vrednosti će biti izračunato?

c) Sprovesti minimax pretraživanje uz primenu alfa-beta odsecanja. Koji od čvorova, ukoliko takvi postoje, će biti odsečeni?

d) Pretpostaviti da je moguće izvršiti razmeštanje grana koje izlaze iz istog čvora na dubini 2 (C, D, E i F). Na primer može se izvršiti zamena pozicija čvorova G i H. Ukoliko razmotrimo sva moguća razmeštanja čvorova, koji je najmanji broj izračunavanja statičkih vrednosti potrebno prilikom alfa-beta pretrage?

Rešenje

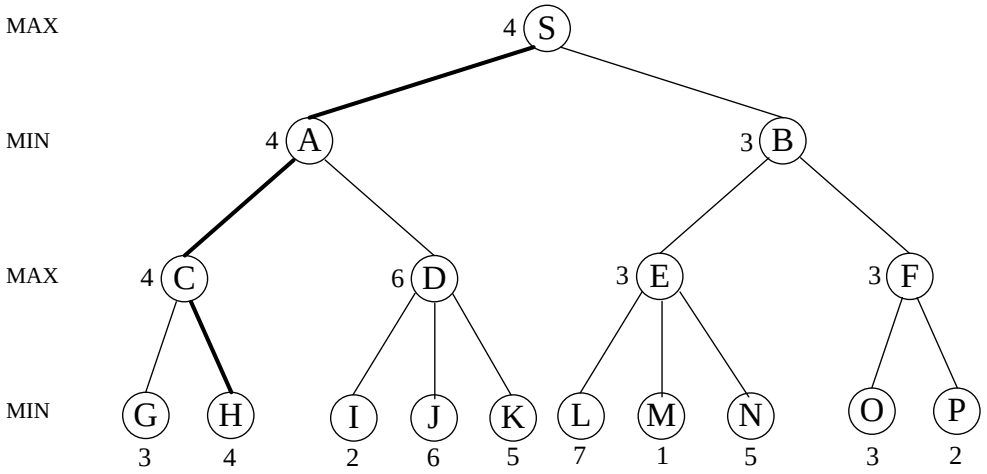
Na turnirima, od igrača se očekuje da naprave očekivani broj poteza u zadatom vremenu, ograničenom od strane nemilosrdnog sata.

U ovim prilikama česta je primena metoda *progresivnog produbljivanja* (engl. progressive deepening) koji analizira svaku situaciju za dubinu 1, zatim za dubinu 2, zatim za dubinu 3, itd. sve dok se ne iskoristi vreme određeno za jedan potez. Na ovaj način, uvek postoji potez spreman da se odigra. Izbor je zasnovan na analizi koja je sprovedena na nivou za jedan manjem od nivoa na kome je algoritam prekinut, usled isteka raspoloživog vremena.

a) Primenom minimax algoritma dobija se stablo prikazano na slici 94.

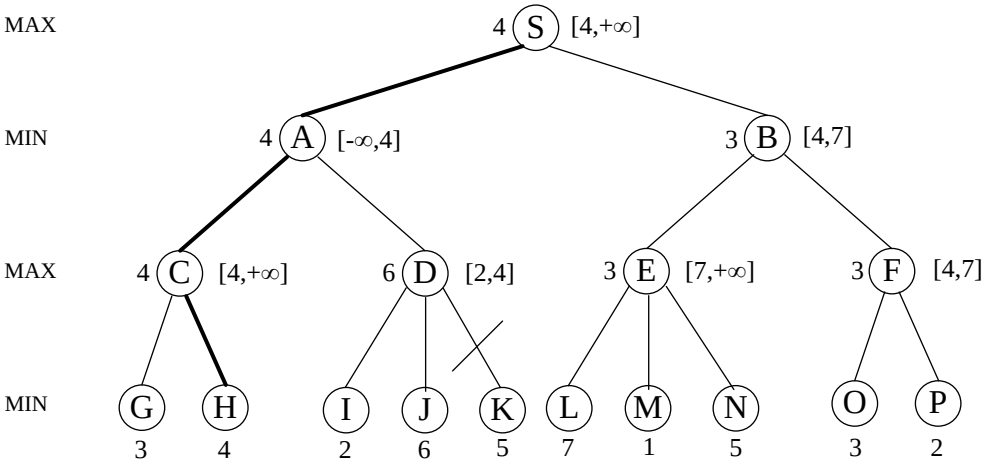
Optimalna putanja za MAX igrača, naznačena na slici, je S, A, C, H. Broj listova stabla ujedno je i broj izračunatih statičkih vrednosti, u ovom zadatku, 10.

b) Najpre vršimo izračunavanje razmatrajući stablo zaključno sa dubinom 1. Ovom prilikom imamo 2 izračunavanja. Najpre izračunavanje vrednosti čvora A a zatim i čvora B. Sledeći nivo produbljivanja zahteva 4 izračunavanja statičkih vrednosti i to za čvorove C, D, E i F. Četvrti, u ovom primeru poslednji korak produbljivanja, uključuje u razmatranje nivo 3 posmatranog stabla. Ovim dobijamo 10 novih izračunavanja statičkih vrednosti. Na osnovu prethodnog, može se izvesti zaključak da je ukupni broj izračunatih statičkih vrednosti 16.



Slika 94

c) Rešenje je jednostavna primena ranije opisanog minimax algoritma sa alfa-beta odsecanjem. Jedini čvor stabla koji će biti odsečen je čvor K, što je naznačeno na slici 95.



Slika 95

d) Promenom redosleda dece nekog čvora moguće je ranije zaključiti da neke od grana istog roditelja nije potrebno obići. Ono što treba zapaziti jeste da brzina odsecanja nema uticaj na vrednosti čvorova koji se nalaze na nivoima iznad. Iz tog razloga smisleno je razmotriti samo čvorove čiji je određeni broj izlaznih grana isečen. U našem primeru to je samo čvor D. Ideja je da se potreba za odsecanjem uočiti još prilikom obilaska prvog deteta. Stoga zamenom čvorova I i J možemo izvršiti odsecanje ranije nego što je to bio slučaj u prethodnoj tački. Čvorovi I i K u ovom slučaju neće biti obiđeni. Najmanji broj statičkih vrednosti koji se može postići jeste 8.

1.4. Primeri na programskom jeziku Java

Zadatak 33: Osnovni skup klasa

Potrebno je kreirati skup klasa (radni okvir), na programskom jeziku Java, koje čine osnovu za različite algoritme pretraživanja.

Rešenje

Na samom početku, važno je napomenuti da je ovde prikazano jedno od, mnogobrojnih, mogućih rešenja. Kako je poznato iz prakse, projektovanje biblioteke klasa vremenski je zahtevan posao, a kreirani modeli podložni su promenama.

Raniji zadaci iz ovog poglavlja ilustrovali su različite algoritme pretraživanja, pri čemu je svako pretraživanje rezultovalo u kreiranju stabla pretrage. Svaki čvor stabla pretrage opisivao je jedno stanje sistema. S toga, jedna od klasa kreiranog radnog okvira predstavlja opis čvora stabla pretrage. U pitanju je klasa Node. Zavisno od konkretnog problema koji se rešava (odnosno opisa stanja problema), ova klasa može biti dodatno proširena.

```
package rs.etf.es.search.lib;

public class Node {
  private String name;

  public Node(String n) {
    name = n;
  }

  public String getName() {
    return name;
  }

  public String toString() {
    return name;
  }
}
```

Određeni broj algoritama pretraživanja (na primer, metoda planinarenja), zahteva definisanje funkcije za izračunavanje heurističke vrednosti čvora. Kako bi se podržao ovaj zahtev, kreiran je interfejs, HeuristicCalculator, koji definiše metodu calculate, čiji je cilj određivanje heurističke vrednosti za prosleđeni čvor. Metoda calculate, vraća vrednost tipa Value.

```
package rs.etf.es.search.lib;

public interface HeuristicCalculator<E extends Node> {

  public Value calculate(E next);
}
```

```
package rs.etf.es.search.lib;

public interface Value {

  public double calculate();
}
```

Nakon razmatranja, ustanovljeno je da heuristika čvora, u generalnom slučaju, može biti proizvoljne složenosti. Ovde je primenjen pristup, da klasa koja opisuje heuristiku čvora mora implementirati interfejs Value i omogući preslikavanje složenog opisa heuristike u vrednost tipa double (metoda calculate).

Pojam ekspanzije čvora, poznat je iz prethodnih zadataka, ovog poglavlja. Radi se o procesu obilaska određenog čvora i dodavanju stanja u koja se može preći u stablo pretrage. Određivanje stanja u koja se može preći zavisi od problema koji se rešava. Iz navedenog razloga, ekspanzija čvora apstrahovana je interfejsom Operator. Ime kreiranog interfejsa treba da upućuje da je u pitanju operator koji se izvršava nad prosleđenim stanjem a kao rezultat vraća listu stanja u koja se, primenom tog operatora, može preći. Metoda possible interfejsa Operator, definiše da li se nad prosleđenim stanjem operator može primeniti.

```
package rs.etf.es.search.lib;
import java.util.*;

public interface Operator<E extends Node> {

  public boolean possible(E current);

  public List<E> expand(E current);
}
```

Takođe, kroz ranije primere, uvedena je apstrakcija putanje. U pitanju je niz čvorova stabla, pri čemu je prvi čvor na putanji koreni čvor stabla pretrage. Ova apstrakcija opisana je klasom Path.

```
package rs.etf.es.search.lib;
import java.util.*;

public class Path<E extends Node> implements Cloneable {
  private List<E> path = new LinkedList<E>();

  public Path(E node) {
    path.add(node);
  }

  public void addNode(E node) {
    path.add(node);
  }

  public int size() {
    return path.size();
  }

  public E get(int index) {
    return path.get(index);
  }
```

```
  public E getLast() {
    return path.get(path.size() - 1);
  }

  public Path<E> clone() {
    try {
      Path<E> newPath = (Path<E>) super.clone();
      newPath.path = new LinkedList<E>(path);
      return newPath;
    } catch (CloneNotSupportedException e) {
      e.printStackTrace();
    }
    return null;
  }
}
```

Klasa Path omogućava dohvatanje broja čvorova na putanji (metoda size), dohvatanje i-tog čvora putanje (metoda get), dohvatanje poslednjeg čvora na putanji (metoda getLast), dodavanje novog čvora na kraj putanje (metoda addNode) i kloniranje putanje.

Nalik uvedenom interfejsu, HeuristicCalculator, za određivanje heurističke vrednosti čvora, definisan je interfejs, CostCalculator, za određivanje cene putanje. Metoda calculate, interfejsa CostCalculator, vraća vrednost Value tipa, iz istih razloga kao u slučaju HeuristicCalculator interfejsa. Dodatno, kako pojedini algoritmi uzimaju u obzir zbir heuristike i cene (A*), metode calculate oba interfejsa vraćaju isti tip.

```
package rs.etf.es.search.lib;

public interface CostCalculator<E extends Node> {

  public Value calculate(Path<E> path);
}
```

Nakon ekspanzije čvora stabla pretrage, potrebno je novo kreirane putanje dodati u listu putanja za obilazak. Mesto novo kreiranih putanja u listi putanja za obilazak zavisi od algoritma pretraživanja. Na primer, algoritam planinarenja zahteva dodavanje putanja na početak liste pri čemu su nove putanje sortirane na osnovu heurističke vrednosti. Uređivanje putanja apstrahovano je interfejsom Evaluator. Listu putanja koju je potrebno urediti određuje sam algoritam pretrage kao i skup vrednosti na osnovu kojih se vrši uređivanje (u slučaju metoda planinarenja to bi bila lista heurističkih vrednosti poslednjih čvorova na putanji).

```
package rs.etf.es.search.lib;
import java.util.*;

public interface Evaluator<E extends Node> {

  public List<Path<E>> evaluate(List<Path<E>> paths, List<Value> values);
}
```

Klasa Strategy, opisuje osnovu algoritama pretraživanja. Svi algoritmi pretraživanja trebalo bi da nasleđuju ovu klasu.

```
package rs.etf.es.search.lib;
import java.util.*;

public abstract class Strategy<E extends Node> {
  private Operator<E>[] operators;
  private List<StrategyObserver<E>> observers = new
                                    LinkedList<StrategyObserver<E>>();
  private StrategyController<E> controller;

  protected List<Path<E>> history = new LinkedList<Path<E>>();;
  protected List<Path<E>> paths = new LinkedList<Path<E>>();
  private Path<E> current;

  public Strategy(E start, StrategyController<E> controller,
                  Operator<E>... operators) {
    if (start == null)
      throw new NullPointerException("start node is null!");
    this.controller = controller;
    this.operators = operators;
    this.paths.add(new Path<E>(start));
  }

  public void search() {
    updateSearchStarted();
    while (hasNext()) {
      current = next();
      history.add(current);
      updateNextPath();
      if (isEnd())
        break;
      List<Path<E>> newPaths = expandCurrent();
      removeLoops(newPaths);
      updateNewPaths(newPaths);
      addPaths(newPaths);
    }
    updateSearchFinished();
  }

  protected abstract void addPaths(List<Path<E>> newPaths);

  private Path<E> next() {
    return paths.remove(0);
  }

  private void updateNewPaths(List<Path<E>> newPaths) {
    for (int i = 0; i < observers.size(); i++)
      observers.get(i).updateNewPaths(newPaths);
  }

  private void removeLoops(List<Path<E>> expanded) {
    for (int i = expanded.size() - 1; i >= 0; i--) {
      Path<E> newPath = expanded.get(i);
      for (int j = 0; j < newPath.size() - 1; j++)
        if (newPath.get(j).equals(newPath.getLast())) {
          expanded.remove(i);
          break;
        }
    }
  }

  private List<Path<E>> expandCurrent() {
    List<E> nodes = new LinkedList<E>();
```

```
    for (int i = 0; i < operators.length; i++) {
      if (operators[i].possible(current.getLast()))
        nodes.addAll(operators[i].expand(current.getLast()));
    }
    List<Path<E>> newPaths = new LinkedList<Path<E>>();
    for (int i = 0; i < nodes.size(); i++) {
      if (nodes.get(i) == null)
        throw new NullPointerException("expand: null is not valid node!");
      Path<E> newPath = current.clone();
      newPath.addNode(nodes.get(i));
      newPaths.add(newPath);
    }
    return newPaths;
  }

  private boolean hasNext() {
    return !paths.isEmpty();
  }

  private boolean isEnd() {
    if (controller != null)
      return controller.isEnd(this);
    return false;
  }

  public Path<E> getCurrent() {
    return current;
  }

  public List<Path<E>> getPaths() {
    return paths;
  }

  public List<Path<E>> getHistory() {
    return history;
  }

  // OBSERVABLE

  public void addObserver(StrategyObserver<E> observer) {
    if (observer == null)
      return;
    observers.add(observer);
  }

  private void updateSearchStarted() {
    for (int i = 0; i < observers.size(); i++)
      observers.get(i).updateSearchStarted(this);
  }

  private void updateNextPath() {
    for (int i = 0; i < observers.size(); i++) {
      observers.get(i).updateNextPath(this);
    }
  }

  private void updateSearchFinished() {
    for (int i = 0; i < observers.size(); i++)
      observers.get(i).updateSearchFinished(this);
  }
}
```

Klasi Strategy mogu biti pridruženi posmatrači (metoda addObserver). Ovi posmatrači, pretplaćeni su na posmatranje ključnih koraka prilikom pretraživanja. Događaje na koje mogu reagovati definisani su interfejsom StrategyObserver. Kako je retko da posmatrač želi reagovati na sve događaje, kreirana je klasa StrategyObserverImpl koja telo svih metoda ostavlja prazno.

```
package rs.etf.es.search.lib;
import java.util.*;

public interface StrategyObserver<E extends Node> {
  public void updateSearchStarted(Strategy<E> strategy);

  public void updateNewPaths(List<Path<E>> newPaths);

  public void updateNextPath(Strategy<E> strategy);

  public void updateSearchFinished(Strategy<E> strategy);
}
```

```
package rs.etf.es.search.lib;
import java.util.*;

public class StrategyObserverImpl<E extends Node> implements
    StrategyObserver<E> {

  public void updateNewPaths(List<Path<E>> newPaths) {}

  public void updateNextPath(Strategy<E> strategy) {}

  public void updateSearchFinished(Strategy<E> strategy) {}

  public void updateSearchStarted(Strategy<E> strategy) {}
}
```

Može se postaviti pitanje, da li posmatračima treba omogućiti direktan pristup do internih delova strategije ili prosleđivati objekat u kome bi bile kopije na koje se posmatrač pretplatio. U ovde razmatranom okruženju primenjen je prvi pristup, pri čemu se očekuje da posmatrači ispoštuju svoju ulogu (ne vrše promenu sistema).

Sada je moguće razmotriti detalje klase Strategy. Konkretno ključne metode ove klase, search (primetiti da je u pitanju uzorak template method).

```
public void search() {
  updateSearchStarted();
  while (hasNext()) {
    current = next();
    history.add(current);
    updateNextPath();
    if (isEnd())
      break;
    List<Path<E>> newPaths = expandCurrent();
    removeLoops(newPaths);
    updateNewPaths(newPaths);
    addPaths(newPaths);
  }
  updateSearchFinished();
```

```
}
```

Metoda najpre obaveštava sve posmatrače o početku pretraživanja. Zatim sledi petlja koja se zaustavlja nakon što su razmotrene sve otkrivene putanje. Na početku listi putanja sadrži jednu putanju dužine 1 (putanja sa startnim čvorom). U svakom ciklusu petlje razmatra se po jedna putanja iz liste putanja. Radi evidencije o redosledu obilaska putanja, trenutno razmatrana putanja dodaje se u istoriju. Nakon toga, posmatrači se obaveštavaju o izboru naredne putanje. Sledi provera da li je pretragu potrebno završiti ili ne. Ovaj uslov proverava objekat, prosleđen putem konstruktora, klase StrategyController. Ukoliko se koristi podrazumevana implementacija (vraća se vrednost false), pretraživanje se završava kase se isprazni lista putanja.

```
package rs.etf.es.search.lib;

public class StrategyController<E extends Node> {

  public boolean isEnd(Strategy<E> strategy) {
    return false;
  }
}
```

Ukoliko se pretraga ne završava, vrši se ekspanzija trenutne putanje, primenom prosleđenih operatora (pozivaju se samo operatori čija metoda possible vraća vrednost true). Nakon izvršene ekspanzije vrši se uklanjanje svih putanja koje imaju cikluse, posle čega se obaveštenje o novo kreiranim putanjama šalje posmatračima. Petlja se završava pozivom apstraktne metode addPaths, čija se implementacija zahteva za konkretan algoritam pretraživanja i koja je zadužena za smeštanje novih putanja u listu putanja i uređenje liste.

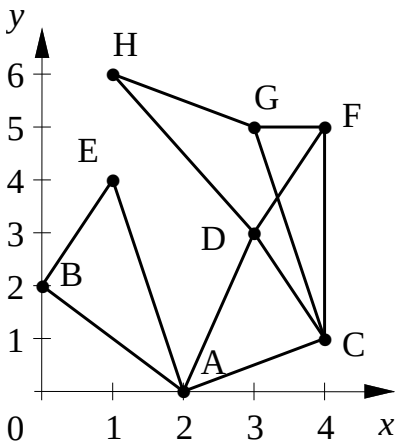
Zadatak 34: Algoritam planinarenja na jeziku Java

a) Napisati skup klasa koje implementiraju odgovarajuće interfejse i nasleđuju odgovarajuće klase iz zadatka 33 u cilju implementacije algoritma pretraživanja metodom planinarenja (engl. *hill-climbing*).

b) Primeniti metodu iz tačke a) za nalaženje puta između tačaka A i H na putnoj mreži sa slike 96, pri čemu heuristička funkcija predstavlja vazdušno rastojanje tekućeg i ciljnog čvora:

$$h = \sqrt{(x - x_c)^2 + (y - y_c)^2}$$

gde su x i y koordinate tekućeg, a x_c i y_c koordinate ciljnog čvora.



Slika 96

Rešenje

a) Kreirana je klasa HillClimbing koja nasleđuje klasu Strategy definisanu u zadatku 33. Implementacija algoritma planinarenja zasnovana je na algoritmu 3 iz dodatka 1:

```
package rs.etf.es.search.algorithms;
import java.util.*;
import rs.etf.es.search.lib.*;

public class HillClimbing<E extends Node> extends Strategy<E> {
  private Evaluator<E> evaluator;
  private HeuristicCalculator<E> calculator;

  public HillClimbing(E start, StrategyController<E> controller,
                      Evaluator<E> evaluator,
                      HeuristicCalculator<E> calculator,
                      Operator<E>... operators) {
    super(start, controller, operators);
    if (calculator == null || evaluator == null)
      throw new NullPointerException("calculator|evaluator is null!");
    this.calculator = calculator;
    this.evaluator = evaluator;
  }

  protected void addPaths(List<Path<E>> newPaths) {
    List<Value> heuristics = new LinkedList<Value>();
    for (int i = 0; i < newPaths.size(); i++) {
      E next = newPaths.get(i).getLast();
      heuristics.add(calculator.calculate(next));
    }
    List<Path<E>> sorted = evaluator.evaluate(newPaths, heuristics);
    paths.addAll(0, sorted);
  }
}
```

Pored objekata koje prihvata konstruktor klase Strategy, konstruktor klase HillClimbing prihvata dodatna dva. Prvi je instanca klase Evaluator, koji u slučaju metode planinarenja vrši uređivanje novo kreiranih putanja (nakon ekspanzije) na osnovu vrednosti heurističkih funkcija krajnjih čvorova na tim putanjama. Metoda addPaths prihvata listu novih putanja,

kreira listu heuristika za ove putanje, vrši uređivanje putanja upotrebom objekta klase Evaluator i tako sortirane dodaje na početak liste putanja koje je potrebno obići.

Najpre je kreirana dodatna bibliotečka klasa, EvaluatorImple, koja implementira interfejs Evaluator. Cilj ove klase bilo je definisanje dve unutrašnje klase za koje se smatra da su pri implementaciji originalnog interfejsa u često upotrebi. U pitanju su klase koje čuvaju parove put-vrednost, i klasa za uređenje ovih objekata na osnovu vrednosti.

```
package rs.etf.es.search.lib;
import java.util.*;

public abstract class EvaluatorImpl<E extends Node> implements Evaluator<E>
{

  public static class PV<T extends Node> {
    private Path<T> path;
    private Value value;

    public PV(Path<T> p, Value v) {
      path = p;
      value = v;
    }

    public Path<T> getPath() {
      return path;
    }

    public Value getValue() {
      return value;
    }
  }

  public static class PVComparator<T extends Node> implements
                                                    Comparator<PV<T>> {
    public int compare(PV<T> pc1, PV<T> pc2) {
      return Double.compare(pc1.getValue().calculate(),
                            pc2.getValue().calculate());
    }
  }
}
```

Za uređenje vrednosti u rastućem poretku kreirana je klasa AscendentEvaluator, koja nasleđuje klase EvaluatorImpl. Najpre se na osnovu dobijenih listi kreiraju parovi put-vrednost, zatim se izvrši sortiranje i kao rezultat vrate uređene putanje.

```
package rs.etf.es.search.algorithms;
import java.util.*;
import rs.etf.es.search.lib.*;

public class AscendantEvaluator<E extends Node> extends EvaluatorImpl<E> {

  public List<Path<E>> evaluate(List<Path<E>> paths, List<Value> values) {
    List<PV<E>> pairs = new LinkedList<PV<E>>();
    for (int i = 0; i < paths.size(); i++) {
      pairs.add(new PV<E>(paths.get(i), values.get(i)));
    }
    Collections.sort(pairs, new PVComparator<E>());
```

```
    List<Path<E>> evaluated = new LinkedList<Path<E>>();
    for (int i = 0; i < pairs.size(); i++) {
      evaluated.add(pairs.get(i).getPath());
    }

    return evaluated;
  }
}
```

Za sortiranje upotrebljena je Javina bibliotečka klasa Collections, pri čemu je za komparator upotrebljena unutrašnja nasleđena klasa PVComparator.

b) Razlikovaćemo dva slučaju vezana za graf nad kojim je potrebno sprovesti obilazak. Jedan, kao u ovom zadatku, kada je kompletan graf poznat unapred, i drugi kada se graf kreira dinamički. U prvom slučaju, pre početka obilaska, primenom bilo kog algoritma pretraživanja, biće izvršeno kreiranje kompletnog grafa. U tu svrhu kreiran je interfejs Graph koji omogućava, dodavanje čvorova grafu, povezivanje čvorova grafa (grane mogu jednosmerne ili dvosmerne), definisanje karakteristike grane (na primer, cena), dohvatanje karakteristike grane, dohvatanje svih suseda i dohvatanje broja čvorova u grafu. Jedna od mogućih implementacija grafa data je na kraju ovog zadatka.

```
package rs.etf.es.graph;
import java.util.*;

public interface Graph<E, T> {

  public void addNode(E node);

  public void addNodes(E... nodes);

  public void connect(E node1, E node2);

  public void connect(E node1, E node2, boolean isDirected);

  public void connect(E node1, E node2, T cost);

  public void connect(E node1, E node2, T cost, boolean isDirected);

  public void connect(E node, E... nodes);

  public void setCost(E node1, E node2, T cost);

  public T getCost(E node1, E node2);

  public List<E> getNeighbors(E node);

  public int size();
}
```

Rešavani problem odnosi se na prostor u 2D ravni i svaki čvor grafa (kasnije čvor stabla pretrage) opisan je položajem u prostoru. U skladu sa tim kreirana je klasa Node2D, za opis čvora stabla pretrage koja nasleđuje klasu Node, definisanu u zadatku 33.

```
package rs.etf.es.search.examples.space;
import rs.etf.es.search.lib.*;

public class Node2D extends Node {
```

```
  private int x;
  private int y;

  public Node2D(String name, int x, int y) {
    super(name);
    this.x = x;
    this.y = y;
  }

  public int getX() {
    return x;
  }

  public int getY() {
    return y;
  }
}
```

Upotrebom definisanog čvora i ranije prikazanog interfejsa Graph, jednostavno je izvršiti opis grafa sa slike 96.

```
package rs.etf.es.search.examples.space;
import rs.etf.es.graph.*;

public class Graph2D {
  private static Node2D start;
  private static Node2D end;
  private static boolean isCreated;
  private static Graph<Node2D, Double> graph;

  public static Graph<Node2D, Double> graph() {
    if (isCreated)
      return graph;

    graph = new GraphImpl<Node2D, Double>();
    isCreated = true;

    Node2D a = new Node2D("A", 2, 0);
    Node2D b = new Node2D("B", 0, 2);
    Node2D c = new Node2D("C", 4, 1);
    Node2D d = new Node2D("D", 3, 3);
    Node2D e = new Node2D("E", 1, 4);
    Node2D f = new Node2D("F", 4, 5);
    Node2D g = new Node2D("G", 3, 5);
    Node2D h = new Node2D("H", 1, 6);

    graph.addNodes(a, b, c, d, e, f, g, h);

    graph.connect(a, b, c, d, e);
    graph.connect(b, a, e);
    graph.connect(c, a, d, f, g);
    graph.connect(d, a, c, f, h);
    graph.connect(e, a, b);
    graph.connect(f, c, d, g);
    graph.connect(g, c, f, h);
    graph.connect(h, d, g);

    start = a;
    end = h;
    return graph;
  }
```

```
  public static Node2D getStart() {
    if (!isCreated)
      graph();
    return start;
  }

  public static Node2D getEnd() {
    if (!isCreated)
      graph();
    return end;
  }
}
```

Kako će biti upotrebljen algoritam planinarenja iz tačke a), potrebno je razmotriti konstruktor klase HillClimbing.

Startni čvor definisan je prethodno prikazanom klasom Graph2D. Kontroler strategije u ovom zadatku potrebno je da zaustavi pretragu kada detektuje unapred definisani čvor. Kako je ovo čest slučaj kreirana klasa EndStrategyController smeštena je u osnovni paket klasa.

```
package rs.etf.es.search.lib;

public class EndStrategyController<E extends Node> extends
    StrategyController<E> {
  private E end;

  public EndStrategyController(E end) {
    this.end = end;
  }

  public boolean isEnd(Strategy<E> strategy) {
    return end.equals(strategy.getCurrent().getLast());
  }
}
```

Sledeći parametar konstruktora je tipa Evaluator. Ovde će biti upotrebljen evaluator definisan u tački a) za potrebe algoritma metodom planinarenja.

Naredni parametar je tipa HeuristicCalculator. Kako je opisano u zadatku 33, objekat ovog tipa vrši izračunavanje vrednosti heuristike za prosleđeni čvor. U ovom primeru ta vrednost je vazdušno rastojanje. Vrednost rastojanja je tipa double. Međutim kako calculate metoda vraća objekat tipa Value kreirana je klasa DoubleValue koja implementira ovaj interfejs i vrši prosleđivanje dobijene vrednosti. Ova klasa smeštena je u osnovni paket jer je čest slučaj za jednostavnije primere.

```
package rs.etf.es.search.lib;

public class DoubleValue implements Value {
  private double val;

  public DoubleValue(double val) {
    this.val = val;
  }

  public double calculate() {
    return val;
```

```
  }
}
```

```
package rs.etf.es.search.examples.space;
import rs.etf.es.search.lib.*;

public class HeuristicCalculator2D implements HeuristicCalculator<Node2D> {
  private Node2D end;

  public HeuristicCalculator2D(Node2D end) {
    this.end = end;
  }

  public Value calculate(Node2D next) {
    return new DoubleValue(Util2D.distance(next, end));
  }
}
```

Prilikom izračunavanja udaljenosti između čvorova upotrebljena je klasa Util2D.

```
package rs.etf.es.search.examples.space;

public class Util2D {

  public static double distance(Node2D first, Node2D second) {
    return (Math.sqrt(Math.pow((first.getX() - second.getX()), 2)
                      + Math.pow((first.getY() - second.getY()), 2)));
  }
}
```

Poslednji argument konstruktora je niz objekata tipa Operator. Ovaj tip definisan je u zadatku 33. Kreirani operator koristi interfejs Graph i prilikom ekspanzije vrši vraćanje suseda čvora koji se ekspanduje. Kako je u pitanju čest slučaj, ovaj operator smešten je u radni okvir.

```
package rs.etf.es.search.lib;
import java.util.*;
import rs.etf.es.graph.*;

public class ExpandGraphNodeOperator<E extends Node> implements Operator<E>
{
  private Graph<E, ?> graph;

  public ExpandGraphNodeOperator(Graph<E, ?> graph) {
    this.graph = graph;
  }

  public List<E> expand(E current) {
    return graph.getNeighbors(current);
  }

  public boolean possible(E current) {
    return true;
  }
}
```

Kako su poznati svi argumenti klase HillClimbing, preostalo je kreiranje glavne klase i pokretanje pretraživanja.

```
package rs.etf.es.search.examples.space.hillclimbing;
import rs.etf.es.search.algorithms.*;
import rs.etf.es.search.examples.space.*;
import rs.etf.es.search.lib.*;

public class HillClimbing2D {
  public static void main(String[] args) {
    StrategyController<Node2D> controller =
                      new EndStrategyController<Node2D>(Graph2D.getEnd());
    Evaluator<Node2D> evaluator = new AscendantEvaluator<Node2D>();
    HeuristicCalculator<Node2D> calculator =
                      new HeuristicCalculator2D(Graph2D.getEnd());
    ExpandGraphNodeOperator<Node2D> operator =
                      new ExpandGraphNodeOperator<Node2D>(Graph2D.graph());

    Strategy<Node2D> strategy =
                      new HillClimbing<Node2D>(Graph2D.getStart(),
                                               controller,
                                               evaluator,
                                               calculator,
                                               operator);

    StrategyObserver2D observer = new StrategyObserver2D();
    strategy.addObserver(observer);

    strategy.search();
  }
}
```

Kako se može videti iz metode main kreiranoj strategiji pretraživanja pridružen je jedan posmatrač. Ovaj posmatrač ima za cilj da na kraju pretrage ispiše pronađenu putanju i redosled obilaska čvorova.

```
package rs.etf.es.search.examples.space;
import java.util.*;
import rs.etf.es.search.lib.*;

public class StrategyObserver2D extends StrategyObserverImpl<Node2D> {

  public void updateSearchFinished(Strategy<Node2D> strategy) {
    System.out.print("history: ");
    printHistory(strategy);
    System.out.print("\npath: ");
    printPath(strategy);
  }

  private void printHistory(Strategy<Node2D> strategy) {
    List<Path<Node2D>> history = strategy.getHistory();
    for (int i = 0; i < history.size(); i++)
      System.out.print(history.get(i).getLast());
  }

  private void printPath(Strategy<Node2D> strategy) {
    Path<Node2D> path = strategy.getCurrent();
    for (int i = 0; i < path.size(); i++)
      System.out.print(path.get(i));
```

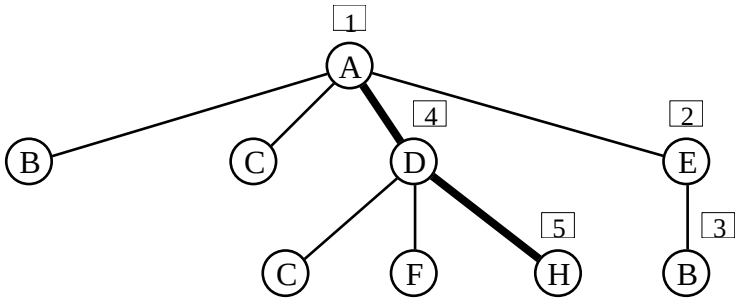
```
  }
}
```

Nakon pokretanja HillClimbing2D dobija se sledeći izlaz:

```
history: AEBDH
path: ADH
```

Elegantniji pristup bila bi upotreba apstraktne fabrike za kreiranje objekata potrebnih konstruktoru klase HillClimbing.

Lako je ustanoviti da je pretraga opisana stablom sa slike 97. I pored toga što uz čvorove nisu navedene vrednosti heurističke funkcije, ove vrednosti se mogu kvalitativno oceniti gledajući datu putnu mrežu.



Slika 97

Jedna od mogućih implementacija interfejsa Graph, data je u nastavku:

```
package rs.etf.es.graph;
import java.util.*;

public class GraphImpl<E, T> implements Graph<E, T> {
  private HashMap<E, List<E>> nodes = new HashMap<E, List<E>>();
  private LinkedList<Pair<E, T>> pairs = new LinkedList<Pair<E, T>>();

  static class Pair<E, T> {
    E n1, n2;
    T c;

    public Pair(E n1, E n2) {
      this(n1, n2, null);
    }

    public Pair(E n1, E n2, T c) {
      this.n1 = n1;
      this.n2 = n2;
      this.c = c;
    }

    public boolean equals(Object o2) {
      if (!(o2 instanceof Pair))
        return false;
      Pair<?, ?> p2 = (Pair<?, ?>) o2;
      return ((n1.equals(p2.n1) && n2.equals(p2.n2))
                    || (n1.equals(p2.n2) && n2.equals(p2.n1)));
    }

    public String toString() {
      return n1 + " " + n2;
    }
```

```
  }

  public void addNode(E node) {
    if (!nodes.containsKey(node)) {
      nodes.put(node, new LinkedList<E>());
    }
  }

  public void addNodes(E... nodes) {
    for (int i = 0; i < nodes.length; i++)
      addNode(nodes[i]);
  }

  public void connect(E node1, E node2) {
    connect(node1, node2, null, false);
  }

  public void connect(E node1, E node2, boolean isDirected) {
    connect(node1, node2, null, isDirected);
  }

  public void connect(E node1, E node2, T cost) {
    connect(node1, node2, cost, false);
  }

  public void connect(E node1, E node2, T cost, boolean isDirected) {
    connectNodes(node1, node2, cost);
    if (!isDirected)
      connectNodes(node2, node1, cost);
  }

  private void connectNodes(E node1, E node2, T cost) {
    addNode(node1);
    addNode(node2);
    List<E> adjacent = nodes.get(node1);
    if (!adjacent.contains(node2))
      adjacent.add(node2);
    int index = pairs.indexOf(new Pair<E, T>(node1, node2));
    if (index == -1)
      pairs.add(new Pair<E, T>(node1, node2, cost));
    else
      pairs.get(index).c = cost;
  }

  public T getCost(E node1, E node2) {
    return pairs.get(pairs.indexOf(new Pair<E, T>(node1, node2))).c;
  }

  public List<E> getNeighbors(E node) {
    return nodes.get(node);
  }

  public void setCost(E node1, E node2, T cost) {
    connectNodes(node1, node2, cost);
    connectNodes(node2, node1, cost);
  }

  public void connect(E node, E... nodes) {
    for (int i = 0; i < nodes.length; i++)
      connect(node, nodes[i]);
  }

  public int size() {
    return nodes.size();
```

```
  }
}
```

Zadatak 35: Algoritam 'prvo najbolji' na jeziku Java

a) Napisati skup klasa koje implementiraju odgovarajuće interfejse i nasleđuju odgovarajuće klase iz zadatka 33 u cilju implementacije algoritma pretraživanja metodom prvo najbolji (engl. *best first*).

b) Primeniti proceduru iz tačke a) za nalaženje puta između tačaka A i H na putnoj mreži sa slike 96 iz zadatka 34, pri čemu heuristička funkcija predstavlja vazdušno rastojanje tekućeg i ciljnog čvora:

$$h = \sqrt{(x - x_c)^2 + (y - y_c)^2}$$

gde su x i y koordinate tekućeg, a x_c i y_c koordinate ciljnog čvora.

Rešenje

a) Pristup rešenju jednak je pristupu koji je korišćen prilikom rešavanja prethodnog zadatka. Naime, kreirana je klasa BestFirst koja nasleđuje klasu Strategy i koja implementira metodu prvo najbolji. Implementacija algortma prvo-najbolji zasnovana je na algoritmu 4 iz dodatka 1.

```
package rs.etf.es.search.algorithms;
import java.util.*;
import rs.etf.es.search.lib.*;

public class BestFirst<E extends Node> extends Strategy<E> {
  private Evaluator<E> evaluator;
  private HeuristicCalculator<E> calculator;

  public BestFirst(E start, StrategyController<E> controller,
                   Evaluator<E> evaluator,
                   HeuristicCalculator<E> calculator,
                   Operator<E>... operators) {
    super(start, controller, operators);
    if (calculator == null || evaluator == null)
      throw new NullPointerException("calculator|evaluator null!");
    this.calculator = calculator;
    this.evaluator = evaluator;
  }

  protected void addPaths(List<Path<E>> newPaths) {
    List<Value> heuristics = new LinkedList<Value>();
    paths.addAll(newPaths);
    for (int i = 0; i < paths.size(); i++) {
      E next = getPaths().get(i).getLast();
      heuristics.add(calculator.calculate(next));
    }
    List<Path<E>> sorted = evaluator.evaluate(getPaths(), heuristics);
    paths = sorted;
  }
}
```

Biće korišćen isti Evaluator kao u zadatku 34.

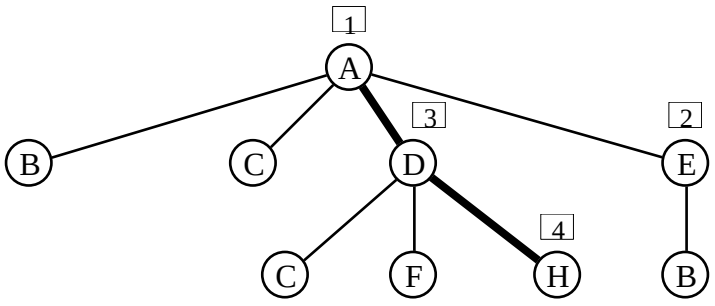
b) Jedina razlika u odnosu na zadatak 34 je prilikom kreiranja strategije pretraživanja.

```
package rs.etf.es.search.examples.space.bestfirst;
import rs.etf.es.search.algorithms.*;
import rs.etf.es.search.examples.space.*;
import rs.etf.es.search.lib.*;

public class BestFirst2D {
  public static void main(String[] args) {
    StrategyController<Node2D> controller =
                      new EndStrategyController<Node2D>(Graph2D.getEnd());
    Evaluator<Node2D> evaluator = new AscendantEvaluator<Node2D>();
    HeuristicCalculator<Node2D> calculator =
                      new HeuristicCalculator2D(Graph2D.getEnd());
    ExpandGraphNodeOperator<Node2D> operator =
                      new ExpandGraphNodeOperator<Node2D>(Graph2D.graph());

    Strategy<Node2D> strategy = new BestFirst<Node2D>(Graph2D.getStart(),
                                                      controller,
                                                      evaluator,
                                                      calculator,
                                                      operator);

    StrategyObserver2D observer = new StrategyObserver2D();
    strategy.addObserver(observer);

    strategy.search();
  }
}
```

Izlaz nakon završetka pretraživanja je:

```
history: AEDH
path: ADH
```

Lako je ustanoviti da je pretraga opisana stablom sa slike 98. I pored toga što uz čvorove nisu navedene vrednosti heurističke funkcije, ove vrednosti se mogu kvalitativno oceniti gledajući datu putnu mrežu. U odnosu na pretragu planinarenjem iz zadatka 34, ekspandovan je jedan čvor manje zahvaljujući globalnom izboru najboljeg čvora pri pretrazi.



Slika 98

Zadatak 36: Algoritam grananja i ograničavanja na jeziku Java

a) Napisati skup klasa koje implementiraju odgovarajuće interfejse i nasleđuju odgovarajuće klase iz zadatka 33 u cilju implementacije algoritma pretraživanja metodom grananja i ograničavanja (engl. *branch-and-bound*).

b) Primeniti proceduru iz tačke a) za nalaženje najkraćeg puta između tačaka A i H na putnoj mreži sa slike 96 iz zadatka 34.

Rešenje

a) Pristup rešavanju sličan je pristupu koji je primenjen prilikom rešavanja zadatka 34. Implementacija algoritma grananja i ograničavanja zasnovana je na algoritmu 5 iz dodatka 1.

```
package rs.etf.es.search.algorithms;
import java.util.*;
import rs.etf.es.search.lib.*;

public class BranchAndBound<E extends Node> extends Strategy<E> {
  private Evaluator<E> evaluator;
  private CostCalculator<E> calculator;

  public BranchAndBound(E start, StrategyController<E> controller,
                        Evaluator<E> evaluator,
                        CostCalculator<E> calculator,
                        Operator<E>... operators) {
    super(start, controller, operators);
    if (evaluator == null || calculator == null)
      throw new NullPointerException("evaluator | calculator is null!");
    this.evaluator = evaluator;
    this.calculator = calculator;
  }

  protected void addPaths(List<Path<E>> newPaths) {
    List<Value> costs = new LinkedList<Value>();
    paths.addAll(newPaths);
    for (int i = 0; i < paths.size(); i++) {
      Path<E> next = paths.get(i);
      costs.add(calculator.calculate(next));
    }
    List<Path<E>> sorted = evaluator.evaluate(getPaths(), costs);
    paths = sorted;
  }
}
```

Upotrebljena je ista klasa Evaluator kao u zadatku 34. Treba primetiti razliku da sada kroz iste argumente klasa Evaluator prihvata listu cena. Grananje i ograničavanje prosleđuje kompletnu listu putanja i cenu za svaku od njih.

b) Argument u konstruktoru klase BranchAndBound koji se razlikuje u odnosu na klasu HillClimbing iz zadatka 34, jeste CostCalculator.

```
package rs.etf.es.search.examples.space;
import rs.etf.es.graph.*;
import rs.etf.es.search.lib.*;
```

```
public class CostCalculator2D implements CostCalculator<Node2D> {
  private Graph<Node2D, Double> graph;

  public CostCalculator2D(Graph<Node2D, Double> graph) {
    this.graph = graph;
  }

  public Value calculate(Path<Node2D> path) {
    double value = 0;
    for (int i = 0; i < path.size() - 1; i++) {
      value += graph.getCost(path.get(i), path.get(i + 1));
    }
    return new DoubleValue(value);
  }

}
```

Kako je za potrebe metode grananja i ograničavanja potrebno definisati i cenu svake grane, kreirana je nova klasa za opis grafa sa slike 96.

```
package rs.etf.es.search.examples.space.branchandbound;
import rs.etf.es.graph.*;
import rs.etf.es.search.examples.space.*;

public class BranchAndBoundGraph2D {
  private static Node2D start;
  private static Node2D end;
  private static boolean isCreated;
  private static Graph<Node2D, Double> graph;

  public static Graph<Node2D, Double> graph() {
    if (isCreated)
      return graph;

    graph = new GraphImpl<Node2D, Double>();
    isCreated = true;

    Node2D a = new Node2D("A", 2, 0);
    Node2D b = new Node2D("B", 0, 2);
    Node2D c = new Node2D("C", 4, 1);
    Node2D d = new Node2D("D", 3, 3);
    Node2D e = new Node2D("E", 1, 4);
    Node2D f = new Node2D("F", 4, 5);
    Node2D g = new Node2D("G", 3, 5);
    Node2D h = new Node2D("H", 1, 6);

    graph.addNodes(a, b, c, d, e, f, g, h);

    graph.connect(a, b, c, d, e);
    graph.setCost(a, b, Util2D.distance(a, b));
    graph.setCost(a, c, Util2D.distance(a, c));
    graph.setCost(a, d, Util2D.distance(a, d));
    graph.setCost(a, e, Util2D.distance(a, e));

    graph.connect(b, e);
    graph.setCost(b, e, Util2D.distance(b, e));

    graph.connect(c, d, f, g);
    graph.setCost(c, d, Util2D.distance(c, d));
    graph.setCost(c, f, Util2D.distance(c, f));
    graph.setCost(c, g, Util2D.distance(c, g));
```

```
    graph.connect(d, f, h);
    graph.setCost(d, f, Util2D.distance(d, f));
    graph.setCost(d, h, Util2D.distance(d, h));

    graph.connect(f, g);
    graph.setCost(f, g, Util2D.distance(f, g));

    graph.connect(g, h);
    graph.setCost(g, h, Util2D.distance(g, h));

    start = a;
    end = h;

    return graph;
  }

  public static Node2D getStart() {
    if (!isCreated)
      graph();
    return start;
  }

  public static Node2D getEnd() {
    if (!isCreated)
      graph();
    return end;
  }
}
```

Ostali argumenti klasa BranchAndBound isti su kao arumenti klase HillClimbing iz zadatka 34. Kreirana je klasa BranchAndBound2D koja prikazuje upotrebu definisane metode pretraživanja.

```
package rs.etf.es.search.examples.space.branchandbound;
import rs.etf.es.search.algorithms.*;
import rs.etf.es.search.examples.space.*;
import rs.etf.es.search.lib.*;

public class BranchAndBound2D {
  public static void main(String[] args) {
    StrategyController<Node2D> controller =
        new EndStrategyController<Node2D>(BranchAndBoundGraph2D.getEnd());
    Evaluator<Node2D> evaluator = new AscendantEvaluator<Node2D>();
    CostCalculator<Node2D> calculator =
        new CostCalculator2D(BranchAndBoundGraph2D.graph());
    ExpandGraphNodeOperator<Node2D> operator =
        new ExpandGraphNodeOperator<Node2D>(BranchAndBoundGraph2D.graph());
    Strategy<Node2D> strategy =
        new BranchAndBound<Node2D>(BranchAndBoundGraph2D.getStart(),
                                   controller,
                                   evaluator,
                                   calculator,
                                   operator);

    StrategyObserver2D observer = new StrategyObserver2D();
    strategy.addObserver(observer);

    strategy.search();
  }
}
```

Nakon završetka pretraživanja dobija se sledeći izlaz:

```
history: ACBDEDECFFGBGFH
path: ADH
```

Stablo pretrage je znatno veće od stabala pretrage po metodima planinarenja i 'prvo najbolji' ali je garantovana optimalnost rešenja. I pored toga što je ciljni čvor unesen u stablo pretrage već pri četvrtoj ekspanziji, ovaj čvor je obiđen tek pošto su produžene sve parcijalne putanje kraće od ciljne putanje.

Zadatak 37: Putna mreža

a) Napisati skup klasa na programskom jeziku Java koje omogućavaju pretraživanje grafa metodom A*. Kao osnovu koristiti radno okruženje definisano u zadatku 33.

b) Primeniti metodu iz tačke a) za nalaženje puta između tačaka S i G na putnoj mreži sa slike 24 iz zadatka 12.

Rešenje

a) Klasa AStar, nasleđuje klasu Strategy i implementira algoritam 6 iz dodatka 1.

```
package rs.etf.es.search.algorithms;
import java.util.*;
import rs.etf.es.search.lib.*;

public class AStar<E extends Node> extends Strategy<E> {
  private Evaluator<E> evaluator;
  private CostCalculator<E> cCalculator;
  private HeuristicCalculator<E> hCalculator;

  public AStar(E start, StrategyController<E> controller,
               Evaluator<E> evaluator, HeuristicCalculator<E> hCalculator,
               CostCalculator<E> cCalculator, Operator<E>... operators) {
    super(start, controller, operators);
    if (evaluator == null || cCalculator == null || hCalculator == null)
      throw new NullPointerException("evaluator | calculator is null!");
    this.evaluator = evaluator;
    this.hCalculator = hCalculator;
    this.cCalculator = cCalculator;
  }

  protected void addPaths(List<Path<E>> newPaths) {
    List<Value> costs = new LinkedList<Value>();
    List<Value> heuristics = new LinkedList<Value>();
    checkOrder(newPaths);
    paths.addAll(newPaths);
    for (int i = 0; i < getPaths().size(); i++) {
      Path<E> next = getPaths().get(i);
      costs.add(cCalculator.calculate(next));
      heuristics.add(hCalculator.calculate(next.getLast()));
    }
    heuristics.addAll(costs);
    List<Path<E>> sorted = evaluator.evaluate(getPaths(), heuristics);
    paths = reducePaths(sorted);
  }
```

```
  private void checkOrder(List<Path<E>> newPaths) {
    for (int i = newPaths.size() - 1; i >= 0; i--)
      for (int j = 0; j < history.size(); j++)
        if (equalEnd(history.get(j), newPaths.get(i)))
          newPaths.remove(i);
  }

  private List<Path<E>> reducePaths(List<Path<E>> sorted) {
    List<Path<E>> reduced = new LinkedList<Path<E>>();

    for (int i = 0; i < sorted.size(); i++) {
      int j;
      for (j = 0; j < reduced.size(); j++)
        if (equalEnd(sorted.get(i), reduced.get(j)))
          break;
      if (j == reduced.size())
        reduced.add(sorted.get(i));
    }
    return reduced;
  }

  private boolean equalEnd(Path<E> p1, Path<E> p2) {
    return p1.getLast().equals(p2.getLast());
  }
}
```

Metod A* vrši sortiranje liste putanja na osnovu zbira cene putanje i heurističke vrednosti poslednjeg čvora na toj putanji. S toga se objektu klase Evaluator vrši prosleđivanje liste koja sadrži kako vrednosti cene tako i vrednosti heuristike. U skladu sa ovim kreiran je klasa AStarEvaluator, koja prihvata listu putanja i listu u kojoj se nazale cene i heuristike, razdvaja cene i heuristike, pronalazi sumu cena i heurističkih vrednosti, vrši sortiranje putanja na osnovu ovih vrednosti i vraća uređenu listu.

```
package rs.etf.es.search.algorithms;
import java.util.*;
import rs.etf.es.search.lib.*;

public class AStarEvaluator<E extends Node> extends EvaluatorImpl<E> {

  public List<Path<E>> evaluate(List<Path<E>> paths, List<Value> values) {
    List<Value> costs = new LinkedList<Value>();
    List<Value> heuristics = new LinkedList<Value>();

    for (int i = 0; i < values.size(); i++)
      if (i < values.size() / 2)
        heuristics.add(values.get(i));
      else
        costs.add(values.get(i));

    List<PV<E>> pairs = new LinkedList<PV<E>>();
    for (int i = 0; i < paths.size(); i++) {
      pairs.add(new PV<E>(paths.get(i),
                          new DoubleValue(heuristics.get(i).calculate()
                                          + costs.get(i).calculate())));
    }
    Collections.sort(pairs, new PVComparator<E>());
    List<Path<E>> evaluated = new LinkedList<Path<E>>();
    for (int i = 0; i < pairs.size(); i++) {
```

```
      evaluated.add(pairs.get(i).getPath());
    }

    return evaluated;
  }
}
```

b) U skladu sa razmatranim problemom, putnom mrežom, kreirana je klasa RoadNode koja opisuje čvor grafa pretrage. Pridružena vrednost predstavlja odstojanje do odredišnog čvora, koje je dato postavkom zadatka.

```
package rs.etf.es.search.examples.road;
import rs.etf.es.search.lib.*;

public class RoadNode extends Node {
  private double distance;

  public RoadNode(String n, double distance) {
    super(n);
    this.distance = distance;
  }

  public double getDistance() {
    return distance;
  }
}
```

Računanje heurističke vrednosti u ovom slučaju je jednostavno, jer je potrebno proslediti podatak koji čvor poseduje.

```
package rs.etf.es.search.examples.road;
import rs.etf.es.search.lib.*;

public class RoadHeuristicCalculator implements
HeuristicCalculator<RoadNode> {

  public Value calculate(RoadNode next) {
    return new DoubleValue(next.getDistance());
  }
}
```

Na osnovu definisanog čvora stabla pretrage jednostavno se kreira graf dat na slici 12.

```
package rs.etf.es.search.examples.road;
import rs.etf.es.graph.*;

public class Road {
  private static Graph<RoadNode, Double> graph;
  private static RoadNode start;
  private static RoadNode end;
  private static boolean isDefined;

  public static Graph<RoadNode, Double> graph() {
    if (isDefined)
      return graph;
```

```
    graph = new GraphImpl<RoadNode, Double>();
    isDefined = true;

    RoadNode s = new RoadNode("S", 11.5);
    RoadNode a = new RoadNode("A", 10.4);
    RoadNode b = new RoadNode("B", 6.7);
    RoadNode c = new RoadNode("C", 7.0);
    RoadNode d = new RoadNode("D", 8.9);
    RoadNode e = new RoadNode("E", 6.9);
    RoadNode f = new RoadNode("F", 3.0);
    RoadNode g = new RoadNode("G", 0);

    graph.addNodes(s, a, b, c, d, e, f, g);

    graph.connect(s, a, 3.0);
    graph.connect(s, d, 10.0);

    graph.connect(a, b, 4.0);
    graph.connect(a, d, 5.0);

    graph.connect(b, c, 2.0);
    graph.connect(b, d, 6.0);

    graph.connect(d, e, 2.0);

    graph.connect(e, f, 4.0);

    graph.connect(f, g, 3.0);

    start = s;
    end = g;
    return graph;
  }

  public static RoadNode getStart() {
    graph();
    return start;
  }

  public static RoadNode getEnd() {
    graph();
    return end;
  }
}
```

Računanje cene putanje za nijansu je kompleksije od računanje heuristike čvora i zasniva se na sabiranju vrednosti cena pridruženih granama grafa.

```
package rs.etf.es.search.examples.road;
import rs.etf.es.graph.*;
import rs.etf.es.search.lib.*;

public class RoadCostCalculator implements CostCalculator<RoadNode> {
  private Graph<RoadNode, Double> graph;

  public RoadCostCalculator(Graph<RoadNode, Double> graph) {
    this.graph = graph;
  }
```

```
  public Value calculate(Path<RoadNode> path) {
    double value = 0;
    for (int i = 0; i < path.size() - 1; i++) {
      value += graph.getCost(path.get(i), path.get(i + 1));
    }
    return new DoubleValue(value);
  }
}
```

Preostalo je definisanje klase koja vrši kreiranje instance klase AStar sa odgovarajućim parametrima i pokretanje pretraživanja.

```
package rs.etf.es.search.examples.road.astar;
import rs.etf.es.search.algorithms.*;
import rs.etf.es.search.examples.road.*;
import rs.etf.es.search.lib.*;

public class RoadAStar {

  public static void main(String[] args) {
    StrategyController<RoadNode> controller =
                    new EndStrategyController<RoadNode>(Road.getEnd());
    Evaluator<RoadNode> evaluator = new AStarEvaluator<RoadNode>();
    HeuristicCalculator<RoadNode> hCalculator =
                    new RoadHeuristicCalculator();
    CostCalculator<RoadNode> cCalculator =
                    new RoadCostCalculator(Road.graph());
    ExpandGraphNodeOperator<RoadNode> operator =
                    new ExpandGraphNodeOperator<RoadNode>(Road.graph());

    Strategy<RoadNode> strategy = new AStar<RoadNode>(Road.getStart(),
                                                      controller,
                                                      evaluator,
                                                      hCalculator,
                                                      cCalculator,
                                                      operator);

    RoadStrategyObserver observer = new RoadStrategyObserver();
    strategy.addObserver(observer);

    strategy.search();
  }
}
```

Kreirani kontroler pretrage istovetan je kontroleru definisanom u zadatku 34.
Novo kreirana klasa RoadStrategyObserver vrši posmatranje izvršavanja i prikazuje rezultate pretrage.

```
package rs.etf.es.search.examples.road;
import java.util.*;
import rs.etf.es.search.lib.*;

public class RoadStrategyObserver extends StrategyObserverImpl<RoadNode> {

  public void updateSearchFinished(Strategy<RoadNode> strategy) {
    System.out.print("history: ");
    printHistory(strategy);
    System.out.print("\npath: ");
    printPath(strategy);
```

```
  }

  private void printHistory(Strategy<RoadNode> strategy) {
    List<Path<RoadNode>> history = strategy.getHistory();
    for (int i = 0; i < history.size(); i++)
      System.out.print(history.get(i).getLast());
  }

  private void printPath(Strategy<RoadNode> strategy) {
    Path<RoadNode> path = strategy.getCurrent();
    for (int i = 0; i < path.size(); i++)
      System.out.print(path.get(i));
  }

}
```

Izvršavanje klase RoadAStar generiše sledeći izlaz:

```
history: SABCDEFG
path: SADEFG
```

Čitaocu se preporučuje da izvrši pretragu kreiranog grafa drugim algoritmima pretraživanja.

Zadatak 38: Pregovori o razoružanju

Zemlje X i Y pregovaraju o razoružanju. Svaka ima projektile različite snage, pri čemu je snaga izražena jednim brojem. Svaki korak u pregovorima rezultuje jednom od sledeće dve akcije:

1) eliminiše sa obe strane po jedan projektil iste snage, ili

2) eliminiše na jednoj strani projektil snage S a na drugoj dva projektila čija je ukupna snaga S.

Navesti niz koraka koji će dovesti do eliminacije svih projektila sa jedne ili obe strane, pri čemu ni u jednom koraku nijedna strana ne sme imati više od jednog projektila od druge strane. Početno stanje je sledeće:

Snage projektila zemlje X: 9, 11, 9, 3, 4, 5, 7, 18

Snage projektila zemlje Y: 12, 2, 16, 5, 7, 4, 20

Napisati program na jeziku Java koji rešava zadati problem.

Rešenje

Definišimo operatore pretrage u skladu sa uslovom zadatka:

1. `op1(P)` označava uklanjanje po jednog projektila snage P iz svake od lista. Preduslov za primenu ovog operatora je da se i u listi X i u listi Y nalazi projektil snage P.
2. `op2(P1,P2,P3)` označava uklanjanje iz liste X projektila P1 i P2, a iz liste Y projektila P3. Preduslov za primenu ovog operatora je da je lista X jednaka ili duža najviše za 1 od liste Y i da je P1 + P2 = P3. Uslov vezan za dužine lista X i Y potreban je da se obezbedi da, prema postavci zadatka, ni u jednom trenutku procesa uklanjanja projektila razlika dužina ove dve liste ne bude veća od 1.

3. `op3(P1,P2,P3)` označava uklanjanje iz liste Y projektila P1 i P2, a iz liste X projektila P3. Preduslov za primenu ovog operatora je da je lista Y jednaka ili duža najviše za 1 od liste X i da je P1 + P2 = P3.

Algoritam pretrage po dubini izabran je na osnovu sledećih razmatranja:

- Rešenje se može naći u relativno malom broju koraka. Taj broj manji je ili jednak dužini kraće liste iz početnog stanja pošto se svakom primenom operatora iz obe liste uklanja bar po jedan projektil.
- Problem ima relativno veliki faktor grananja. Operatori `op2` i `op3` mogu se na određeno stanje sa različitim argumentima. Na primer, ako je u određenom stanju pretrage dužina liste X, koju ćemo označiti sa n, manja do jednaka dužini liste Y, potrebno je razmotriti $\binom{n}{2}$ različitih kombinacija argumenata operatora `op2` gde $\binom{n}{2}$ = n * (n-1) predstavlja broj različitih parova projektila iz liste od n projektila. Naravno, mora biti zadovoljen uslov da zbir snaga izabranih projektila odgovara snazi jednog projektila iz liste Y, što smanjuje broj primena operatora `Op2`, ali je taj broj i dalje veliki. U početnom stanju treba, na primer, ispitati 56 parova projektila iz liste X od kojih uslove zadovoljava 9 parova: (9,11), (9,3), (9,7), (11,9), (11,5), (9,3), (9,7), (3,4) i (5,7). Neki parovi se ponavljaju jer se u listi X nalaze dva različita projektila iste snage 9.
- Graf pretrage je acikličan jer nova stanja poseduju manji broj projektila od stanja - prethodnika u grafu pretrage.

Najpre je kreirana klasa Depth koja implementira algoritam pretraživanja metodom obilaska čvorova po dubini. Algoritam sve novo kreirane putanje stavlja na početak liste putanja koje je potrebno obići. Kako je poznato ovaj metod ne zahteva heuristiku čvora niti cenu putanje te stoga nije potrebna ni klasa Evaluator.

```
package rs.etf.es.search.algorithms;
import java.util.*;
import rs.etf.es.search.lib.*;

public class Depth<E extends Node> extends Strategy<E> {

  public Depth(E start, StrategyController<E> controller,
               Operator<E>... operators) {
    super(start, controller, operators);
  }

  protected void addPaths(List<Path<E>> newPaths) {
    paths.addAll(0, newPaths);
  }
}
```

Čvor stabla pretrage, stanje sistema, opisan je klasom NegotiateNode. Stanje sistema sastoji se od dva niza vrednosti koje opisuje snage projektila dve zemlje (liste X i Y). Polje info nosi dodatnu informaciju o nastanku čvora. Ova informacija biće upisana od strane operatora koji je kreirao čvor. Metoda isEmpty vrši proveru da li je u pitanju čvor u kome su eliminisani projektili sa jedne ili obe strane (rešenje problema na osnovu postavke zadatka).

```
package rs.etf.es.search.examples.negotiate;
import java.util.*;
```

```
import rs.etf.es.search.lib.*;

public class NegotiateNode extends Node implements Cloneable {
  private static int id = 0;
  private List<Integer> X;
  private List<Integer> Y;

  private String info;

  public NegotiateNode(Integer[] X, Integer[] Y) {
    super(Integer.toString(id++));
    if (X.length - Y.length > 1 || X.length - Y.length < -1)
      throw new RuntimeException("sizes of the lists are not correct");
    this.X = new LinkedList<Integer>();
    this.Y = new LinkedList<Integer>();
    for (int i = 0; i < X.length; i++)
      this.X.add(X[i]);
    for (int i = 0; i < Y.length; i++)
      this.Y.add(Y[i]);
  }

  public List<Integer> getX() {
    return X;
  }

  public List<Integer> getY() {
    return Y;
  }

  public void setInfo(String i) {
    info = i;
  }

  public String getInfo() {
    return info;
  }

  protected NegotiateNode clone() {
    NegotiateNode pn = null;
    try {
      pn = (NegotiateNode) super.clone();
      pn.X = new LinkedList<Integer>();
      pn.Y = new LinkedList<Integer>();
      for (int i = 0; i < X.size(); i++)
        pn.X.add(X.get(i));
      for (int i = 0; i < Y.size(); i++)
        pn.Y.add(Y.get(i));
    } catch (Exception e) {
    }
    return pn;
  }

  public boolean isEmpty() {
    return X.size() == 0 || Y.size() == 0;
  }

  public String toString() {
    return "X " + X.toString() + "  Y " + Y.toString();
  }
}
```

Klasa NegotiateStrategyController, predstavlja kontroler strategije. Logika je tako implementirana da se zaustavljanje pretrage dešava kada je pronađeno max rešenja, odnosno sva rešenja problema, ukoliko se koristi podrazumevani konstruktor.

```
package rs.etf.es.search.examples.negotiate;
import rs.etf.es.search.lib.*;

public class NegotiateStrategyController extends
       StrategyController<NegotiateNode> {

  private int counter;
  private int max;

  public NegotiateStrategyController() {
    this(-1);
  }

  public NegotiateStrategyController(int max) {
    this.max = max;
  }

  public boolean isEnd(Strategy<NegotiateNode> strategy) {
    if (strategy.getCurrent().getLast().isEmpty())
      counter++;
    return counter == max;
  }
}
```

Kako je razmatrano na početku rešenja zadatka, u skladu sa uslovima zadatka, definisana su tri operatora. Ovi operatori definišu prelaze iz razmatranog stanja. Svaki od operatora implementira interfejs Operator, definisan u zadatku 33.

```
package rs.etf.es.search.examples.negotiate;
import java.util.*;
import rs.etf.es.search.lib.*;

public class Op1 implements Operator<NegotiateNode> {

  public List<NegotiateNode> expand(NegotiateNode current) {
    List<NegotiateNode> nodes = new LinkedList<NegotiateNode>();
    for (int i = 0; i < current.getX().size(); i++)
      for (int j = 0; j < current.getY().size(); j++)
        if (current.getX().get(i) == current.getY().get(j)) {
          NegotiateNode pn = current.clone();
          int x = pn.getX().remove(i);
          int y = pn.getY().remove(j);
          String info = "op1 (" + x + " " + y + ")";
          pn.setInfo(info);
          nodes.add(pn);
        }
    return nodes;
  }

  public boolean possible(NegotiateNode current) {
    return true;
  }
}
```

```
package rs.etf.es.search.examples.negotiate;
import java.util.*;
import rs.etf.es.search.lib.*;

public class Op2 implements Operator<NegotiateNode> {

  public boolean possible(NegotiateNode node) {
    return node.getX().size() == node.getY().size()
           || node.getX().size() - 1 == node.getY().size();
  }

  public List<NegotiateNode> expand(NegotiateNode current) {
    List<NegotiateNode> nodes = new LinkedList<NegotiateNode>();
    for (int i = 0; i < current.getX().size() - 1; i++)
      for (int j = i + 1; j < current.getX().size(); j++)
        for (int k = 0; k < current.getY().size(); k++)
          if (current.getX().get(i) + current.getX().get(j)
                                    == current.getY().get(k)) {
            NegotiateNode pn = current.clone();
            int x2 = pn.getX().remove(j);
            int x1 = pn.getX().remove(i);
            int y = pn.getY().remove(k);
            String info = "op2 (" + x1 + " " + x2 + " " + y + ")";
            pn.setInfo(info);
            nodes.add(pn);
          }
    return nodes;
  }
}


package rs.etf.es.search.examples.negotiate;
import java.util.*;
import rs.etf.es.search.lib.*;

public class Op3 implements Operator<NegotiateNode> {

  public boolean possible(NegotiateNode node) {
    return node.getX().size() == node.getY().size()
           || node.getX().size() == node.getY().size() - 1;
  }

  public List<NegotiateNode> expand(NegotiateNode current) {
    List<NegotiateNode> nodes = new LinkedList<NegotiateNode>();
    for (int i = 0; i < current.getX().size(); i++)
      for (int j = 0; j < current.getY().size() - 1; j++)
        for (int k = j + 1; k < current.getY().size(); k++)
          if (current.getX().get(i) == current.getY().get(j)
                                     + current.getY().get(k)) {
            NegotiateNode pn = current.clone();
            int x = pn.getX().remove(i);
            int y1 = pn.getY().remove(k);
            int y2 = pn.getY().remove(j);
            String info = "op3 (" + x + " " + y1 + " " + y2 + ")";
            pn.setInfo(info);
            nodes.add(pn);
          }
    return nodes;
  }
}
```

Dodatno, kreirana je klasa NegotiateObserver koja posmatra izvršavanje pretrage. Klasa ima za cilj ispis pronađenih rešenja, kao i ispis njihovog broja.

```
package rs.etf.es.search.examples.negotiate;
import rs.etf.es.search.lib.*;

public class NegotiateObserver extends StrategyObserverImpl<NegotiateNode>
{
  int c = 0;

  public void updateNextPath(Strategy<NegotiateNode> strategy) {
    if (strategy.getCurrent().getLast().isEmpty())
      printOperation(strategy.getCurrent());
  }

  public void printOperation(Path<NegotiateNode> current) {
    c++;
    System.out.println("-------------");
    for (int i = 1; i < current.size(); i++)
      System.out.print(current.get(i).getInfo());
    System.out.println();
  }

  public void updateSearchFinished(Strategy<NegotiateNode> strategy) {
    System.out.println(c);
  }
}
```

Klasa NegotiateMain, prikazuje pretragu upotrebom kreiranih klasa za primer iz postavke zadatka.

```
package rs.etf.es.search.examples.negotiate;
import rs.etf.es.search.algorithms.*;
import rs.etf.es.search.lib.*;

public class NegotiateMain {
  static Integer[] X = {
      9, 11, 9, 3, 4, 5, 7, 18
  };
  static Integer[] Y = {
      12, 2, 16, 5, 7, 4, 20
  };

  public static void main(String[] args) {
    StrategyController<NegotiateNode> controller =
                    new NegotiateStrategyController(3);
    Operator<NegotiateNode> op1 = new Op1();
    Operator<NegotiateNode> op2 = new Op2();
    Operator<NegotiateNode> op3 = new Op3();

    NegotiateNode start = new NegotiateNode(X, Y);
    Strategy<NegotiateNode> strategy =
                    new Depth<NegotiateNode>(start,
                                             controller,
                                             op1, op2, op3);

    NegotiateObserver observer = new NegotiateObserver();
    strategy.addObserver(observer);
```

```
    strategy.search();
  }
}
```

Bitno je primetiti, da u ovom primeru graf pretrage nije poznat unapred. Naime, on se kreira u vreme izvršavanja primenom ranije definisanih operatora. Operatori imaju za cilj ekspanziju prosleđenog čvora u skladu sa definicijom.

Ukoliko se izostavi argument konstruktora, vrši se pretraživanje svih stanja i pronalaženje svih rešenja problema. Ukupan broj rešenja je 1056. Prva tri rešenja koje program ispisuje su:

```
-------------
op1(4 4) op1(5 5) op1(7 7) op2(9 11 20) op2(9 3 12) op3(18 16 2)
-------------
op1(4 4) op1(5 5) op1(7 7) op2(9 11 20) op3(18 16 2) op2(9 3 12)
-------------
op1(4 4) op1(5 5) op1(7 7) op2(9 3 12) op2(11 9 20) op3(18 16 2)
3
```

Zadatak 39: Problem N kraljica

Na šahovsku tablu dimenzije *n* x *n* potrebno je smestiti *n* kraljica tako da nijedna od njih ne napada bilo koju drugu. Rešiti problem primenom neke od metoda pretraživanja po izboru.

Rešenje

Problem koji se rešava, jeste generalizacija problema šest kraljica iz zadatka 20. Razmatranja iz zadatka 20, važi i ovde. Naime, biće upotrebljena ista heuristika, a od algoritama pretraživanja izabran je metod planinarenja.

Metod planinarenja kao i Evaluator koji ide uz ovaj metod pretrage, implementirani su u zadatku 34.

Ovde je potrebno definisati stanje pretrage, operatore za ekspanziju čvora, funkciju za izračunavanje heuristike čvora, kontroler stretegije pretraživanja i posmatrač pretrege.

Klasa NQueenNode definiše jedno stanje pretrage. Stanje je opisano šahovskom tablom, rednim brojem naredne vrste koju je potrebno popuniti i indekse vrste i kolone polja koje je poslednje setovano u razmatranom stanju.

```
package rs.etf.es.search.examples.nqueens;
import rs.etf.es.search.lib.*;

public class NQueenNode extends Node implements Cloneable {
  private static int id;
  private boolean table[][];
  private int n;

  private int nextRow = 0;
  private int newI = -1;
  private int newJ = -1;

  public NQueenNode(int n) {
    super(Integer.toString(id));
    this.n = n;
```

```
    this.table = new boolean[n][n];
    initTable();
  }

  private void initTable() {
    for (int i = 0; i < table.length; i++)
      for (int j = 0; j < table[i].length; j++)
        table[i][j] = false;
  }

  public int getN() {
    return n;
  }

  public int getNewI() {
    return newI;
  }

  public int getNewJ() {
    return newJ;
  }

  public boolean hasNew() {
    return newI != -1 && newJ != -1;
  }

  public boolean hasNextRow() {
    return nextRow < n;
  }

  public int getNextRow() {
    return nextRow;
  }

  public void setNextRow(int next) {
    nextRow = next;
  }

  public boolean isSet(int i, int j) {
    return table[i][j];
  }

  public void set(int i, int j, boolean f) {
    table[i][j] = f;
    newI = i;
    newJ = j;
  }

  public String toString() {
    String s = "";
    for (int i = 0; i < table.length; i++) {
      for (int j = 0; j < table[i].length; j++)
        s += table[i][j] ? "Q " : "_ ";
      s += "\n";
    }
    return s;
  }

  public NQueenNode clone() {
    NQueenNode newNode = null;
    try {
      newNode = (NQueenNode) super.clone();
      newNode.table = new boolean[n][n];
      for (int i = 0; i < table.length; i++)
```

```
        for (int j = 0; j < table.length; j++)
          newNode.table[i][j] = table[i][j];
    } catch (CloneNotSupportedException e) {
      e.printStackTrace();
    }
    return newNode;
  }
}
```

Računanje heurističke vrednosti čvora, računa se po formuli iz zadatka 20, a opisano je klasom NQueenHeuristicCalculator.

```
package rs.etf.es.search.examples.nqueens;
import rs.etf.es.search.lib.*;

public class NQueenHeuristicCalculator implements
    HeuristicCalculator<NQueenNode> {

  public Value calculate(NQueenNode next) {
    if (!next.hasNew())
      return new DoubleValue(0);
    return new DoubleValue(h(next));
  }

  private double h(NQueenNode next) {
    return 6 * diag(next.getNewI(), next.getNewJ(), next.getN())
           + next.getNewJ();
  }

  private double diag(int i, int j, int n) {
    double diagonal = Math.min(i, j) + Math.min(n - i, n - j) + 1;
    double antidiagonal = Math.min(i, n - j) + Math.min(j, n - i) + 1;
    return Math.max(diagonal, antidiagonal);
  }
}
```

Operator za ekspanziju za tekuće stanje pronalazi moguće naslednike. Preciznije, ukoliko postoji ne popunjena vrsta matrice pokušava se smeštanje kraljice u svaku od kolona naredne slobodne vrste. Samo validne putanje (putanje u kojima raspored kraljica zadovoljava uslove zadatka), dodaju se u listu naslednika.

```
package rs.etf.es.search.examples.nqueens;
import java.util.*;
import rs.etf.es.search.lib.Operator;

public class NQueenOperator implements Operator<NQueenNode> {

  public List<NQueenNode> expand(NQueenNode current) {
    List<NQueenNode> nodes = new LinkedList<NQueenNode>();
    for (int j = 0; j < current.getN(); j++) {
      NQueenNode newNode = current.clone();
      newNode.setNextRow(current.getNextRow() + 1);
      newNode.set(current.getNextRow(), j, true);
      if (!isAttack(newNode))
        nodes.add(newNode);
    }
    return nodes;
```

```
  }

  public boolean possible(NQueenNode current) {
    return current.hasNextRow();
  }

  private boolean isAttack(NQueenNode newNode) {
    if (existInTheColumn(newNode))
      return true;

    if (existOnDiagonale(newNode))
      return true;

    return false;
  }

  private boolean existInTheColumn(NQueenNode newNode) {
    int newJ = newNode.getNewJ();
    int newI = newNode.getNewI();
    for (int i = 0; i < newI; i++)
      if (newNode.isSet(i, newJ))
        return true;
    return false;
  }

  private boolean existOnDiagonale(NQueenNode newNode) {
    int newJ = newNode.getNewJ();
    int newI = newNode.getNewI();
    for (int i = 0; i < newI; i++)
      for (int j = 0; j < newNode.getN(); j++)
        if (newNode.isSet(i, j) &&
                        Math.abs(newI - i) == Math.abs(newJ - j))
          return true;
    return false;
  }
}
```

Usvojeno je da se pretraga završava prilikom otkrivanja prvog rešenja. Preporučuje se čitaocu da izvrši izmenu u cilju generisanja prvih n rešenja.

```
package rs.etf.es.search.examples.nqueens;
import rs.etf.es.search.lib.*;

public class NQueenStrategyController extends
StrategyController<NQueenNode> {

  public boolean isEnd(Strategy<NQueenNode> strategy) {
    return !strategy.getCurrent().getLast().hasNextRow();
  }
}
```

Kreirani posmatrač pretraživanja vrši ispis putanje po završetku obilaska.

```
package rs.etf.es.search.examples.nqueens;
import rs.etf.es.search.lib.*;

public class NQueenStrategyObserver extends
StrategyObserverImpl<NQueenNode> {

  public void updateSearchFinished(Strategy<NQueenNode> strategy) {
```

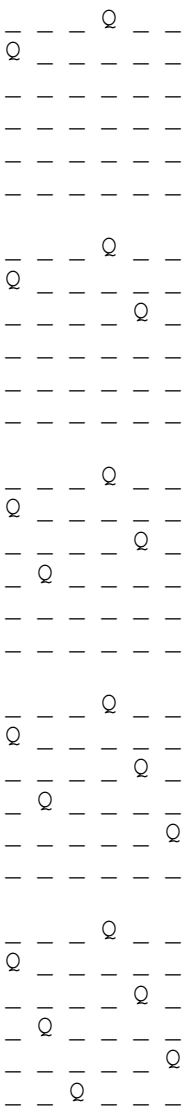
```
    printPath(strategy);
  }

  private void printPath(Strategy<NQueenNode> strategy) {
    Path<NQueenNode> path = strategy.getCurrent();
    for (int i = 0; i < path.size(); i++)
      System.out.print(path.get(i) + "\n");
  }
}
```

Za kraj, kreirana je klasa NQueenMain koja prikazuje upotrebu definisanih klasa za rešavanje problema iz zadatka 20.

```
package rs.etf.es.search.examples.nqueens;
import rs.etf.es.search.algorithms.*;
import rs.etf.es.search.lib.*;

public class NQueenMain {
  private static final int N = 6;

  public static void main(String[] args) {
    StrategyController<NQueenNode> controller =
                  new NQueenStrategyController();
    NQueenNode start = new NQueenNode(N);
    Evaluator<NQueenNode> evaluator = new AscendantEvaluator<NQueenNode>();
    HeuristicCalculator<NQueenNode> calculator =
                  new NQueenHeuristicCalculator();
    Operator<NQueenNode> operator = new NQueenOperator();

    Strategy<NQueenNode> strategy =
                  new HillClimbing<NQueenNode>(start,
                                               controller,
                                               evaluator,
                                               calculator,
                                               operator);

    NQueenStrategyObserver observer = new NQueenStrategyObserver();
    strategy.addObserver(observer);

    strategy.search();
  }
}
```

Pokretanjem poslednje klase dobija se sledeći izlaz:

```
_ _ _ _ _ _
_ _ _ _ _ _
_ _ _ _ _ _
_ _ _ _ _ _
_ _ _ _ _ _
_ _ _ _ _ _

_ _ _ Q _ _
_ _ _ _ _ _
_ _ _ _ _ _
_ _ _ _ _ _
_ _ _ _ _ _
_ _ _ _ _ _
```



Jednostavno je izvršiti izmene kontrolera pretrage i posmatrača kako bi se ispisao ukupni broj rešenja za proizvoljno n. Konkretno za n = 6, postoje četiri rešenja, za n = 8, broj rešenja je 92.

2. Modeli predstavljanja znanja

2.1. Formalna logika

Zadatak 40: Predikati START, END i DUR

Dati su predikati

START(*e*,*t*) - istinit ako je događaj *e* započeo u trenutku *t*,

END(*e*,*t*) - istinit ako se neki događaj *e* završio u trenutku *t*, i

DUR(*e*,*d*) - istinit ako je događaj *e* trajao *d* vremenskih jedinica.

a) Napisati dobro formirane formule (WFF) koje bi omogućile zaključivanje o kraju nekog događaja, bez obzira na to da li je poznata vrednost *t* u predikatu END, ili o početku nekog događaja bez obzira na to da li je poznata vrednost *t* u predikatu START.
b) Definisati novi predikat AFTER(*e1*,*e2*) koji je istinit ako se događaj *e1* desio posle događaja *e2*, koristeći formule iz prethodne tačke.
c) Definisati novi predikat TOK(*e1*,*e2*) koji je istinit ako se događaj *e1* završio u trenutku kada se *e2* dešava.

Rešenje

a) Za zaključivanje o početku događaja koristimo dobro formiranu formulu:

$$\forall e\ \forall t\ \forall d\ \forall s\{\ [\ END(e,t) \wedge DUR(e,d) \wedge ZBIR(s,d,t)\ ] \Rightarrow START(e,s)\ \}$$

Ako se događaj e završio u trenutku t, a znamo da je trajao d vremenskih jedinica, to znači da je ovaj događaj morao da počne u trenutku t-d. Analogno možemo zaključiti o kraju događaja znajući njegovo trajanje i vremenski trenutak početka:

$$\forall e\ \forall t\ \forall d\ \forall s\{\ [\ START(e,t) \wedge DUR(e,d) \wedge RAZLIKA(s,t,d)\ ] \Rightarrow END(e,s)\ \}$$

b) Potrebno je da bude ispunjen uslov da je vremenski trenutak t_2 početka događaja e_2 sledi vremenski trenutak t_1 kraja događaja e_1 (nema preklapanja dešavanja događaja):

$$\forall e_1\ \forall e_2\ \forall t_1\ \forall t_2\ [\ START(e_2,t_2) \wedge END(e_1,t_1) \wedge VEĆE(t_2, t_1) \Rightarrow AFTER(e_2, e_1)\ ]$$

Predikat VEĆE(t_2, t_1) je tačan ako je veće t_2 od t_1.

c) Vremenski trenutak t_1 kada se završava događaj e_1 treba da se nalazi između vremena početka t_2 i kraja t_3 događaja e_2:

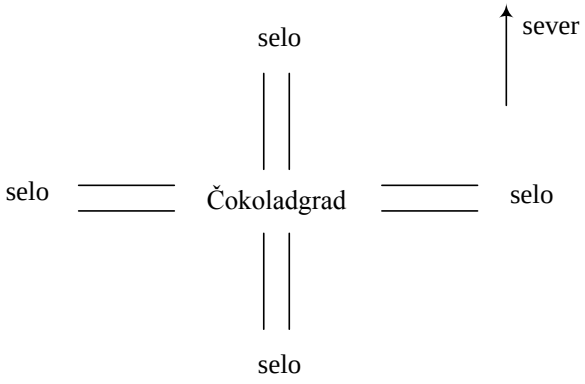
$$\forall e_1\ \forall e_2\ \forall t_1\ \forall t_2\ \forall t_3 \quad [\ END(e_1,t_1) \wedge START(e_2,t_2) \wedge END(e_2,t_3) \wedge IZMEĐU(t_1, t_2, t_3) \Rightarrow TOK(e_1, e_2)\ ]$$

Zadatak 41: Ostrvo uživanja

Čokoladgrad, glavni grad Ostrva uživanja, nalazi se na raskrsnici četiri puta koja postoje na ostrvu. Svaki od puteva vodi na jednu od četiri strane sveta, i ide do jednog od četiri sela koja takođe postoje na ostrvu (slika 99). Poznato je:

1. Selo na Okeanskom putu je za 1 milju dalje od Čokoladgrada nego što je to slučaj sa Sladoled-selom.
2. Najbliže selo udaljeno je 2 milje od Čokoladgrada.
3. Ne postoji par sela koja su podjednako udaljena od Čokoladgrada.
4. Selo Urmašica udaljeno je 6 milja od sela koje se nalazi na kraju Obalskog puta.
5. Selo na Putu Slasti udaljeno je 9 milja od sela Jesenloze.
6. Selo na jugu dvaput je dalje od Čokoladgrada no što je to slučaj sa selom na severu..

Predstaviti ove činjenice u obliku stavova predikatske logike.



Slika 99

Rešenje

Za predstavljanje zadatih činjenica uvešćemo sledeće predikate:

- NA(p,s) je ispunjeno ako je selo s na putu p.
- U_SMERU(d,s) je ispunjeno ako je selo s na strani sveta d.
- JEDNAKO(x,y) označava relacioni operator jednakosti x i y, gde x i y mogu biti ili oznake sela ili razdaljine. Isti operator upotrebićemo, dakle, u različitim kontekstima što se naziva preopterećenje (engl. *overloading*) operatora.
- MANJE(x,y) je ispunjeno ako je razdaljina x manja od razdaljine y.

Funkcija DALJINA(x,y) daje kao rezultat udaljenost mesta x od mesta y.

Činjenice se sada mogu predstaviti na sledeći način:

1. $\forall x$ [NA(Okeanski_put,x) $\Rightarrow$
   JEDNAKO(DALJINA(x, Čokoladgrad), DALJINA(Sladoled_Selo, Čokoladgrad)+1)]
2. $\neg\exists x$ [MANJE(DALJINA(x, Čokoladgrad), 2)]
3. $\forall x \forall y$ [JEDNAKO(DALJINA(x, Čokoladgrad), DALJINA(y, Čokoladgrad)) $\Rightarrow$
   JEDNAKO(x,y)]
4. $\forall x$ [NA(Obalski_put,x) $\Rightarrow$ JEDNAKO(DALJINA(x, Urmašica), 6)]
5. $\forall x$ [NA(Put_slasti,x) $\Rightarrow$ JEDNAKO(DALJINA(x, Jesenloza), 9)]

6. $\forall x\ \forall y$ [U_SMERU(Jug, x) $\wedge$ U_SMERU(Sever,y) $\Rightarrow$

JEDNAKO(DALJINA(x, Čokoladgrad), 2*DALJINA(y, Čokoladgrad))]

Zadatak 42: Svet blokova (interpretacija predikatskih formula)

Svet blokova predstavlja jednu od interpretacija sledećih dobro formiranih formula:

ON(C,A)

ONTABLE(A)

ONTABLE(B)

CLEAR(C)

CLEAR(B)

$(\forall x)$[CLEAR(x) $\Rightarrow \neg(\exists y)$ ON(y,x)]

Naći dve drugačije interpretacije (van sveta blokova) koje zadovoljavaju konjunkciju ovih formula.

Rešenje

Predikatske formule interpretiramo tako što dodelimo određenja značenja (semantiku) pojedinim predikatima, funkcijama, konstantama i promenljivama. Imena predikata, funkcija, konstanti i operatora obično se biraju tako da asociraju na njihovo značenje, ali nas to ne sprečava da formulu koja ima određeno značenje interpretiramo na neki drugi način tako što ćemo definisati nova značenja za pojedine predikate. Istoj formuli možemo odrediti proizvoljan broj različitih interpretacija.

Zadatim formulama možemo dati interpretaciju odnosa među zaposlenima u preduzeću:

- A, B i C su osobe koje rade u istom preduzeću.
- Predikat ON(C,A) označava da je osoba C nadređena osobi A.
- Predikati ONTABLE(A) i ONTABLE(B) označavaju da su osobe A i B izvršioci poslova (nemaju podređenih).
- Predikati CLEAR(C) i CLEAR(B) označavaju da su osobe B i C (su)vlasnici preduzeća.
- Formula $(\forall x)$[CLEAR(x) $\Rightarrow \neg(\exists y)$ ON(y,x)] u ovom slučaju znači da vlasnik preduzeća nema iznad sebe nadređenu osobu.

Navešćemo drugačiju interpretaciju koja se odnosi na matematički pojam skupa:

- A, B i C predstavljaju različite skupove.
- Predikat ON(C,A) označava da je skup C podskup skupa A.
- Predikati ONTABLE(A) i ONTABLE(B) označavaju da su skupovi A i B podskupovi skupa TABLE. Skup TABLE je univerzalan skup, to jest, sadrži sve ostale skupove.
- Predikati CLEAR(C) i CLEAR(B) označavaju da su B i C prazni skupovi.
- Formula $(\forall x)$[CLEAR(x) $\Rightarrow \neg(\exists y)$ ON(y,x)] izražava svojstvo praznog skupa da ne postoji skup koji je podskup praznog skupa.

Čitalac za vežbu može pronaći druge interpretacije zadatih formula.

Zadatak 43: Jovanovi preci

Pretpostavimo da smo činjenicu da je Jovan Petrov otac predstavili sa OTAC(Petar, Jovan), činjenicu da je Milica Petrova majka predikatom MAJKA(Petar, Milica) i činjenicu da je Milan jedan od Petrovih predaka predikatom PREDAK(Petar, Milan).

Napisati dobro formiranu formulu koja treba da predstavi činjenicu:

Svaki Petrov predak je ili njegov otac, ili njegova majka ili jedan od njihovih predaka.

Rešenje

Zadati iskaz može se predstaviti sledećom predikatskom formulom:

$\forall x \{PREDAK(Petar,x) \Rightarrow [OTAC(Petar,x) \vee MAJKA(Petar,x) \vee [\exists y(OTAC(Petar,y) \vee MAJKA(Petar,y)) \wedge PREDAK(y,x)]]\}$

Iskaz $P \Rightarrow Q$ se tumači na sledeći način: Svako P je Q. Drugim rečima, ako je P tačno možemo odmah da utvrdimo da je Q tačno, ali ako P nije tačno ne možemo ništa reći o istinitosnoj vrednosti Q (jer, na primer, može biti nekih drugih pretpostavki koje povlače tačnost Q). Ako je Q tačno, ne možemo ništa reći o istinitosnoj vrednosti P (jer, kao što je malopre rečeno, i neke druge pretpostavke osim P mogu da povlače zaključak Q), dok u slučaju netačnosti Q možemo odmah da utvrdimo netačnost P (poznato je da je $P \Rightarrow Q$ ekvivalentno sa $\neg Q \Rightarrow \neg P$). Prema tome, P je *dovoljan* uslov za Q, a Q je *neophodan* uslov za P.

Označimo sa P činjenicu da je neko (ljudsko biće) x Petrov predak, a sa Q činjenicu da je x ili Petrov otac, ili Petrova majka, ili jedan od njihovih predaka. Zadati iskaz striktno se predstavlja u formi $P \Rightarrow Q$ (što je u rešenju i učinjeno), mada se, poznajući značenja predikata može ustvrditi i $Q \Rightarrow P$. Prema tome, važi ekvivalencija $P \Leftrightarrow Q$ to jest, P je neophodan i dovoljan uslov za Q, a takođe, Q je neophodan i dovoljan uslov za P.

Obratimo pažnju i na složeni uslov da je x ili Petrov otac, ili Petrova majka ili jedan od njihovih predaka. Formulacija ili ... ili odgovara logičkoj operaciji *ekskluzivno ili* u oznaci $\oplus$. Vrednost izraza $P \oplus Q$ je tačna ako je tačno jedan od iskaza P i Q tačan (za razliku od običnog ILI koje je tačno i u slučaju kada su i P i Q tačni). U rešenju bi, striktno rečeno, bila potrebna operacija ekskluzivno ILI, međutim, uzimajući u obzir semantička ograničenja (da na primer ista osoba ne može biti i Petrov otac i Petrova majka), moguće je upotrebiti običnu ILI operaciju.

Zadatak 44: Inteligencija računarskog sistema

Predstaviti sledeću sentencu preko dobro formiranih formula predikatskog računa:

a) Računarski sistem je inteligentan ako može da obavi zadatak koji, ako ga obavlja čovek, zahteva inteligenciju.

b) Ako je problem pretraživanja komutativan tada, za bilo koje stanje pretrage S, svaki član skupa operatora primenljivih na stanje S je takođe primenljiv na bilo koje stanje dobijeno primenom nekog primenljivog operatora na S.

Rešenje

a) Čitanjem iskaza možemo definisati sledeće predikate:

- ČOVEK(c) koji je tačan ako je c čovek
- SISTEM(s) je ispunjeno ako s predstavlja računarski sistem
- ZADATAK(x) koji je tačan ako je z zadatak
- OBAVLJA(z,x) tačan je ako zadatak z obavlja x, gde x može biti čovek ili računarski sistem.
- ZAHTEVA_INTELIGENCIJU(z,c) je ispunjeno ako zadatak z zahteva inteligenciju od čoveka c da bi ga obavio
- INTELIGENTAN(s) je ispunjeno ako računarski sistem s poseduje inteligenciju.

Sada se dati iskaz može predstaviti sa

$$\forall s\ \exists z\{\ SISTEM(s) \wedge ZADATAK(z) \wedge OBAVLJA(z,s) \wedge$$
$$\forall c[ČOVEK(c) \wedge OBAVLJA(z,c) \Rightarrow ZAHTEVA\_INTELIGENCIJU(z,c)]$$
$$\Rightarrow INTELIGENTAN(s)\}$$

b) Da bismo predstavili zadati iskaz, koristićemo sledeće predikate:

- PROBLEM(p) je ispunjen u slučaju da p predstavlja neki problem pretraživanja
- OPERATOR(o,p) je ispunjen u slučaju da o predstavlja neki operator promene stanja u problemu p
- KOMUTATIVAN(p) je ispunjen ako je problem p komutativan
- STANJE(s,p) važi ako s predstavlja neko stanje u problemu p
- PRIMENLJIV(o,s) je ispunjeno ako je operator o primenjiv na stanje s

Definišemo i jednu funkciju:

- NovoStanje(s,o) vraća kao rezultat naslednika stanja s za operator o.

Dati iskaz može se predstaviti sledećom formulom:

$$PROBLEM(p) \wedge$$
$$[STANJE(s,p) \wedge OPERATOR(o,p) \wedge PRIMENLJIV(o,s) \wedge$$
$$OPERATOR(o1,p) \wedge PRIMENLJIV(o1,s)$$
$$\Rightarrow PRIMENLJIV(o,NovoStanje(s,o1))]$$
$$\Rightarrow KOMUTATIVAN(p)$$

Podrazumeva se da su sve promenljive u gornjoj formuli univerzalno kvantifikovane.

Iskazi dati govornim jezikom često mogu da se interpretiraju na različite načine. Na primer, u rešenju pod a) podrazumevano je da su svi ljudi inteligentni, to jest da mogu da reše svaki

zadatak pri čemu za neke zadatke moraju, a za neke ne da upotrebe inteligenciju. Međutim, može se usvojiti i drugačija pretpostavka: neki ljudi su inteligentni, a neki nisu i postoje zadaci koje samo inteligentni ljudi mogu da reše. Čitaocu se ostavlja da iskaz zadat u tački a) predstavi predikatskom formulom u slučaju alternativnog tumačenja ljudske inteligencije.

Zadatak 45: Nalaženje konjuktivne normalne forme

Odrediti konjuktivnu normalnu formu za sledeću formulu:

$$\forall x \{ \text{Cigla}(x) \Rightarrow \{ \exists y [\text{Na}(x,y) \wedge \neg\text{Piramida}(y)] \wedge \neg\exists y [\text{Na}(x,y) \wedge \text{Na}(y,x)] \wedge$$

$$\wedge \forall y [\neg\text{Cigla}(y) \Rightarrow \neg\text{Jednako}(x,y)] \}\}$$

Rešenje

Svaka dobro formirana formula može se dovesti u *konjuktivnu normalnu formu* (KNF), to jest, biti predstavljena nizom *klauzula* pri čemu se između pojedinih klauzula podrazumeva operator konjukcije. Klauzula je niz literala povezanih disjunkcijom. Postupak transformacije formule u KNF sastoji se iz niza koraka:

1. Eliminisanje implikacija ($E_1 \Rightarrow E_2$ transformiše se u $\neg E_1 \vee E_2$)

Sledi

$$\forall x [\neg\text{Cigla}(x) \vee (\exists y [\text{Na}(x,y) \wedge \neg\text{Piramida}(y)] \wedge \neg\exists y [\text{Na}(x,y) \wedge \text{Na}(y,x)] \wedge$$

$$\wedge \forall y [\neg(\neg\text{Cigla}(y)) \vee \neg\text{Jednako}(x,y)])]$$

2. 'Spuštanje' negacija do atomskih formula

($\neg(E_1 \wedge E_2)$ transformiše se u $\neg E_1 \vee \neg E_2$,

$\neg(E_1 \vee E_2)$ transformiše se u $\neg E_1 \wedge \neg E_2$,

$\neg(\neg E_1)$ transformiše se u E_1 ,

$\neg\forall x [E_1(x)]$ transformiše se u $\exists x [\neg E_1(x)]$,

$\neg\exists x [E_1(x)]$ transformiše se u $\forall x [\neg E_1(x)]$)

Sledi

$$\forall x [\neg\text{Cigla}(x) \vee (\exists y [\text{Na}(x,y) \wedge \neg\text{Piramida}(y)] \wedge \forall y [\neg\text{Na}(x,y) \vee \neg\text{Na}(y,x)] \wedge$$

$$\wedge \forall y [\text{Cigla}(y) \vee \neg\text{Jednako}(x,y)])]$$

3. Uklanjanje egzistencijalnih kvantifikatora

Posmatrajmo izraz

$$\forall x \exists y [\text{Na}(x,y) \wedge \neg\text{Piramida}(y)].$$

Za svaku vrednost x uvek se može naći neka vrednost y takva da formula važi. Drugim rečima, postoji funkcija Φ koja (nije bitno na koji način) za svaku vrednost x daje odgovarajuću vrednost y. Sada posmatranu formulu možemo da zamenimo sledećom:

$$\text{Na}(x,\Phi(x)) \wedge \neg\text{Piramida}(\Phi(x))$$

Egzistencijalni kvantifikator više se ne javlja u formuli. Funkcije uvedene radi zamene egzistencijalnih kvantifikatora zovu se Skolemove funkcije, po skandinavskom matematičaru koji ih je prvi uveo. U gornjem primeru, funkcija Φ po prirodi stvari zavisi od x. Generalno, argumenti funkcije su sve promenljive koje su vezane univerzalnim kvantifikatorom na onom mestu u formuli na kome se pojavljuje član ∃y.

Ako Skolemovu funkciju u problemu koji razmatramo nazovemo Drži, imaćemo:

$$\forall x\ [\ \neg Cigla(x) \vee (\ Na(x,Drži(x)) \wedge \neg Piramida(Drži(x)) \wedge \forall y\ [\ \neg Na(x,y) \vee \neg Na(y,x)\ ]$$
$$\wedge\ \forall y\ [Cigla(y) \vee \neg Jednako(x,y)\ ])]$$

4. Preimenovanje promenljivih tako da svakom kvantifikatoru odgovara posebna promenljiva (ovo je priprema za sledeći korak)

$$\forall x\ [\ \neg Cigla(x) \vee (\ Na(x,Drži(x)) \wedge \neg Piramida(Drži(x)) \wedge \forall y\ [\ \neg Na(x,y) \vee \neg Na(y,x)\ ]$$
$$\wedge\ \forall z\ [Cigla(z) \vee \neg Jednako(x,z)\ ])]$$

5. Premeštanje svih univerzalnih kvantifikatora na levu stranu bez promene njihovog redosleda

$$\forall x\ \forall y\ \forall z\ [\ \neg Cigla(x) \vee (\ Na(x,Drži(x)) \wedge \neg Piramida(Drži(x)) \wedge$$
$$\wedge\ [\ \neg Na(x,y) \vee \neg Na(y,x)\ ] \wedge [Cigla(z) \vee \neg Jednako(x,z)\ ])]$$

6. 'Spuštanje" disjunkcija do najnižeg nivoa (prema zakonu distribucije ∨ u odnosu na ∧

$$(\ E1 \wedge E2\ ) \vee E3 \text{ transformiše se u } (\ E1 \vee E3\ ) \wedge (\ E2 \vee E3\ )\ )$$

Sledi

$$\forall x\ \forall y\ \forall z\ [\ (\ \neg Cigla(x) \vee Na(x,Drži(x))) \wedge (\ \neg Cigla(x) \vee \neg Piramida(Drži(x))) \wedge$$
$$\wedge\ (\ \neg Cigla(x) \vee \neg Na(x,y) \vee \neg Na(y,x)) \wedge$$
$$\wedge\ (\ \neg Cigla(x) \vee Cigla(z) \vee \neg Jednako(x,z))\ ]$$

7. Eliminacija konjukcija (svaki član treba napisati kao zasebnu formulu)

$$\forall x\ [\ \neg Cigla(x) \vee Na(x,Drži(x))\ ]$$

$$\forall x\ [\ \neg Cigla(x) \vee \neg Piramida(Drži(x))\ ]$$

$$\forall x\ \forall y\ [\ \neg Cigla(x) \vee \neg Na(x,y) \vee \neg Na(y,x)\ ]$$

$$\forall x\ \forall z\ [\ \neg Cigla(x) \vee Cigla(z) \vee \neg Jednako(x,z))\ ]$$

8. Preimenovanje promenljivih tako da ne postoji ista promenljiva u različitim formulama

$$\forall x\ [\ \neg Cigla(x) \vee Na(x,Drži(x))\ ]$$

$$\forall u\ [\ \neg Cigla(u) \vee \neg Piramida(Drži(u))\ ]$$

$$\forall v\ \forall y\ [\ \neg Cigla(v) \vee \neg Na(v,y) \vee \neg Na(y,v)\ ]$$

$$\forall w\ \forall z\ [\ \neg Cigla(w) \vee Cigla(z) \vee \neg Jednako(w,z))\ ]$$

9. Uklanjanje kvantifikatora

$$\neg Cigla(x) \vee Na(x,Drži(x))$$

$$\neg Cigla(u) \vee \neg Piramida(Drži(u))$$

$\neg$Cigla(v) $\vee$ $\neg$Na(v,y) $\vee$ $\neg$Na(y,v)

$\neg$Cigla(w) $\vee$ Cigla(z) $\vee$ $\neg$Jednako(w,z))

Zaključno sa ovim korakom završen je postupak transformacije formule u KNF. Zavisno od složenosti formule, pojedini koraci mogu se preskočiti ili objediniti.

Zadatak 46: Saša i kikiriki (zaključivanje rezolucijom)

Dati su iskazi:

1. Saša voli sve vrste hrane.
2. Jabuke su hrana.
3. Piletina je hrana.
4. Hrana je sve ono što neko jede i ne otruje se.
5. Srđan jede kikiriki i još je živ.
6. Ceca jede sve što Srđan jede.

a) Pretvoriti iskaze u dobro formirane formule predikatske logike.

b) Odrediti konjuktivnu normalnu formu za formule dobijene pod a).

c) Rezolucijom pokazati da Saša voli kikiriki.

Rešenje

a) Potrebno je najpre definisati predikate sa značenjem vezanim za kontekst problema. Predikatima predstavljamo osobine objekata. Na primer, predikat HRANA(x) je istinit ako je objekat x neka vrsta hrane. Vrednost ovog predikata definiše se formulom 4. Konkretne objekte identifikujemo konstantama pa ih pišemo velikim slovom, na primer.: JABUKA, PILETINA, KIKIRIKI su pojedine vrste hrane, a SRĐAN, CECA su konkretne osobe.

Predikatima takođe iskazujemo i relacije među objektima: predikat VOLI(x,y) označava da (osoba) x voli (hranu) y; predikat OTRUJE_SE(x,y) je tačan ako se osoba x otrovala hranom y; predikat JEDE(x,y) je tačan ako osoba x jede hranu y.

1. $\forall$x [HRANA(x) $\Rightarrow$ VOLI (SAŠA, x)]
2. HRANA(JABUKA)
3. HRANA(PILETINA)
4. $\forall$x $\forall$y [JEDE(y,x) $\wedge$ $\neg$OTRUJE_SE(y,x) $\Rightarrow$ HRANA(x)]
5. JEDE(SRĐAN, KIKIRIKI) $\wedge$ $\neg$OTRUJE_SE(SRĐAN, KIKIRIKI)
6. $\forall$x [JEDE(SRĐAN,x) $\Rightarrow$ JEDE(CECA,x)]

b) Prevođenje formula predikatske logike u konjuktivnu normalnu formu je neophodan postupak za primenu pravila rezolucije.

1. $\neg$HRANA(x) $\vee$ VOLI (SAŠA, x)

2. HRANA(JABUKA)

3. HRANA(PILETINA)

4. $\neg$JEDE(y,x_1) $\vee$ OTRUJE_SE(y,x_1) $\vee$

$\vee$ HRANA(x_1)

5'. JEDE(SRĐAN, KIKIRIKI)

5". $\neg$OTRUJE_SE(SRĐAN, KIKIRIKI)

6. $\neg$JEDE(SRĐAN,x_2) $\vee$ JEDE(CECA,x_2)

c) Rezolucija je jedno od pravila izvođenja u predikatskom računu pri čemu se kombinacijom dve klauzule oblika P $\vee$ Q i $\neg$P $\vee$ R dobija nova klauzula Q $\vee$ R. Simbolički se ovo označava kao:

$$P \vee Q, \neg P \vee R \rightarrow Q \vee R$$

Pri tome klauzule mogu imati više od dva literala koji se svi (osim literala P i $\neg$P) pojavljuju u rezultantnoj formuli. Dokazivanje primenom rezolucije svodi se na dodavanje negacije tvrđenja hipotezama i pokušaj ustanovljavanja protivrečnosti tako formiranog sistema stavova. Protivrečnost je utvrđena dobijanjem klauzule bez literala koju označavamo sa NIL.

Stavovima iz tačke b) dodajemo negaciju tvrđenja iz tačke c):

7. $\neg$VOLI (SAŠA, KIKIRIKI)

iz 4. i 5'., unifikacijom promenljivih x_1 = KIKIRIKI i y=SRĐAN i primenom rezolucije dobija se:

8. OTRUJE_SE(SRĐAN,KIKIRIKI) $\vee$ HRANA(KIKIRIKI)

iz 8. i 5". rezolucijom se dobija

9. HRANA(KIKIRIKI)

iz 1. i 9., stavljajući x = KIKIRIKI, rezolucijom se dobija

10. VOLI (SAŠA, KIKIRIKI)

iz stavova 7. i 10. dobija se prazna klauzula NIL. Prema tome, dobijena je protivrečnost u skupu stavova koji sadrži polazne stavove i negaciju tvrđenja, čime je utvrđeno da je polazno tvrđenje tačno.

Zadatak 47: Rodbinske veze

Date su sledeće tvrdnje:

1. Ako je osoba X brat osobe Z i osoba Y takođe brat osobe Z, onda je osoba X brat i osobe Y ili su X i Y ista osoba.
2. Ako je osoba X muško i ima istu majku kao i osoba Y, onda je X brat osobe Y ili su X i Y ista osoba.
3. Marija je majka Milana i Ane.
4. Milan je muško.
5. Jovan je Anin brat.

6. Milan i Jovan nisu ista osoba.
7. Milan i Ana nisu ista osoba.

a) Napisati formule predikatskog računa i prevesti ih u klauzalni oblik.

b) Rezolucijom dokazati ili pobiti tvrdnju da je Milan Jovanov brat.

Rešenje

a) Činjenice se mogu predstaviti predikatskim formulama na sledeći način:

1. $\forall x\ \forall y\ \forall z\ [\ Brat(x,z) \wedge Brat(y,z) \Rightarrow Brat(x,y) \vee Ista\_osoba(x,y)\ ]$

2. $\forall x\ \forall y\ \forall z\ [\ Muško(x) \wedge Majka(z,x) \wedge Majka(z,y) \Rightarrow Brat(x,y) \vee Ista\_osoba(x,y)\ ]$

3. $Majka(Marija, Milan) \wedge Majka(Marija, Ana)$

4. $Muško(Milan)$

5. $Brat(Jovan, Ana)$

6. $\neg Ista\_osoba(Milan, Jovan)$

7. $\neg Ista\_osoba(Milan, Ana)$

Sva tvrđenja osim tvrđenja 1. i 2. se već nalaze u klauzalnoj formi pošto se radi o literalima. Tvrđenja 1. i 2. dovode se u klauzalnu formu uklanjanjem implikacije i primenom DeMorganovog zakona uz preimenovanje promenljivih druge formule radi jednoznačnosti.

1. $\neg Brat(x,z) \vee \neg Brat(y,z) \vee Brat(x,y) \vee Ista\_osoba(x,y)$

2. $\neg Muško(u) \vee \neg Majka(w,u) \vee \neg Majka(w,v) \vee Brat(u,v) \vee Ista\_osoba(u,v)$

3'. $Majka(Marija, Milan)$

3". $Majka(Marija, Ana)$

4. $Muško(Milan)$

5. $Brat(Jovan, Ana)$

6. $\neg Ista\_osoba(Milan, Jovan)$

7. $\neg Ista\_osoba(Milan, Ana)$

b) Prethodnim tvrdnjama dodajemo negaciju pretpostavke da je Milan Jovanov brat:

8. $\neg Brat(Milan, Jovan)$

i tražimo protivrečnost primenjujući pravilo rezolucije. S obzirom da ima dosta stavova, potrebno je usvojiti neku strategiju za izbor dva stava na koje primenjujemo rezoluciju u svakom koraku zaključivanja. U ovom slučaju primenićemo strategiju *prvenstva jedinice* (engl. *unit preference*) prema kojoj se prioritet pri izboru daje stavovima sa najmanjim brojem članova. U ovom slučaju između stavova 3' do 8 proizvoljno biramo stav 4 koji jedino može da se upari sa stavom 2:

$$2., 4. \xrightarrow{u=Milan} 9.\ \neg Majka(w,Milan) \vee \neg Majka(w,v) \vee Brat(Milan,v) \vee Ista\_osoba(Milan,v)$$

U nastavku biramo stav 3'. Ovaj stav može da se upari sa stavovima 2 i 9. Od ova dva stava biramo 9 jer ima manje članova i primenjujemo rezoluciju:

3'., 9. $\xrightarrow{w=Marija}$ 10. $\neg$Majka(Marija,v) $\vee$ Brat(Milan,v) $\vee$ Ista_osoba(Milan,v)

Stav 3" možemo upariti sa stavovima 2, 9 i 10 od kojih biramo stav 10 kao najkraći:

3"., 10. $\xrightarrow{v=Ana}$ 11. Brat(Milan,Ana) $\vee$ Ista_osoba(Milan,Ana)

Stav 7 uparujemo sa stavom 11 kao najkraćim od stavova 1, 2, 9, 10 i 11.

7., 11. $\longrightarrow$ 12. Brat(Milan,Ana)

Dobijeni stav 12 biramo sledeći. Ovaj stav može se upariti jedino sa stavom 1.

1., 12. $\xrightarrow{x=Milan,\ z=Ana}$ 13. $\neg$Brat(y,Ana) $\vee$ Brat(Milan,y) $\vee$ Ista_osoba(Milan,y)

Od neupotrebljenih stavova sa jednim predikatom ostali su još 5 i 6. Stav 5 može se upariti sa stavovima 1 i 13 pri čemu biramo 13 jer ima manje članova.

5., 13. $\xrightarrow{y=Jovan}$ 14. Brat(Milan,Jovan) $\vee$ Ista_osoba(Milan, Jovan)

Stav 6. može se upariti sa stavovima 1, 2, 9, 11, 13 i 14. Od ovih stavova biramo stav 14 jer ima samo dva člana.

6., 14. $\longrightarrow$ 15. Brat(Milan,Jovan)

Novi stav 15 može se upariti sa stavovima 1 i 8 od kojih izbor pada na stav 1.

8., 15. $\longrightarrow$ NIL

što znači da je pretpostavka tačna.

Zadatak 48: Kriminalci i njihovi zločini

Dati su sledeći stavovi:

1. Za svaki zločin postoji počinilac.
2. Samo kriminalci čine zločine.
3. Samo kriminalci bivaju uhapšeni.
4. Uhapšeni kriminalci ne čine zločine.
5. Zločini se i dalje čine.

Primenom rezolucije dokazati stav "Ima kriminalaca koji nisu uhapšeni".

Rešenje

Potrebno je date stavove prevesti u formule predikatske logike:

1. $\forall x \{ \text{Zločin}(x) \Rightarrow \exists y [\text{Počinilac}(x,y)] \}$

2. $\forall z \forall u [\text{Zločin}(z) \wedge \text{Počinilac}(z,u) \Rightarrow \text{Kriminalac}(u)]$

3. $\forall v [\text{Uhapšen}(v) \Rightarrow \text{Kriminalac}(v)]$

4. $\forall w \{ \text{Kriminalac}(w) \wedge \text{Uhapšen}(w) \Rightarrow \neg\exists a [\text{Zločin}(a) \wedge \text{Počinilac}(a,w)] \}$

5. $\exists b$ [Zločin(b)]

Pri prevođenju sa govornog jezika treba obratiti pažnju da se očuva smisao iskaza. Na primer, stav 3. glasi da je činjenica da je osoba kriminalac neophodan uslov da bi ona bila uhapšena. Da je ovo predstavljeno kao:

3. $\forall v$ [Kriminalac(v) $\Rightarrow$ Uhapšen(v)]

to bi značilo da su svi kriminalci uhapšeni, ali bi dopuštalo mogućnost da je pri tome uhapšen i poneki čestit građanin.

Ovim stavovima dodajemo negaciju tvrđenja kao pripremu za sprovođenje procedure dokazivanja tvrđenja:

6. $\neg\exists c$ [Kriminalac(c) $\wedge$ $\neg$Uhapšen(c)]

Pre primene rezolucije potrebno je stavove prevesti u konjuktivnu normalnu formu:

1. $\neg$Zločin(x) $\vee$ Počinilac(x, Φ(x))

2. $\neg$Zločin(z) $\vee$ $\neg$Počinilac(z, u) $\vee$ Kriminalac(u)

3. $\neg$Uhapšen(v) $\vee$ Kriminalac(v)

4. $\neg$Kriminalac(w) $\vee$ $\neg$Uhapšen(w) $\vee$ $\neg$Zločin(a) $\vee$ $\neg$Počinilac(a,w)

5. Zločin(B)

6. $\neg$Kriminalac(c) $\vee$ Uhapšen(c)

Primenom rezolucije na stavove 3. i 6. odmah se dobija stav NIL čime je dokaz gotov. Dokaz se može sprovesti i bez korišćenja stava 3.:

1., 5. $\xrightarrow{x = B}$ 7. Počinilac(B,Φ(B))

2., 7. $\xrightarrow{z = B,\ u = \Phi(B)}$ 8. $\neg$Zločin(B) $\vee$ Kriminalac(Φ(B))

5., 8. $\longrightarrow$ 9. Kriminalac(Φ(B))

4.,7. $\xrightarrow{a = B,\ w = \Phi(B)}$ 10. $\neg$Kriminalac(Φ(B)) $\vee$ $\neg$Uhapšen(Φ(B)) $\vee$ $\neg$Zločin(B)

9., 10. $\longrightarrow$ 11. $\neg$Uhapšen(Φ(B)) $\vee$ $\neg$Zločin(B)

5., 11. $\longrightarrow$ 12. $\neg$Uhapšen(Φ(B))

6., 12. $\xrightarrow{c = \Phi(B)}$ 13. $\neg$Kriminalac(Φ(B))

9., 13. $\longrightarrow$ NIL

Zadatak 49: Perica i Chop Suey

Poznate su činjenice:

1. Perica voli sva laka jela.
2. Jela francuske kuhinje su teška.

3. Jela kineske kuhinje su laka.

4. Chop Suey je jelo kineske kuhinje.

Koristeći rezoluciju odgovoriti na pitanje: koje jelo voli Perica?

Rešenje

Prevodimo iskaze u stavove predikatske logike:

1. $\forall x$ [Lako_jelo(x) $\Rightarrow$ Perica_voli(x)]

2. $\forall y$ [Francusko_jelo(y) $\Rightarrow \neg$Lako_jelo(y)]

3. $\forall z$ [Kinesko_jelo(z) $\Rightarrow$ Lako_jelo(z)]

4. Kinesko_jelo(Chop_Suey)

Dokazaćemo da postoji jelo koje Perica voli, to jest

$\exists x$ [Perica_voli(x)]

Zbog toga ćemo gornjim stavovima dodati negaciju tvrđenja:

5. $\neg\exists x$ [Perica_voli(x)]

Prevedimo iskaze u konjuktivnu normalnu formu:

1. $\neg$Lako_jelo(x) $\vee$ Perica_voli(x)

2. $\neg$Francusko_jelo(y) $\vee \neg$Lako_jelo(y)

3. $\neg$Kinesko_jelo(z) $\vee$ Lako_jelo(z)

4. Kinesko_jelo(Chop_Suey)

5. $\neg$Perica_voli(w)

Sada rezolucijom tražimo protivrečnost. Primenićemo strategiju *skupa podrške* (engl. *set-of-support*) pri izboru stavova za spajanje, koja se sastoji u tome da se za spajanje uvek prvi biraju oni stavovi koji predstavljaju ili negaciju tvrđenja ili stavove izvedene iz negacije tvrđenja. Razmatramo, prema tome, redom stavove 1 do 4 radi spajanja sa stavom 5; jedina moguća kombinacija je:

1., 5. $\xrightarrow{x = w}$ 6. $\neg$Lako_jelo(w)

Pošto su isprobane sve kombinacije stava 5 sa ostalim početnim stavovima, razmatramo kombinacije stava 6 sa ostalim stavovima. Jedina moguća kombinacija je:

3., 6. $\xrightarrow{z = w}$ 7. $\neg$Kinesko_jelo(w)

Najzad, razmatranjem kombinacija stava 7. sa ostalim stavovima, jedina moguća kombinacija

4., 7. $\xrightarrow{w = Chop\_Suey}$ NIL

Dokazano je dakle, da Perica voli Chop_Suey.

Pri prevođenju pretpostavki u predikatske formule treba obratiti pažnju da iskaz 1. znači da činjenica da je neko jelo lako povlači zaključak da Perica voli to jelo, a ne obrnuto. Drugim rečima, formula

$\forall x$ [Perica_voli(x) $\Rightarrow$ Lako_jelo(x)]

ne bi predstavljala korektan prevod stava 1. Poslednja formula može se prevesti iskazom: Perica voli SAMO laka jela (ali ne obavezno SVA laka jela).

Čitaocu se preporučuje da pokuša rešenje pronaći primenom neke od alternativnih strategija izbora stavova za rezoluciju.

Zadatak 50: Svet blokova (zaključivanje rezolucijom)

Dati su sledeći stavovi iz sveta blokova:

- Blok *A* nalazi se na stolu.
- Blok *A* je plave boje.
- Blok *B* nalazi se na bloku A.
- Blok *B* je plave boje.
- Blok *C* nalazi se na stolu.
- Blok *C* je crvene boje.
- Blok *D* nalazi se na bloku *B*.
- Blok *D* je zelene boje.
- Blok *X* je ispod nekog drugog bloka *Y* ako se *Y* nalazi na X, ili se *X* nalazi ispod bloka koji je ispod bloka *Y*.

Prethodne stavove formulisati u obliku stavova predikatske logike, zatim primenom rezolucije uz strategiju skupa podrške (*set-of-support)* dokazati stav: blok *A* nalazi se ispod zelenog bloka.

Rešenje

Formulacija pretpostavki u obliku predikatskih formula:

1. Blok(A)
2. Blok(B)
3. Blok(C)
4. Blok(D)
5. Plav(A)
6. Plav(B)
7. Crven(C)
8. Zelen(D)
9. Na(A,Sto)
10. Na(B,A)
11. Na(C,Sto)
12. Na(D,B)
13. $\forall x \; \forall y$ [Blok(x) $\wedge$ Blok(y) $\wedge$ Na(y,x) $\Rightarrow$ Ispod(x,y)]
14. $\forall x \; \forall y$ [Blok(x) $\wedge$ Blok(y) $\wedge \exists z$ (Blok(z) $\wedge$ Ispod(x,z) $\wedge$ Ispod(z,y)) $\Rightarrow$ Ispod(x,y)]

Prethodnim stavovima dodajemo negaciju tvrđenja:

15. $\neg\exists z$ [Blok(z) $\wedge$ Zeleno(z) $\wedge$ Ispod(A,z))

Stavove 13., 14. i 15. potrebno je prevesti u konjuktivnu normalnu formu:

13. $\neg$Blok(x) $\vee$ $\neg$Blok(y) $\vee$ $\neg$Na(y,x) $\vee$ Ispod(x,y)

14. $\neg$Blok(u) $\vee$ $\neg$Blok(v) $\vee$ $\neg$Blok(w) $\vee$ $\neg$Ispod(u,w) $\vee$ $\neg$Ispod(w,v) $\vee$ Ispod(u,v)

15. $\neg$Blok(x) $\vee$ $\neg$Zeleno(z) $\vee$ $\neg$Ispod(A,z)

Problem izbora stavova za spajanje pri rezoluciji je značajan jer u slučaju postojanja velikog broja polaznih stavova dolazi do kombinatorne eksplozije ukoliko se pokušaju napraviti sve moguće kombinacije stavova (primena rezolucije takođe generiše nove stavove). Sa druge strane, ukoliko redukujemo kombinatoriku spajanja stavova rizikujemo da ne otkrijemo pravi redosled spajanja. *Skup podrške* je takva strategija izbora stavova kod koje se za spajanje uvek biraju prvo oni stavovi koji predstavljaju ili negaciju tvrđenja ili stavove izvedene u nekom od prethodnih koraka primene rezolucije. Rezolucija uz *set-of-support* strategiju:

8., 15. $\xrightarrow{z = D}$ 16. $\neg$Blok(D) $\vee$ $\neg$Ispod(A,D)

4., 16. $\longrightarrow$ 17. $\neg$Ispod(A,D)

14., 17. $\xrightarrow{u=A,\ v=D}$ 18. $\neg$Blok(A) $\vee$ $\neg$Blok(D) $\vee$ $\neg$Blok(w) $\vee$ $\neg$Ispod(A,w) $\vee$ $\neg$Ispod(w,D)

1., 18. $\longrightarrow$ 19. $\neg$Blok(D) $\vee$ $\neg$Blok(w) $\vee$ $\neg$Ispod(A,w) $\vee$ $\neg$Ispod(w,D)

4., 19. $\longrightarrow$ 20. $\neg$Blok(w) $\vee$ $\neg$Ispod(A,w) $\vee$ $\neg$Ispod(w,D)

2., 20. $\xrightarrow{w=B}$ 21. $\neg$Ispod(A,B) $\vee$ $\neg$Ispod(B,D)

13., 21. $\xrightarrow{x=A,\ y=B}$ 22. $\neg$Blok(A) $\vee$ $\neg$Blok(B) $\vee$ $\neg$Na(B,A) $\vee$ $\neg$Ispod(B,D)

1., 22. $\longrightarrow$ 23. $\neg$Blok(B) $\vee$ $\neg$Na(B,A) $\vee$ $\neg$Ispod(B,D)

2., 23. $\longrightarrow$ 24. $\neg$Na(B,A) $\vee$ $\neg$Ispod(B,D)

24., 10. $\longrightarrow$ 25. $\neg$Ispod(B,D)

13., 25. $\xrightarrow{x=B,\ y=D}$ 26. $\neg$Blok(B) $\vee$ $\neg$Blok(D) $\vee$ $\neg$Na(D,B)

2., 26. $\longrightarrow$ 27. $\neg$Blok(D) $\vee$ $\neg$Na(D,B)

4., 17. $\longrightarrow$ 28. $\neg$Na(D,B)

12., 28. $\longrightarrow$ NIL

Zadatak 51: Rezolucija uz izbor stavova po širini

a) Sledeće WFF transformisati u konjuktivnu normalnu

formu (CNF):

1. $\forall x \forall y \forall s$ [C(x,s) $\wedge$ C(y,s) $\Rightarrow$ O(x,y,P(x,y,s))]

2. $\forall x \forall y \forall s$ [O(x,y,s) $\Rightarrow \neg$ C(y,s)]

b) Poznate su činjenice:

1. $P \Rightarrow R$

2. $Q \Rightarrow R$

3. $P \vee Q$

Pokazati da je R teorema, odnosno da R sledi iz prethodnih činjenica, koristeći rezoluciju uz strategiju izbora stavova *po širini*.

Rešenje

a)

1. $\neg C(x_1,s_1) \vee \neg C(y_1,s_1) \vee O(x_1,y_1,P(x_1,y_1,s_1))$

2. $\neg O(x_2,y_2,s_2) \vee \neg\, C(y_2,s_2)$

b) Činjenice u konjuktivnoj normalnoj formi:

1. $\neg P \vee R$

2. $\neg Q \vee R$

3. $P \vee Q$

Dodajemo negaciju teoreme:

4. $\neg R$

Rezolucijom tražimo protivrečnost. Za izbor stavova za primenu rezolucije primenićemo strategiju izbora *po širini* (engl. *breadth-first*), koja se sastoji u tome da se razmotre redom sve moguće kombinacije postojećih stavova pre nego što se pređe na novodobijene stavove.

Stav 1 može se redom kombinovati sa stavovima 3 i 4 a ne može sa stavom 2:

1., 3. $\longrightarrow$ 5. $Q \vee R$

1., 4. $\longrightarrow$ 6. $\neg P$

Stav 2 može se redom kombinovati sa stavovima 3 i 4:

2., 3. $\longrightarrow$ 7. $P \vee R$

2., 4. $\longrightarrow$ 8. $\neg Q$

Stavovi 3 i 4 ne mogu se kombinovati. Sada se razmatraju kombinacije stava 5 sa stavovima 1 do 4. Moguće kombinacije su:

2., 5. $\longrightarrow$ 9. $R \vee R$, što se elementarnom transformacijom svodi na R

4., 5. $\longrightarrow$ 10. Q

Razmatranjem kombinacija stava 6 sa stavovima 1 do 5, zaključujemo da je jedina moguća kombinacija:

3., 6. $\longrightarrow$ 11. Q

Ovaj stav identičan je već dobijenom stavu 10. U sistem za zaključivanje može se ugraditi detekcija ovakvih situacija da bi se izbeglo nepotrebno razmatranje stava 11 u narednim kombinacijama; međutim, treba voditi računa o tome da sprovođenje detekcije takođe zahteva određeno vreme pri zaključivanju.

Sledi razmatranje kombinacija stava 7 sa stavovima 1 do 6. Zaključujemo da su moguće kombinacije:

1., 7. $\longrightarrow$ 12. P

4., 7. $\longrightarrow$ 13. P

6., 7. $\longrightarrow$ 14. R

Sledi razmatranje kombinacija stava 8 sa stavovima 1 do 7. Zaključujemo da su moguće kombinacije:

3., 8. $\longrightarrow$ 15. P

5., 8. $\longrightarrow$ 16. R

Sledi razmatranje kombinacija stava 9 sa stavovima 1 do 8. Zaključujemo da su moguće kombinacije:

4., 9. $\longrightarrow$ NIL

čime je polazna pretpostavka dokazana.

Ovaj primer jasno ilustruje problem kombinatorne eksplozije pri zaključivanju primenom rezolucije: broj mogućih kombinacija stavova za spajanje je eksponencijalna funkcija broja polaznih stavova i članova u tim stavovima. U ovom slučaju, bilo je potrebno 13 koraka da bi se za 4 polazna jednostavna stava našlo konačno rešenje primenom taktike spajanja stavova po širini. Ova taktika je 'neinteligentna' a njen je cilj da se ne preskoči nijedna od mogućih kombinacija stavova. Za isti primer, moguće je naći rešenje u samo 3 koraka:

2., 3. $\longrightarrow$ '5. $P \vee R$

1., 5. $\longrightarrow$ '6. R

4., 6. $\longrightarrow$ NIL.

Zadatak 52: Rezolucija uz strategiju prvenstva jedinice

Date su pretpostavke

1. A(C1)
2. $\forall x \; \forall y \; \{[\, A(x) \vee B(y) \,] \Rightarrow C(x,y) \}$
3. $\forall x \; \{[\, A(x) \wedge B(x) \,] \Rightarrow C(x,x) \}$

a) Prevesti pretpostavke u konjuktivnu normalnu formu.

b) Rezolucijom dokazati $\exists x \; [C(x,x)]$. Za izbor stavova koristiti strategiju *prvenstva jedinice* (engl. *unit preference*).

Rešenje

a) Pretpostavke u konjuktivnoj normalnoj formi su:

1. A(C1)

2'. $\neg A(x_1) \vee C(x_1, y_1)$

2''. $\neg B(y_2) \vee C(x_2, y_2)$

3. $\neg A(x_3) \vee \neg B(x_3) \vee C(x_3, x_3)$

b) Dodajemo negaciju teoreme u klauzalnoj formi:

4. $\neg C(x_4, x_4)$

Primenom rezolucije, uz *unit-preference* strategiju:

1., 2'. $\xrightarrow{x_1=C_1}$ 5. $C(C_1, y_1)$

4., 5. $\xrightarrow{x_4=y_1=C_1}$ NIL

Zadatak 53: Svet blokova (funkcija Puton)

Date su sledeće aksiome:

1. $\forall x\ \forall y\ \forall s\ \{ Clear(x,s) \wedge Clear(y,s) \wedge \neg Equal(x,y) \Rightarrow On[x,y,Puton(x,y,s)] \}$
2. $\forall x\ \forall y\ \forall s\ \{ \neg On(x,y,s) \Rightarrow Clear(y,s) \}$
3. $\forall x\ \forall y\ \forall s\ \{ On(x,y,s) \wedge Clear(x,s) \Rightarrow On[x,Table,Puton(x,Table,s)] \}$
4. $\forall x\ \forall y\ \forall s\ \{ On(x,y,s) \wedge Clear(x,s) \Rightarrow Clear[y,Puton(x,Table,s)] \}$
5. $\forall x\ \forall y\ \forall z\ \forall s\ \{ On(x,y,s) \wedge \neg Equal(z,x)] \Rightarrow On[x,y,Puton(z,Table,s)]\}$
6. On(A,B,S)
7. On(B,C,S)
8. Clear(A,S)
9. ¬Equal(A,B)
10. ¬Equal(A,C)
11. ¬Equal(B,C)

a) Prevesti aksiome u konjuktivnu normalnu formu.

b) Dokazati rezolucijom: $\exists s$ [Clear(C,s)]

Rešenje

a) Pretpostavke u KNF glase:

1. $\neg Clear(x_1,s_1)\ \ \neg Clear(y_1,s_1) \vee Equal(x_1,y_1) \vee On[x_1,y_1,Puton(x_1,y_1,s_1)]$

2. $On(x_2,y_2,s_2) \vee Clear(y_2,s_2)$

3. $\neg On(x_3,y_3,s_3) \vee \neg Clear(x_3,s_3) \vee On[x_3,Table,Puton(x_3,Table,s_3)]$

4. $\neg On(x_4,y_4,s_4) \vee \neg Clear(x_4,s_4) \vee Clear[y_4,Puton(x_4,Table,s_4)]$

5. $\neg On(x_5,y_5, s_5) \vee Equal(z, x_5) \vee On[x_5, y_5, Puton(z, Table, s_5)]$

6. On(A,B,S)

7. On(B,C,S)

8. Clear(A,S)

9. ¬Equal(A,B)

10. ¬Equal(A,C)

11. ¬Equal(B,C)

b) Pretpostavkama dodajemo negaciju tvrđenja

$$\neg\exists s\ [\ Clear(C,s)\ ]$$

u konjuktivnoj normalnoj formi:

12. $\neg Clear(C,s_6)$

Sada stavove kombinujemo primenom rezolucije

4., 8. $\xrightarrow{x_4=A,\ s_4=S}$ 13. $\neg On(A,y_4,S) \vee Clear[y_4,Puton(A,Table,S)]$

6., 13. $\xrightarrow{y_4=B}$ 14. Clear[B,Puton(A,Table,S)]

4. , 14. $\xrightarrow{x_4=B,\ s_4=Puton(A,Table,S)}$ 15. $\neg On[B,y_4,Puton(A,Table,S)] \vee$

$\vee Clear[y_4,Puton(B,Table,Puton(A,Table,S))]$

12., 15. $\xrightarrow{y_4=C,\ s_6=Puton[B,Table,Puton(A,Table,S)]}$ 16. $\neg On[B,C,Puton(A,Table,S)]$

5., 16. $\xrightarrow{x_5=B,\ y_5=C,\ z=A,\ s_5=S}$ 17. $\neg On(B, C, S) \vee Equal(A, B)$

9., 17. $\longrightarrow$ 18. $\neg On(B, C, S)$

7., 18. $\longrightarrow$ NIL

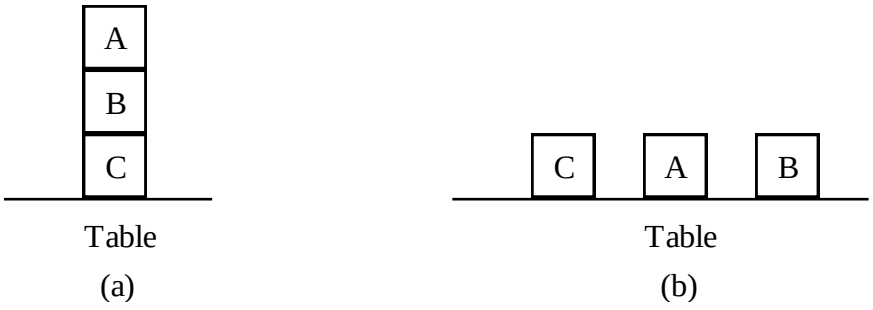
Zadati sistem aksioma moguće je interpretirati na sledeći način:

- A, B i C predstavljaju blokove na stolu Table,
- predikati On i Clear opisuju međusobni položaj blokova u određenom stanju sistema blokova: On(x, y, s) je ispunjeno ako je u stanju s blok x na bloku (ili stolu) y. Clear(x, s) je ispunjeno ako se u stanju s na bloku x ne nalazi nijedan drugi blok. Predikat Equal(x,y) opisuje identičnost blokova x i y.
- S predstavlja početno stanje sistema prikazano na slici 100*a*,
- funkcija Puton(x, y, s) daje kao rezultat novo stanje u koje sistem dolazi iz stanja s stavljanjem bloka x na blok (ili sto) y. Iskazi 1., 3., 4. i 5. definišu preduslove za primenu ove funkcije, to jest, operatora promene stanja sistema.

Prema ovoj interpretaciji, tražilo se da se nađe pokaže da se sistem može iz početnog stanja dovesti u stanje u kome se na bloku C ne nalazi nijedan drugi blok. U postupku rešavanja dokazano je da takvo stanje postoji (slika 100*b*). Traženo stanje opisano je izrazom sa kojim je unificirana promenljiva s_6 prilikom spajanja stavova 12 i 15:

$$s_6 = Puton[B, Table, Puton(A, Table,S)]$$

Interpretacija ovoga izraza je da iz početnog stanja treba prvo staviti blok A na sto, pa zatim i blok B na sto.



Slika 100

Zadatak 54: Kontradikcija u pretpostavkama

Na osnovu sledećih pretpostavki

1. $\forall x\ [\neg \text{Jednako}(x,x+1)]$
2. Jednako(2,3)

a) Rezolucijom izvesti zaključak: Sve jabuke su kruške.

b) Zašto je u tački a) bilo moguće izvesti takav zaključak?

Rešenje

a) Potrebno je zaključak predstaviti u formi predikatske formule, na primer:

$$\forall x\ [\ \text{Jabuka}(x) \Rightarrow \text{Kruška}(x)\ ]$$

Prevedimo pretpostavke i negaciju zaključka u konjuktivnu normalnu formu:

1. $\neg$Jednako(x,x+1)
2. Jednako(2,3)
3'. Jabuka(C)
3". $\neg$Kruška(C)

Primenom rezolucije utvrđujemo da u skupu stavova postoji kontradikcija

1., 2. $\xrightarrow{x=2}$ NIL

čime je 'dokazano' da polazno tvrđenje važi. Primetimo negacija tvrđenja uopšte nije korišćena u izvođenju zaključka.

b) Iz kontradiktornih pretpostavki moguće je izvući bilo kakav zaključak.

Zadatak 55: Problem unifikacije stavova

Objasniti zašto se sledeći skupovi literala ne mogu unificirati:

a) {P(A), P(B)}

b) {P(f(A),x), P(x,A)}

c) { P(f(x,x), A), P(f(y,f(y,A)), A) }

Rešenje

Dokazivanje rezolucijom zahteva da se u klauzulama koje se razmatraju nađu dva člana (literala) koja se poklapaju u svemu osim što je jedan od njih negiran. Unifikacija je proces nalaženja smena koje dovode do poklapanja literala. Unifikacija podleže pravilima predstavljenim algoritmom 14 u dodatku 1.

a) Predstavimo zadate literale listama:

P, A

P, B

Vidimo da se prvi elementi (imena predikata) poklapaju u obe liste, dok se drugi elementi ne poklapaju. Pošto se radi o konstantama, nije ispunjen uslov iz tačke 3.1. procedure unifikacije pa se zadati literali ne mogu unificirati.

b) U ovom slučaju imamo liste

P, f(A), x

P, x, A

Prvi elementi se poklapaju. Za druge elemente lista, prema tački 3.2. procedure, uvodimo smenu x = f(A). Posle zamene svih pojava promenljive x, u obe liste, izrazom f(A) liste imaju sledeći izgled:

P, f(A), f(A)

P, f(A), A

Poslednji element prve liste je konstantna vrednost koju funkcija f vraća za argument A, dok je u drugoj listi poslednji element konstanta A. U opštem slučaju, ne radi se o istim vrednostima pa nije zadovoljena tačka 3.1. procedure unifikacije.

c) Predstavimo zadate literale listama:

P, f(x,x), A

P, f(y,f(y,A)), A

Prvi i treći elementi su identični u obe liste. Drugi element u obe liste je funkcija f sa dva argumenta. Potrebno je upariti svaki od argumenata putem zamene promenljivih. Prvi argumenti uparuju se smenom x = y. Posle zamene oba dešavanja promenljive x u prvoj listi promenljivom y, liste imaju sledeći izgled:

P, f(y,y), A

P, f(y,f(y,A)), A

Sada se može zaključiti da bi za uparivanje drugih argumenata funkcije f bilo potrebno izvršiti zamenu y = f(y,A). Ova zamena krši ograničenje da se promenljiva ne sme zameniti izrazom koji sadrži tu promenljivu. Prema tome, unifikacija ni u ovom slučaju nije moguća.

Zadatak 56: Poslednji element liste

Izraz cons(*x*,*y*) označava listu dobijenu umetanjem *x* na čelo liste *y*. Praznu listu označavamo sa EMPTY; lista 2 data je sa cons(2,EMPTY); lista (1,2) sa cons (1,cons(2,EMPTY)); itd. Izraz LAST(*x*,*y*) znači da je *y* poslednji element liste *x*. Imamo sledeće aksiome:

1. $(\forall u)$ LAST(cons(u,EMPTY),u)

2. $(\forall x)(\forall y)(\forall z)\{LAST(y,z) \Rightarrow LAST[cons(x,y),z]\}$

Primenom rezolucije dokazati sledeću teoremu :

$(\exists v)LAST\{cons[2,cons(1,EMPTY)],v\}$

Rešenje

Aksiome prevodimo u konjuktivnu normalnu formu:

1. LAST[cons(u,EMPTY),u]

2. $\neg$LAST(y,z) $\vee$ LAST[cons(x,y),z]

Aksiomama dodajemo negaciju teoreme

$\neg(\exists v)LAST\{cons[2,cons(1,EMPTY)],v\}$

prevedenu u konjuktivnu normalnu formu

3. $\neg$LAST{cons[2,cons(1,EMPTY)],v}

Primenimo rezoluciju sa strategijom skupa podrške: Stavovi 1 i 3 ne mogu se unificirati jer se drugi argumenti funkcije cons razlikuju (u stavu 1 to je EMPTY, u stavu 3 radi se o cons(1,EMPTY)). Zato se uzimaju stavovi 2 i 3 za primenu rezolucije:

2 i 3. $\xrightarrow{x = 2,\ z = v,\ y = cons(1,EMPTY)}$ 4. $\neg$LAST[cons(1,EMPTY),v]

Pošto nema više stavova za spajanje sa stavom 3 (negacijom teoreme), prelazimo na spajanje stava 4 sa ostalim stavovima. Rezolucijom iz stavova 1 i 4 se dobija protivrečnost, iz čega sledi tačnost polazne teoreme:

1. i 4. $\xrightarrow{u = v = 1}$ NIL

Zadatak 57: Dokazivanje tautologija

Primenom rezolucije dokazati da je svaka od sledećih formula tautologija:

a) $(P \Rightarrow Q) \Rightarrow [(R \vee P) \Rightarrow (R \vee Q)]$

b) $[(P \Rightarrow Q) \Rightarrow P] \Rightarrow P$

c) $(\neg P \Rightarrow P) \Rightarrow P$

d) $(P \Rightarrow Q) \Rightarrow (\neg Q \Rightarrow \neg P)$

Rešenje

U opštem slučaju dokazivanja neke predikatske formule rezolucijom, posedujemo skup aksioma P i tvrđenje T. Rezoluciju primenjujemo na skup P $\cup \neg$T to jest na skup koji se sastoji od polaznih aksioma i negacije tvrđenja. Ukoliko primenom rezolucije dobijemo prazan stav, to jest protivrečnost u skupu P $\cup \neg$T to znači da skup P $\cup \neg$T ne može biti zadovoljen ni za jednu interpretaciju (semantiku koju dodeljujemo pojedinim konstantama, promenljivama, funkcijama i predikatima) aksioma i tvrđenja. Iz ovoga sledi da je skup P $\cup$ T zadovoljen u svakoj interpretaciji, sledstveno tome T je teorema koja se dobija iz polaznih aksioma P nezavisno od interpretacije.

Zamislimo sada da imamo samo tvrđenje T bez bilo kakvih aksioma. Ponovimo razmatranje iz prethodnog pasusa stavljajući da je P prazan skup: Ako rezolucijom pokažemo da $\neg$T nije zadovoljeno, dokazali smo da je T zadovoljeno nezavisno od interpretacije i nezavisno od ikakvih polaznih pretpostavki. Formule koje su zadovoljene nezavisno od interpretacije, dakle važe u svakoj interpretaciji nazivaju se *valjanim* formulama u predikatskoj logici.

Propoziciona (iskazna) logika je podskup predikatske logike bez promenljivih. Formule iskazne logike koje su tačne za svaku interpretaciju (to jest, bez obzira koju istinitosnu vrednost imaju predikati koje se pojavljuju u formuli) nazivaju se *tautologije*. Tautologije u iskaznoj logici predstavljaju isto ono što valjane formule predstavljaju u predikatskoj logici.

Prema tome, da bismo dokazali da je neka formula tautologija, potrebno je primenom rezolucije naći protivrečnost u skupu stavova koji predstavljaju negaciju polazne formule.

a) Negiramo datu formulu:

$$\neg\{(P \Rightarrow Q) \Rightarrow [(R \vee P) \Rightarrow (R \vee Q)]\}$$

Prevodimo dobijenu formulu u konjuktivnu normalnu formu (KNF).

- Eliminacijom implikacije dobija se:

  $\neg\{\neg(\neg P \vee Q) \vee [\neg(R \vee P) \vee (R \vee Q)]\}$

- Negaciju spuštamo na atomske formule. Prebacimo najpre krajnje levu negaciju 'pod zagradu':

  $(\neg P \vee Q) \wedge (R \vee P) \wedge \neg(R \vee Q)$

- Transformisanjem poslednjeg člana formule dobijamo:

  $(\neg P \vee Q) \wedge (R \vee P) \wedge \neg R \wedge \neg Q$

- Podelom na klauzule dobijamo traženu formu:

  1. $\neg P \vee Q$

  2. $R \vee P$

  3. $\neg R$

  4. $\neg Q$

Primenom rezolucije tražimo protivrečnost u navedenim stavovima (stavovi za spajanje će biti birani tako da se cilj postigne u što manje koraka):

1., 2. $\rightarrow$ 5. $Q \vee R$

3., 5. $\rightarrow$ 6. Q

4., 6. $\rightarrow$ NIL

Dobijanjem protivrečnosti dokazano je da je polazna formula tautologija.

b) Prevodimo negaciju formule u KNF:

$\neg\{[(P \Rightarrow Q) \Rightarrow P] \Rightarrow P\}$

- Eliminacija implikacija

$\neg\{\neg[\neg(\neg P \vee Q) \vee P] \vee P\}$

- Sledeća dva koraka odnose se na spuštanje negacija do predikata

$\neg\{\neg[(P \wedge \neg Q) \vee P] \vee P\}$

$[(P \wedge \neg Q) \vee P] \wedge \neg P$

- Primena zakona distribucije na stavove unutar srednje zagrade

$[(P \vee P) \wedge (\neg Q \vee P)] \wedge \neg P$

- Podela na klauzule (primetiti da se prva klauzula uprošćava koristeći činjenicu da je $P \vee P$ isto što i samo jedno P)

1. P
2. $\neg Q \vee P$
3. $\neg P$

Kombinovanjem 1. i 3. stava odmah se dobija prazan stav čime je dokaz gotov.

c) Prevodimo negaciju formule u KNF:

$\neg[(\neg P \Rightarrow P) \Rightarrow P]$

- Eliminacija implikacija:

$\neg[\neg(P \vee P) \vee P]$

- Eliminacija spoljne negacije

$(P \vee P) \wedge \neg P$

Podelom se dobijaju stavovi P i $\neg P$. Primenom rezolucije na ova dva stava odmah se dobija prazan stav.

d) Prevodimo negaciju formule u KNF:

$\neg[(P \Rightarrow Q) \Rightarrow (\neg Q \Rightarrow \neg P)]$

- Eliminacija implikacija:

$\neg[\neg(\neg P \vee Q) \vee (Q \vee \neg P)]$

- Eliminacija spoljne negacije

$(\neg P \vee Q) \wedge \neg(Q \vee \neg P)$

- Primena De Morganovog zakona na drugu disjunkciju

$(\neg P \vee Q) \wedge \neg Q \wedge P$

Podelom se dobijaju se stavovi:

1. $\neg P \vee Q$
2. $\neg Q$
3. P

Dvostrukom primenom rezolucije

1., 2. $\rightarrow$ 4. $\neg P$

3., 4. $\rightarrow$ NIL

dobija se prazan stav što kompletira dokaz.

Zadatak 58: Dokazivanje valjanosti formule

Pokazati da je sledeća dobro formirana formula valjana primenom rezolucije:

$$(\forall x)\{P(x) \wedge [Q(A) \vee Q(B)]\} \Rightarrow (\exists x)\,[P(x) \wedge Q(x)\,]$$

Rešenje

Dobro formirana formula predikatske logike naziva se *valjanom* ako je tačna za svaku moguću interpretaciju predikata koji se pojavljuju u toj formuli. U prethodnom zadatku imali smo specijalan slučaj valjanih formula koje važe u propozicionoj logici i nazivaju se tautologije. Dokaz da je neka formula valjana može se sprovesti na isti način kao i u slučaju tautologija (videti analizu prethodnog zadatka): na stavove dobijene negiranjem polazne formule primenjuje se rezolucija dok se ne dobije prazan stav. Dobijanje praznog stava je dokaz da postoji protivrečnost u negaciji polazne formule što dokazuje da je polazna formula valjana.

Negiramo datu formulu:

$$\neg\{(\forall x)[P(x) \wedge (Q(A) \vee Q(B))] \Rightarrow (\exists x)\,(P(x) \wedge Q(x)\,)\,\}$$

Prevodimo formulu u konjuktivnu normalnu formu kao pripremu za sprovođenje rezolucije.

- Eliminišemo implikaciju:

$$\neg\{\neg(\forall x)[P(x) \wedge (Q(A) \vee Q(B))] \vee (\exists x)\,(P(x) \wedge Q(x)\,)\,\}$$

- Spoljnu negaciju 'uvlačimo pod zagradu':

$$(\forall x)[P(x) \wedge (Q(A) \vee Q(B))] \wedge \neg(\exists x)\,(P(x) \wedge Q(x)\,)$$

- Preostalu negaciju spuštamo do atomskih formula:

$$(\forall x)[P(x) \wedge (Q(A) \vee Q(B))] \wedge (\forall x)\,(\neg P(x) \vee \neg Q(x)\,)$$

- Kvantifikatore izvlačimo na levu stranu uz preimenovanje desnog pojavljivanja promenljive x:

$$(\forall x)(\forall y)\{[P(x) \wedge (Q(A) \vee Q(B))] \wedge (\neg P(y) \vee \neg Q(y)\,)\,\}$$

- Razlažemo formulu na pojedine klauzule

1. $P(x)$
2. $Q(A) \vee Q(B)$

3. $\neg P(y) \vee \neg Q(y)$

Na ovako dobijene stavove primenjujemo rezoluciju:

1., 3. $\xrightarrow{x=y}$ 4. $\neg Q(y)$

2., 4. $\xrightarrow{y=A}$ 5. $Q(B)$

4., 5. $\xrightarrow{y=B}$ NIL

Dobijen je prazan stav NIL čime je dokaz završen.

Zadatak 59: Problem skolemizacije

Pokazati korišćenjem rezolucije da dobro formirana formula $(\exists x)P(x)$ logički sledi iz dobro formirane formule $[P(A1) \vee P(A2)]$. Međutim, Skolemova forma za $(\exists x)P(x)$, to jest, $P(A)$, ne sledi logički iz $[P(A1) \vee P(A2)]$. Objasniti.

Rešenje

Pokažimo najpre, uz pomoć rezolucije, da tačnost $(\exists x)P(x)$ sledi iz pretpostavke $[P(A1) \vee P(A2)]$. Pretpostavku i negaciju tvrđenja prevodimo u konjuktivnu normalnu formu:

1. $P(A1) \vee P(A2)$

2. $\neg P(x)$

Dvostrukom primenom rezolucije dobijamo prazan stav čime je polazno tvrđenje dokazano:

1., 2. $\xrightarrow{x = A1}$ 3. $P(A2)$

2., 4. $\xrightarrow{x = A2}$ NIL

Razmotrimo sada drugo tvrđenje. Pretpostavka i negacija tvrđenja imaju sledeće konjuktivne normalne forme:

1. $P(A1) \vee P(A2)$

2. $\neg P(A)$

Rezolucija se ne može primeniti na stavove 1. i 2. jer se ne poklapaju konstante u predikatima $P(A1)$ i $\neg P(A)$. Prema tome, rezolucijom se u ovom slučaju ne može dokazati ispravnost tvrđenja.

Da bi se objasnio ovakav rezultat treba se podsetiti strategije dokazivanja stavova rezolucijom: pretpostavi se tačnost negacije tvrđenja, negacija se dodaje polaznim pretpostavkama, pa se rezolucijom se pokušava ustanoviti nekonzistentnost ovako dobijenog skupa stavova. U slučaju pojave nekonzistentnosti dokazano je da negacija tvrđenja ne može biti tačna pri datim pretpostavkama (koje su po definiciji tačne), pa se iz toga zaključuje tačnost polaznog tvrđenja pri datim pretpostavkama.

U našem slučaju pretpostavka tvrdi da je predikat $P(x)$ tačan za neku vrednost promenljive x. Ta vrednost je ili A1 ili A2 (ili obe ove vrednosti).

Negacija tvrđenja P(x) je znatno 'jače' tvrđenje od negacije tvrđenja P(A): ¬P(x) znači da predikat P(x) nije ispunjen ni za jednu moguću vrednost x. Ovaj stav se može osporiti (što je primenom rezolucije i pokazano) na osnovu pretpostavke, čak i ako se iz pretpostavke ne zna tačna vrednost promenljive x za koju važi P(x).

Skolemizacijom formule (∃x)P(x) dobijena je formula P(A). Drugim rečima, pošto postoji neka vrednost za x tako da predikat P(x) važi, uveli smo neku konstantu A i ustvrdili da P(A) ima tačnu istinitosnu vrednost. Pri tome ne definišemo način na koji se A može odrediti. Negacija ovog tvrđenja, ¬P(A) znači da postoji neka konkretna vrednost promenljive x za koju ne važi P(x). Ovo je znatno 'slabije' tvrđenje nego negacija originalnog (neskolemizovanog) tvrđenja i ne može se osporiti pretpostavkom, s obzirom da pretpostavka ne tvrdi da je P(x) tačno za svako x. Prema tome, skolemizacija može dovesti do 'gubitka opštosti' tvrđenja u toj meri da ono ne može biti dokazano.

Zadatak 60: Raspored Cigala

Dati su sledeći iskazi:

- Cigla A nalazi se na stolu.
- Cigla B nalazi se na cigli A.
- Cigla C nalazi se na cigli A.
- Cigla D nalazi se na cigli B.
- Cigla A je plave boje.
- Cigla B je plave boje.
- Cigla C je crvene boje.
- Cigla D je zelene boje.
- Cigla je iznad neke druge cigle X ako se nalazi na njoj, ili se nalazi na cigli koja je iznad cigle X.

a) Predstaviti ove iskaze u obliku stavova predikatske logike.

b) Primenom rezolucije dokazati da se cigla D nalazi iznad plave cigle.

Rešenje

a) Upotrebljeni predikati su Na, Boja i Iznad, kao i konstante za cigle, boje i sto:

1. Na(A, Sto)
2. Na (B, A)
3. NA (C, A)
4. Na (D, B)
5. Boja (A, Plava)
6. Boja (B, Plava)
7. Boja (C, Crvena)

8. Boja (D, Zelena)

9. $(\forall x)(\forall y)\ (Na(x, y) \vee (\exists z)(Na(x,z) \wedge Iznad(z,y)) \Rightarrow Iznad(x,y))$

b) Da bismo primenili rezoluciju, moramo da prevedemo bazu znanja u KNF. Za formulu 9. imamo:

$$(\forall x)(\forall y)\ (\neg(Na(x, y) \vee (\exists z)(Na(x,z) \wedge Iznad(z,y))) \vee Iznad(x,y))$$

$$(\forall x)(\forall y)\ ((\neg Na(x, y) \wedge (\forall z)(\neg Na(x,z) \vee \neg Iznad(z,y))) \vee Iznad(x,y))$$

$$(\neg Na(x, y) \wedge (\neg Na(x,z) \vee \neg Iznad(z,y))) \vee Iznad(x,y)$$

Dobijamo dve klauzule, uz preimenovanje promenljivih:

9a. $\neg Na(x, y) \vee Iznad(x,y)$

9b. $\neg Na(x1, z) \vee \neg Iznad(z,y1) \vee Iznad(x1,y1)$

Upit glasi:

$$(\exists x)\ (Iznad(D,x) \wedge Boja(x,Plava))$$

Za primenu rezolucije dodajemo negaciju upita bazi znanja:

10. $\neg Iznad(D, x2) \vee \neg Boja(x2,Plava)$

Primenićemo strategiju skupa podrške za izbor klauzula na koje primenjujemo pravilo:

10, 9a $\xrightarrow{x/D,\quad y/x2}$ 11. $\neg Na(D,x3) \vee \neg Boja(x3,Plava)$

11, 4 $\xrightarrow{x3/B}$ 12. $\neg Boja(B,Plava)$

12, 6 $\longrightarrow$ NIL

Primetimo da posle primene rezolucije, u dobijenoj klauzuli takođe preimenujemo promenljive. Na taj način u bazi znanja nikad nemamo istu promenljivu u dve različite klauzule. Druga mogućnost je da preimenovanje promenljivih obavljamo neposredno pre primene rezolucije.

Pošto smo dobili praznu klauzulu, dokazali smo tvrđenje. Dodatno, pošto smo imali substituciju $x3/B$, dokazali smo da cigla B zadovoljava upit.

Zadatak 61: Premisa-zaključak

Primenom rezolucije dokazati da iz premise

$$(\forall x)\ ((\exists y)(S(x,y) \wedge M(y)) \Rightarrow (\exists y)\ (I(y) \wedge E(x,y)))$$

sledi zaključak

$$\neg\ (\exists x)I(x) \Rightarrow (\forall x)(\forall y)(S(x,y) \Rightarrow \neg M(y))$$

Rešenje

Pretvorimo premisu u KNF:

$$(\forall x)((\exists y)(S(x,y) \wedge M(y)) \Rightarrow (\exists y)(I(y) \wedge E(x,y)))$$

$(\forall x)(\neg (\exists y)(S(x,y) \wedge M(y)) \vee (\exists y)(I(y) \wedge E(x,y)))$

$(\forall x)((\forall y)(\neg S(x,y) \vee \neg M(y)) \vee (\exists y)(I(y) \wedge E(x,y)))$

Sada za promenljivu *y* za koju je vezan egzistencijalni kvantifikator primenjujemo pravilo skolenizacije. Ona je pod uticajem samo promenljive *x*, pa ćemo je zameniti funkcijom sa jednim argumentom, neka to bude $\Gamma(x)$ (argument ove funkcije će biti promenljiva *x*):

$$(\forall x)((\forall y)(\neg S(x,y) \vee \neg M(y)) \vee (I(\Gamma(x)) \wedge E(x,\Gamma(x))))$$

Sada možemo da uklonimo kvantifikatore:

$$(\neg S(x,y) \vee \neg M(y)) \vee (I(\Gamma(x)) \wedge E(x,\Gamma(x)))$$

$$(\neg S(x,y) \vee \neg M(y) \vee I(\Gamma(x))) \wedge (\neg S(x,y) \vee \neg M(y) \vee E(x,\Gamma(x)))$$

Dobijamo dve klauzule:

1. $\neg S(x,y) \vee \neg M(y) \vee I(\Gamma(x))$
2. $\neg S(x1,y1) \vee \neg M(y1) \vee E(x1,\Gamma(x1))$

Bazi znanja dodajemo negaciju zaključka prevedenog u KNF:

$$\neg(\neg(\exists x)I(x) \Rightarrow (\forall x)(\forall y)(S(x,y) \Rightarrow \neg M(y)))$$

$$\neg((\exists x)I(x) \vee (\forall x)(\forall y)(\neg S(x,y) \vee \neg M(y)))$$

$$\neg(\exists x)I(x) \wedge \neg(\forall x)(\forall y)(\neg S(x,y) \vee \neg M(y))$$

$$(\forall x)\neg I(x) \wedge (\exists x)(\exists y)(S(x,y) \wedge M(y))$$

Sada primenjujemo pravilo skolenizacije. Pošto promenljive vezane za egzistencijalni kvantifikator nisu pod uticajem univerzalnih kvantifikatora, zamenjujemo ih konstantama:

$$(\forall x)\neg I(x) \wedge (S(X,Y) \wedge M(Y))$$

Dobili smo tri klauzule:

3. $\neg I(x2)$
4. $S(X,Y)$
5. $M(Y)$

Dalje primenjujemo rezoluciju:

$1, 4 \xrightarrow{x/X,\quad y/Y} 6. \neg M(Y) \vee I(\Gamma(X))$

$5, 6 \longrightarrow 7. I(\Gamma(X))$

$3, 7 \xrightarrow{x2/\Gamma(X)} \text{NIL}$

Pošto smo dobili praznu klauzulu, tvrđenje je dokazano.

2.2. Produkcioni sistemi

Zadatak 62: Zaključivanje direktnim ulančavanjem

Posmatrajmo sledeću bazu znanja koja se sastoji od pravila (produkcija) i činjenica.

if b(x) then a(x)

if c(x) and d(x) then b(x)

if e and f(x) then d(x)

if g(x) then c(x)

g(2)

f(5)

g(5)

e

a) Koje sve nove činjenice i po kom redosledu proizilaze primenom direktnog ulančavanja sa fokusiranjem pažnje (engl. *focus of attention*) na nove činjenice?

b) Prikazati proces zaključivanja o istinitosti cilja a(x) povratnim ulančavanjem.

c) Koje sve nove činjenice ili pravila i po kom redosledu proizilaze iz ove baze znanja ako se primenjuje ciklično hibridno ulančavanje (engl. *rule cycle hybrid*)?

Rešenje

Produkcioni sistemi su forma pogodna za izražavanje takozvanog proceduralnog znanja. Produkcioni sistem je skup pravila oblika logičke implikacije:

$$preduslov \Rightarrow zaključak$$

U *analitičkim* (deduktivnim) produkcionim sistemima uobičajeno je da je *preduslov* logički izraz a *zaključak* predikat ili implikacija više predikata i rad sistema se svodi na utvrđivanje istinitosne vrednosti određenih predikata.

Sintetički produkcioni sistemi pored skupa pravila poseduju i model sveta koji se ažurira primenom pravila. U takvim sistemima *preduslov* je logički izraz, a *zaključak* sadrži niz akcija koji menjaju trenutno stanje u modelu sveta.

Produkcioni sistem u ovom zadatku spada u analitičke produkcione sisteme. Predikatski stavovi koji se pojavljuju u pravilima dele se na:

- pretpostavke, koje se pojavljuju isključivo u preduslovima pravila,
- ciljeve, koji se pojavljuju isključivo u zaključcima i
- međupredikate (engl. *intermediate predicates*), koji se mogu pojaviti i u preduslovima jednih i u zaključcima drugih pravila.

Zaključivanje pomoću produkcionog sistema predstavlja proces utvrđivanja istinitosne vrednosti zaključaka na osnovu zadatih istinitosnih vrednosti pretpostavki (I za neke od međupredikata moguće je zadati istinitosnu vrednost u kom slučaju oni igraju ulogu pretpostavki. Međupredikati mogu igrati i ulogu ciljeva ukoliko nam je od važnosti njihova istinitosna vrednost na kraju procesa zaključivanja). U procesu zaključivanja može se koristiti jedna od sledećih osnovnih strategija:

- *Direktno ulančavanje*. Kod ove strategije na osnovu pretpostavki, koristeći pravila u smeru od preduslova ka zaključcima, određuju se svi mogući zaključci.
- *Povratno ulančavanje*. Kod ove strategije polazi se od datog cilja i koristeći produkcije u smeru od zaključaka ka preduslovima utvrđuje se da li su zadovoljeni svi preduslovi neophodni za dati cilj.
- *Hibridno ulančavanje*. Strategije direktnog i povratnog ulančavanja mogu se iskombinovati na više načina sa ciljem da se iskoriste prednosti svake od njih.

U rešenju zadatka detaljno je prikazan proces zaključivanja primenom direktnog ulančavanja. Nove činjenice u bazu znanja dodavane su ispred postojećih činjenica (što ima uticaj na tok zaključivanja). Ovo je takozvano zaključivanje sa fokusiranjem pažnje na nove činjenice.

U zadatom produkcionom sistemu stavovi e, f(x) i g(x) predstavljaju pretpostavke, stavovi b(x), c(x) i d(x) su međupredikati a stav a(x) je ciljni stav. Polazne činjenice g(2), f(5), g(5) i e nam definišu istinitost stavova - pretpostavki za određene konkretne vrednosti promenljivih.

Zadati produkcioni sistem sadrži predikate koji imaju argumente u kojima se pojavljuju promenljive. Promenljive unutar pravila posmatraju se kao univerzalno kvantifikovane. U procesu zaključivanja promenljive iz pravila se vezuju, odnosno, dobijaju konkretne vrednosti tako su zaključci predstavljeni stavovima koji sadrže konstante. U procesu povratnog ulančavanja (videti deo pod b) mogu se u okviru zaključaka pojaviti promenljive koje se tada posmatraju kao egzistencijalno kvantifikovane, to jest, cilj je ispunjen ako se nađe bar jedna konkretna vrednost promenljive za koju je dati predikat ispunjen.

Primenimo algoritam zaključivanja direktnim ulančavanjem naveden u dodatku 1 (algoritam 12) na zadati produkcioni sistem. Početna situacija opisana je na sledeći način:

Lista neupotrebljenih činjenica L : g(2), f(5), g(5), e
Produkcioni sistem:
1. if b(x) then a(x)
2. if c(x) and d(x) then b(x)
3. if e and f(x) then d(x)
4. if g(x) then c(x)
Dobijene činjenice: -

Prvo se sledi činjenica g(2). Jedino preduslov pravila 4 uparuje ovu činjenicu, pri čemu promenljiva x dobija vrednost 2. Kada se predikat g(x) ukloni iz pravila 4 na osnovu koraka 2.1.1., dobija se novo 'pravilo' c(2) (podsetimo se da je promenljiva x vezana u procesu uparivanja preduslova pravila 4. sa činjenicom g(2)). Prema koraku 2.1.2 dobijena je nova činjenica koja se stavlja na čelo liste L. Jedino zaključak pravila 4 uparuje činjenicu c(2), ali se ovo pravilo ne uklanja iz baze jer je zaključak opštiji s obzirom da sadrži promenljivu.

Koraci 2.1.3. i 2.1.4. preskaču se u ovom slučaju. Situacija pre nove iteracije u algoritmu je sledeća:

Lista neupotrebljenih činjenica L : c(2), f(5), g(5), e
Produkcioni sistem:
1. if b(x) then a(x)
2. if c(x) and d(x) then b(x)
3. if e and f(x) then d(x)
4. if g(x) then c(x)
Dobijene činjenice: c(2)

U novoj iteraciji algoritma zaključivanja sledi se činjenica c(2). Jedino pravilo 2. u preduslovu poseduje predikat c(x). Prema 2.1.1. od pravila 2. kreira se novo pravilo

if d(2) then b(2)

koje se prema 2.1.3 stavlja ispred pravila 2. Pravilo 2 se ne eliminiše jer su i preduslov i zaključak generalniji od preduslova i zaključka novog pravila. Posle ove iteracije imamo sledeću situaciju:

Lista neupotrebljenih činjenica L : f(5), g(5), e
Produkcioni sistem:
1. if b(x) then a(x)
1a. if d(2) then b(2)
2. if c(x) and d(x) then b(x)
3. if e and f(x) then d(x)
4. if g(x) then c(x)
Dobijene činjenice: c(2)

U narednoj iteraciji sledi se činjenica f(5) i bira pravilo 3 kao jedino primenljivo. U bazu znanja dodaje se novo pravilo 2a. tako da je trenutna situacija sledeća:

Lista neupotrebljenih činjenica L : g(5), e
Produkcioni sistem:
1. if b(x) then a(x)
1a. if d(2) then b(2)
2. if c(x) and d(x) then b(x)
2a. if e then d(5)
3. if e and f(x) then d(x)
4. if g(x) then c(x)
Dobijene činjenice: c(2)

Naredna činjenica koja se sledi je g(5). Na osnovu pravila 4 koje je jedino primenljivo u ovoj situaciji dobija se nova činjenica c(5) i dodaje na čelo liste L, koja ima izgled:

Lista neupotrebljenih činjenica L : c(5), e

dok baza znanja ostaje neizmenjena.

Sada se sledi činjenica c(5). Ovoj činjenici odgovara pravilo 2 od koga nastaje novo pravilo 1b pa je situacija sledeća:

Lista neupotrebljenih činjenica L : e
Produkcioni sistem:
1. if b(x) then a(x)
1a. if d(2) then b(2)
1b. if d(5) then b(5)
2. if c(x) and d(x) then b(x)
2a. if e then d(5)
3. if e and f(x) then d(x)
4. if g(x) then c(x)
Dobijene činjenice: c(2), c(5)

Činjenici e odgovaraju pravila 2a. i 3. Na osnovu pravila 2a. dobija se nova činjenica d(5) koja se stavlja na čelo liste L, a pravilo 2a. se uklanja iz baze pošto zaključak potpuno odgovara novodobijenoj činjenici. Na osnovu pravila 3 dobija se novo pravilo 2b. a pravilo 3 se uklanja kao redundantno, pa imamo sledeću situaciju:

Lista neupotrebljenih činjenica L : d(5)
Produkcioni sistem:
1. if b(x) then a(x)
1a. if d(2) then b(2)
1b. if d(5) then b(5)
2. if c(x) and d(x) then b(x)
2b. if f(x) then d(x)
4. if g(x) then c(x)
Dobijene činjenice: c(2), c(5), d(5)

Činjenicu d(5) uparuju preduslovi pravila 1b i 2. (to se, naravno ne može reći za pravilo 1a koje sadrži predikat d ali sa različitom konstantom kao argumentom). Na osnovu pravila 1b. dobija se nova činjenica b(5), a pravilo 1b se uklanja iz baze. Na osnovu pravila 2 nastaje novo pravilo

if c(5) then b(5)

koje nema potrebe unositi u bazu pošto već imamo b(5) kao činjenicu. Pravilo 2 se ne može ukloniti iz baze jer je njegov zaključak generalniji. Situacija je sada:

Lista neupotrebljenih činjenica L: b(5)
Produkcioni sistem:
1. if b(x) then a(x)
1a. if d(2) then b(2)
2. if c(x) and d(x) then b(x)
2b. if f(x) then d(x)
4. if g(x) then c(x)
Dobijene činjenice: c(2), c(5), d(5), b(5)

Činjenici b(5) odgovara pravilo 1 koje daje novu činjenicu a(5). Baza znanja se ne menja. S obzirom da se sleđenjem činjenice a(5) ne dobija nikakva nova činjenica, lista neupotrebljenih činjenica je konačno ispražnjena. Završna situacija je, prema tome:

Lista neupotrebljenih činjenica L: -
Produkcioni sistem:
1. if b(x) then a(x)
1a. if d(2) then b(2)
2. if c(x) and d(x) then b(x)
2b. if f(x) then d(x)
4. if g(x) then c(x)
Dobijene činjenice: c(2), c(5), d(5), b(5), a(5)

Zadatak 63: Zaključivanje povratnim ulančavanjem

Posmatrajmo bazu znanja iz postavke zadatka 62:

if b(x) then a(x)

if c(x) and d(x) then b(x)

if e and f(x) then d(x)

if g(x) then c(x)

g(2)

f(5)

g(5)

e

Prikazati proces zaključivanja o istinitosti cilja a(x) povratnim ulančavanjem.

Rešenje

Promenljiva x koja se pojavljuje u ciljnom predikatu a smatra se egzistencijalno kvantifikovanom. Drugim rečima, da bi cilj bio ispunjen, dovoljno je naći jednu konkretnu vrednost C za promenljivu x za koju je predikat a(C) ispunjen.

Povratno rezonovanje je 'ciljno usmereno' jer polazi od zadatog cilja (upita u bazu znanja), upotrebom pravila u smeru od zaključaka ka preduslovima ispituje da li su ispunjene sve činjenice koje zahteva dati cilj. PROLOG programi su produkcioni sistemi, a pri izvršavanju tih programa primenjuje se upravo zaključivanje putem povratnog ulančavanja.

Razmotrimo zaključivanje povratnim ulančavanjem za zadati produkcioni sistem prema algoritmu 11 navedenom u dodatku 1.

Inicijalno, upit a(x) se razmatra kao tekući cilj u proceduri TEST. Tekući (i jedini) predikat cilja je a(x). (U narednoj tabeli, tekući predikat cilja je podvučen). S obzirom da ovaj predikat nije razmatran, promenljiva x nije vezana, a redni broj N stava koji je korišćen u razmatranju tekućeg predikata ima inicijalnu vrednost 0.

tekući cilj C1:	početni upit
	<u>a(x)</u>
N	0
vezivanja	-

Predikat a(x) ne može se upariti ni sa jednom činjenicom. Od pravila je relevantno samo pravilo 1. Zato se uz predikat a(x) pamti redni broj ovoga pravila (potreban je za eventualno vraćanje unazad pri zaključivanju kao što će kasnije biti objašnjeno), a promenljiva x iz stava a(x) vezuje se za promenljivu x iz pravila 1 koja će biti označena kao x' da bi se ove dve promenljive mogle razlikovati.

tekući cilj C1:	početni upit
	<u>a(x)</u>
N	1
vezivanja	x = x'

Prema tački 2.2. algoritma, prelazi se na razmatranje preduslova pravila 1 čime se formira novi tekući cilj C2 i rekurzivno poziva procedura TEST.

tekući cilj C2:	preduslov pravila 1
	<u>b(x')</u>
N	0
vezivanja	-

Razmatranjem stava b(x) uočava se da ovom stavu ne odgovara ni jedna činjenica. Ovom stavu odgovara jedino zaključak pravila 2. Prema tome, uz ovaj stav pamti se redni broj pravila, i vrši vezivanje promenljive x' za promenljivu x iz pravila 2 koju ćemo označiti sa x".

tekući cilj C2:	preduslov pravila 1
	b(x’)
N	2
vezivanja	x’ = x’’

Uzima u razmatranje preduslov pravila 2 koji postaje novi tekući cilj. Sledi novi poziv procedure TEST.

tekući cilj C3:	preduslov pravila 2
	c(x") and d(x")
N	0
vezivanja	-

Sledeći stav koji se razmatra je levi stav u složenom preduslovu pravila 2, a to je c(x). Ovom stavu odgovara pravilo 4. Promenljiva x" vezuje se za promenljivu iz pravila 4 koji ćemo označiti kao x'".

tekući cilj C3:	preduslov pravila 2
	c(x") and d(x")
N	4
vezivanja	x" = x'"

Preduslov g(x) pravila 4 dolazi na red za razmatranje. Sledi poziv procedure TEST za novi cilj.

tekući cilj C4:	preduslov pravila 4
	g(x'")
N	0
vezivanja	-

Stav g(x) uparuje se sa činjenicom g(2) čiji je redni broj N jednak 5 i pri tome promenjiva x'" dobija vrednost 2.

tekući cilj C4:	preduslov pravila 4
	g(x'")
N	5
vezivanja	x'" = 2

Stav g(x) za x=2 je zadovoljen što povlači i zadovoljenost pravila 4 kao tekućeg cilja i povratak iz poslednjeg poziva procedure TEST u prethodni poziv. U prethodnom pozivu, tekući cilj je bio C3. Pošto je tekući predikat c(x") zadovoljen za x" = 2 prelazi se razmatranje predikata d(x") prema tački 4 algoritma.

tekući cilj C3: preduslov pravila 2			
	c(x")	and	d(x")
N	4		0
vezivanja	x" = 2	-	

Pošto se tekući predikat d(x") za x" =2 razmatra prvi put, za njega je N = 0. Tekući predikat ne uparuje ni jedna od činjenica, već samo zaključak pravila 3.

tekući cilj C3: preduslov pravila 2			
	c(x")	and	d(x")
N	4		3
vezivanja	x" = 2	-	

S obzirom da je promenljiva x" vezana za vrednost 2, u preduslovu pravila 3 zamenjuje se promenljiva vrednošću 2. Ovo postaje tekući cilj u novom pozivu procedure TEST.

tekući cilj C4: preduslov pravila 3			
	e	and	f(2)
N	0		
vezivanja	-		

Prvi predikat e u preduslovu pravila 3 zadovoljen je činjenicom iz baze (N=8), pa se prelazi na razmatranje drugog predikata.

tekući cilj C4: preduslov pravila 3			
	e	and	f(2)
N	8		0
vezivanja	-		-

Drugi stav f(2) ne može biti zadovoljen, jer ne postoji činjenica niti pravilo koje može biti upareno sa ovim stavom. Na ovom mestu dolazi do povratka u zaključivanju (*backtracking-a*) prema koraku 3 algoritma. Razmatra se ponovo stav koji se nalazi levo od stava f(2) u preduslovu pravila 3 a to je e.

tekući cilj C4: preduslov pravila 3			
	e	and	f(2)
N	8		0
vezivanja	-		-

Pretraživanjem baze znanja počev od 9. reda (jer je N=8 za tekući predikat) ustanovljavamo da predikat e ne može biti zadovoljen ni činjenicom ni zaključkom pravila. Pošto se radi o krajnje levom predikatu cilja C4, ustanovljava se da cilj C4 nije zadovoljen. Sledi povratak iz poslednjeg poziva procedure TEST u prethodni.

tekući cilj C3: preduslov pravila 2			
	c(x")	and	d(x")
N	4		3
vezivanja	x" = 2	-	

Razmatranjem baze znanja od 4. reda nalazi se da se predikat d(x") ne može se zadovoljiti na alternativan način, pa se zaključivanje vraća na levi stav pravila 2, a to je stav c(x). Pri tome se poništava veza promenljive x" = 2.

tekući cilj C3: preduslov pravila 2			
	c(x")	and	d(x")
N	4		3
vezivanja	-		-

Ne postoji činjenica koja zadovoljava tekući predikat c(x). Zbog toga se razmatraju pravila počev od 4. reda baze znanja prema tački 2.2. pa se bira pravilo 4. Preduslovi pravila 4 već su na ovom nivou bili razmatrani kao cilj C4. Promenljiva x" vezuje se za promenljivu x"' iz pravila 4. Sledi novi poziv procedure test za C4 kao tekući cilj.

tekući cilj C4: preduslov pravila 4	
	g(x"')
N	5
vezivanja	x"' = 2

Pošto se prethodno vezivanje x"' = 2 raskida, pretraživanje baze znanja kreće od šestog reda prema tački 2.1. algoritma. Pri tome se pronalazi činjenica g(5) i vrši vezivanje promenljive x"' za novu vrednost 5.

tekući cilj C4: preduslov pravila 4	
	g(x"')
N	8
vezivanja	x"' = 5

Cilj C4 je zadovoljen, kontrola se vraća prethodnom pozivu procedure TEST i tekući cilj postaje C3. Promenljiva x" dobija vrednost 5.

tekući cilj C3: preduslov pravila 2			
	c(x")	and	d(x")
N	4		3
vezivanja	x" = 5	-	

Razmatra se drugi stav pravila 2, a to je d(x") za x" =5.

tekući cilj C3: preduslov pravila 2			
	c(x")	and	d(x")
N	4		3
vezivanja	x" = 5	-	

Pošto činjenice ne mogu upariti tekući predikat, razmatraju se pravila počev od trećeg reda bazu znanja. Ovaj predikat uparuje zaključak pravila 3. Sledi novi poziv procedure TEST pri čemu preduslov pravila 3 uz smenu x = 5 postaje novi cilj C6. Ovaj cilj se razlikuje od cilja C5 jer je promenljiva x zamenjena drugom vrednošću; prema tome je N = 0, to jest, razmatranje tekućeg predikata počinje od prvog reda baze znanja.

tekući cilj C6: preduslov pravila 3			
	e	and	f(5)
N	0		
vezivanja	-		

Prvi predikat e u preduslovu pravila 3 zadovoljen je činjenicom iz baze (N=8), pa se prelazi na razmatranje drugog predikata.

tekući cilj C6: preduslov pravila 3			
	e	and	f(5)
N	8		0
vezivanja	-		-

Tekući predikat f(5) zadovoljen je činjenicom 6.

tekući cilj C6: preduslov pravila 3			
	e	and	f(5)
N	8		6
vezivanja	-		-

Prema tome, cilj C6 je potpuno zadovoljen, pa se kontrola vraća prethodnom pozivu procedure TEST.

tekući cilj C3: preduslov pravila 2			
	c(x")	and	d(x")
N	4		3
vezivanja	x" = 5	-	

Tekući cilj C3 je u potpunosti zadovoljen, pa se kontrola vraća na prethodni nivo.

tekući cilj C2:	preduslov pravila 1
	b(x')
N	2
vezivanja	x' = x"

U tekućem cilju C2 zadovoljen je predikat b(x') pri čemu je x' = x" = 5. Time je zadovoljeno i pravilo jedan pa se kontrola vraća prvom pozivu procedure TEST.

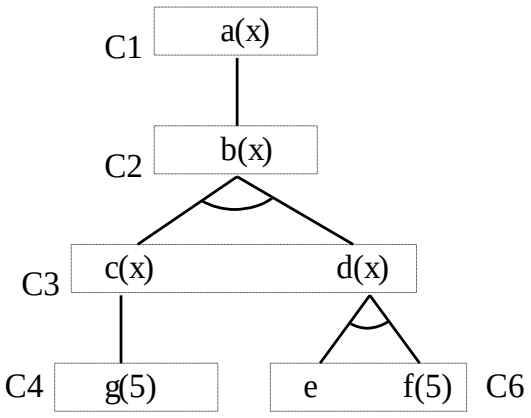
tekući cilj C1:	početni upit
	a(x)
N	1
vezivanja	x = x'

Promenljiva x dobija vrednost 5, što znači da je zadovoljen i krajnji cilj a(x), za x=5.

U procesu zaključivanja, redom su zadovoljavana pravila: 4 (za x=2), 4 (za x=5), 3, 2, 1 i početni upit. Zaključak svakog od ovih pravila generisao je po jednu novu činjenicu: c(2), c(5), d(5), b(5) i a(5). Prema tome, dobijene su iste činjenice u istom redosledu kao u zadatku 62 u kome je primenjeno zaključivanje direktnim ulančavanjem u istoj bazi znanja. U opštem slučaju broj i redosled dobijenih činjenica ne moraju se poklapati kod ova dva načina zaključivanja.

U algoritmu zaključivanja povratnim ulančavanjem navedeno je (u koraku 6. procedure TEST) da je potrebno pamtiti zadovoljene ciljeve zbog mogućnosti njihovog ponovnog razmatranja prilikom vraćanja pri zaključivanju. Pogodna struktura podataka je AND/OR stablo. Svaki čvor stabla odgovara jednom zadovoljenom predikatu u ciljevima. Konektori su označeni pravilima kojima su zadovoljeni pojedini predikati ciljeva. Svaki cilj predstavlja skup onih čvorova stabla koji svi imaju isti ulazni konektor.

Na primer, AND/OR stablo za dati problem po završetku rada algoritma ima izgled prikazan na slici 101 punim linijama. Isprekidanim linijama prikazani su pojedini podciljevi. Primetiti da u stablu nema cilja C5 jer on nije zadovoljen tokom zaključivanja.



Slika 101

Zadatak 64: Zaključivanje cikličkim hibridnim ulančavanjem

Posmatrajmo bazu znanja iz postavke zadatka 62:

if b(x) then a(x)

if c(x) and d(x) then b(x)

if e and f(x) then d(x)

if g(x) then c(x)

g(2)

f(5)

g(5)

e

Koje sve nove činjenice ili pravila i po kom redosledu proizilaze iz ove baze znanja ako se primenjuje *ciklično hibridno ulančavanje* (engl. rule cycle hybrid)?

Rešenje

Pri zaključivanju cikličkim hibridnim ulančavanjem (algoritam 13 iz dodatka 1), ciklički se ponavljaju sledeće akcije: vrši se razmatranje pravila po redosledu kojim su zadata u bazi znanja; ako svi predikati iz preduslova nekog pravila uparuju činjenice iz baze, pravilo uspeva i njegova leva strana (uz odgovarajuće smene promenljivih) dodaje se u bazu kao nova činjenica.

Razmotrimo zadati produkcioni sistem. U prvoj iteraciji algoritma:

- Preduslovi pravila 1 i 2 nisu ispunjeni u prvom prolazu.
- Preduslov pravila 3 jeste ispunjen pošto se može upariti sa činjenicama e i f(5), pa se na čelo liste činjenica dodaje d(5). Pravilo 3 se pri tom ne eliminiše jer se njegov zaključak ne sadrži u dobijenoj činjenici.
- Preduslov pravila 4 može se upariti sa činjenicama g(2), pa zatim i g(5) pa se činjenicama dodaju dve nove c(2) i c(5) a pravilo 4 ostaje u razmatranju.

Na kraju prve iteracije baza znanja ima sledeći izgled:

Pravila

if b(x) then a(x)

if c(x) and d(x) then b(x)

if e and f(x) then d(x)

if g(x) then c(x)

Činjenice:	početne	dobijene
	g(2)	d(5)
	f(5)	c(2)
	g(5)	c(5)
	e	

U drugoj iteraciji, zaključivanje se odvija na sledeći način:

- pravilo 1 ne uspeva.
- pravilo 2 uspeva za x = 5, pa se u bazu dodaje nova činjenica b(5).
- Razmatranjem pravila 3 i 4 ne dobijaju se nove činjenice.

Na kraju druge iteracije baza znanja ima sledeći izgled:

Pravila

if b(x) then a(x)
if c(x) and d(x) then b(x)
if e and f(x) then d(x)
if g(x) then c(x)

Činjenice:	*početne*	*dobijene(1)*	*dobijene(2)*
	g(2)	d(5)	b(5)
	f(5)	c(2)	
	g(5)	c(5)	
	e		

U trećoj iteraciji:

- pravilo 1 uspeva i bazi se dodaje činjenica a(5).
- Ostala pravila ne generišu nove činjenice.

Na kraju treće iteracije baza znanja ima sledeći izgled:

Pravila

if b(x) then a(x)
if c(x) and d(x) then b(x)
if e and f(x) then d(x)
if g(x) then c(x)

Činjenice:	*početne*	*dobijene(1)*	*dobijene(2)*	*dobijene(3)*
	g(2)	d(5)	b(5)	a(5)
	f(5)	c(2)		
	g(5)	c(5)		
	e			

U sledećoj iteraciji, nijedno pravilo ne generiše nove činjenice i time je zaključivanje završeno. Zaključivanjem su dobijene činjenice d(5), c(2), c(5), b(5) i a(5), tim redom. U poređenju sa zaključivanjem uz korišćenje strategije direktnog ulančavanja u istoj bazi znanja (zadatak 62), radi se o istim činjenicama ali je redosled dobijanja drugačiji.

Redosled činjenica, pravila, zatim redosled predikata u preduslovima pravila kao i u upitima ima u veći ili manji uticaj na tok pojedinih vrsta zaključivanja. Tabela 6 sumarno pokazuje koliki uticaj imaju ovi redosledi pri čemu su oni poređani po opadajućem značaju.

Po načinu upotrebe pravila - od preduslova ka zaključku - zaključivanje putem cikličkog ulančavanja podseća na direktno ulančavanje. Iz tabele 6 se međutim vidi da je kod cikličkog hibridnog ulančavanja redosled pravila značajniji od redosleda činjenica kao što je to slučaj i kod povratnog ulančavanja, a suprotno direktnom ulančavanju.

Povratno ulančavanje	Direktno ulančavanje	Cikličko hibridno ulančavanje
predikati upita	činjenice	pravila
pravila	pravila	predikati u IF delovima
predikati u IF delovima	predikati u IF delovima	činjenice
činjenice		

Tabela 6

Zadatak 65: Negacije i zaključivanje povratnim i direktnim ulančavanjem

Baza znanja sadrži sledeća pravila:

R1: if fact1 then goal1

R2: if a and b then goal1

R3: if c(x) then goal2(x)

R4: if not(d) then a

R5: if d then b

R6: if not(e) then c(2)

R7: if fact2 and fact3 then d

R8: if fact2 and fact4 then e

Činjenice:

fact2

fact3

a) Koristeći povratno ulančavanje ispitati istinitost ciljeva goal1 i goal2(x)

b) Koristeći direktno ulančavanje sa fokusiranjem pažnje na nove činjenice odrediti sve moguće zaključke.

Rešenje

U zadatom produkcionom sistemu ciljni predikati su goal1 i goal2(x), pretpostavke su fact1, fact2, fact3 i fact 4 a svi ostali predikati spadaju u međupredikate. U okviru pravila pojavljuju se i negacije predikata. Ove negacije tretiraju se u skladu sa *pretpostavkom o zatvorenom*

svetu (engl. *closed world assumption*) da je negacija predikata tačna ako sa datim činjenicama ne možemo utvrditi istinitost traženog predikata (drugim rečima, ako nema dovoljno informacija da se utvrdi da je nešto istinito, uzimamo je da njegova negacija istinita). PROLOG u zaključivanju koje, kao što je već rečeno, vrši povratnim ulančavanjem, takođe koristi pretpostavku o zatvorenom svetu.

Zaključivanje se vrši prema algoritmu 11 iz dodatka 1 sa dodatnim pravilom vezanim za negirane predikate u ciljevima:

- U koraku 2 algoritma dodaje se: Ako se u tekućem cilju tekući predikat P pojavljuje negiran, onda je not(P) zadovoljeno ako i samo ako P nije zadovoljen.

Drugim rečima, ispitujemo ispunjenost predikata P, pa ako utvrdimo da je predikat P ispunjen, znači da nije ispunjena njegova negacija i obrnuto, ako P nije ispunjen, znači da je ispunjena njegova negacija.

a) Zaključivanje sa datim pravilima obavlja se sledećim redosledom ciljeva:

- goal1; nema ga u činjenicama; razmatra se pravilo R1 i njegov preduslov
- fact1; u činjenicama ga nema, a nema ni pravila; fact1 nije zadovoljeno, a time ni pravilo R1
- goal1; ostalo je još pravilo R2; razmatra se njegov preduslov
- a; nema ga u činjenicama; razmatra se pravilo R4; potrebno je razmotriti not(d)
- d; nema ga u činjenicama; razmatra se pravilo R7
- fact2 jeste činjenica
- fact3 jeste činjenica pa pravilo R7 uspeva; d je zadovoljeno; not(d) nije zadovoljeno; R4 ne uspeva; R2 ne uspeva;
- goal1; nema više nerazmotrenih pravila; prvi cilj goal1 nije zadovoljen; ostaje drugi cilj goal2(x) za razmatranje
- goal2(x); nema ga u činjenicama; razmatra se pravilo R3
- c(x); nema ga u činjenicama; razmatra se pravilo R6; potrebno je razmotriti not(e)
- e; nema ga u činjenicama; razmatra se R8
- fact2 je činjenica
- fact4; nema ga u činjenicama i ne pojavljuje se u zaključcima pravila pa nije ispunjeno; pravilo R8 ne uspeva
- e; nije ispunjeno pošto više nema pravila za razmatranje; not(e) je ispunjeno; uspeva R6; uspeva c(x) za x=2; uspeva R3 za x=2; uspeva goal2(2).

U toku zaključivanja dobijene su sledeće nove činjenice prema redosledu dobijanja: d, not(e), c(2), goal2(2).

b) Kod direktnog ulančavanja (algoritam 12 iz dodatka 1) potrebno je pravila sa negacijama predikata preskočiti pri razmatranju sve dok se ostalim pravilima ne zaključe sve moguće činjenice.

Tok zaključivanja u zadatom produkcionim sistemu je sledeći (redom su navedeni činjenica koja se u datom trenutku sledi, pravilo koje se pri tom razmatra i akcija koja se preduzima kao rezultat razmatranja pravila):

Korak	Lista činjenica	Pravilo	Akcija
1.	fact2, fact3	R7	Briše se R7, dodaje se R9: if fact3 then d
2.	fact2, fact3	R8	Briše se R8, dodaje se R10: if fact4 then e

[činjenica fact2 je razmotrena, nisu dodate nove činjenice, pa se prelazi na sledeću zadatu činjenicu]

3.	fact3	R9	d je nova činjenica; R9 se briše
4.	d	R5	b je nova činjenica; R5 se briše

[pravilo R4 se u ovom trenutku ignoriše sve dok ima činjenica za razmatranje]

5.	b	R2	R2 se briše; dodaje se R11: if a then goal1

U ovom trenutku zaključivanja produkcioni sistem ima sledeći izgled:

R1: if fact1 then goal1

R11: if a then goal1

R3: if c(x) then goal2(x)

R4: if not(d) then a

R6: if not(e) then c(2)

R10: if fact4 then e

Nema više novih činjenica za razmatranje, pa treba razmotriti pravila sa negacijama predikata. not(d) nije ispunjeno jer je d činjenica, a not(e) je ispunjeno jer e nije činjenica. Prema tome, not(e) je nova činjenica za razmatranje.

Korak	Lista činjenica	Pravilo	Akcija
6.	not(e)	R7	R7 se briše; c(2) je nova činjenica
7.	c(2)	R3	goal2(2) je nova činjenica; R3 se ne briše
8.	goal2(2)	-	nema pravila za razmatranje, niti nerazmotrenih činjenica, pa je zaključivanje okončano

Zaključivanjem su dobijene sledeće činjenice (prema redosledu dobijanja): d, b, not(e), c(2), goal2(2). U odnosu na tačku a) dobijena je jedna činjenica više - činjenica b. U zaključivanju povratnim ulančavanjem razmatra se ispunjenost samo onih predikata koji mogu uticati na ispunjenost zadatog cilja, dok se kod direktnog ulančavanja izvode svi mogući zaključci na osnovu zadate baze znanja.

Zadatak 66: Negacije i zaključivanje direktnim i cikličkim hibridnim ulančavanjem

Posmatrajmo sledeću bazu znanja koja se sastoji od produkcionih pravila i činjenica.

1. if v and t then a
2. if b and u and not(t) then a
3. if n(x) and b then m(x)
4. if c then b
5. if r and s then t
6. if v and r then u
7. r
8. v
9. c
10. n(12)

a) Navesti sve nove činjenice ili pravila izvedena direktnim ulančavanjem prema redosledu njihovog dobijanja. Pravila sa not ostavljaju se za kraj.

b) Naći sve nove činjenice ili pravila izvedena iz cikličkog hibridnog ulančavanja prema redosledu njihovog dobijanja. Pravila sa not ostavljaju se za kraj.

c) Problem sa cikličkim hibridnim ulančavanjem je u tome što se njime ponavljaju ispitivanja istih pravila u svakom ciklusu. Opisati način na koji bi se izbeglo ispitivanje nekih pravila na osnovu toga šta se desilo u prethodnom ciklusu.

Rešenje

a) Inicijalno stanje je:

Lista neupotrebljenih činjenica: r, v, c, n(12).

Produkcioni sistem:

1. if v and t then a
2. if b and u and not(t) then a
3. if n(x) and b then m(x)
4. if c then b
5. if r and s then t
6. if v and r then u

Dobijene činjenice: -

Tok zaključivanja (prema algoritmu 12 u dodatku 1) je sledeći:

Sledi se prva od datih činjenica r; pravila 5 i 6 se brišu a nastaju pravila 5a (if s then t) i 6a (if v then u)

Lista neupotrebljenih činjenica: v, c, n(12).

Produkcioni sistem:

1. if v and t then a
2. if b and u and not(t) then a
3. if n(x) and b then m(x)
4. if c then b

5a. if s then t

6a. if v then u

Dobijene činjenice: -

Sledi se v; pravilo 1 se briše, a dodaje se 1a (if t then a); od pravila 6a, koje se briše, nastaje nova činjenica u.

Lista neupotrebljenih činjenica: u, c, n(12).

Produkcioni sistem:

1a. if t then a

2. if b and u and not(t) then a
3. if n(x) and b then m(x)
4. if c then b

5a. if s then t

Dobijene činjenice: u

Sledi se u; od pravila 2 (koje se briše) nastaje pravilo 2a (if b and not(t) then a). Treba uočiti da pravilo smo pravilo 2 razmatrali iako ono sadrži predikat sa operatorom not. Radi se, međutim o predikatu not(t), tako da bismo pravilo 2 isključili iz razmatranja jedino u ako bismo smo sledili činjenicu t što ovde nije bio slučaj.

Lista neupotrebljenih činjenica: c, n(12).

Produkcioni sistem:

1a. if t then a

2a. if b and not(t) then a
3. if n(x) and b then m(x)
4. if c then b

5a. if s then t

Dobijene činjenice: u

Sledi se c i na osnovu pravila 4, koje se briše, nastaje nova činjenica b.

Lista neupotrebljenih činjenica: b, n(12).

Produkcioni sistem:

1a. if t then a

2a. if b and not(t) then a

3. if n(x) and b then m(x)
5a. if s then t
Dobijene činjenice: u, b

Sledi se b; od pravila 2a (koje se briše) nastaje 2b (if not(t) then a); od pravila 3 (briše se) nastaje pravilo 3a (if n(x) then m(x)).

Lista neupotrebljenih činjenica: n(12).
Produkcioni sistem:
1a. if t then a
2b. if not(t) then a
3a. if n(x) then m(x)
5a. if s then t
Dobijene činjenice: u, b

Sledi se poslednja od datih činjenica n(12); pravilo 3a dokazuje činjenicu m(12).

Lista neupotrebljenih činjenica: m(12).
Produkcioni sistem:
1a. if t then a
2b. if not(t) then a
3a. if n(x) then m(x)
5a. if s then t
Dobijene činjenice: u, b, m(12)

Sledi se m(12); ne dobijaju se nove činjenice niti pravila.

Pošto su razmotrene sve činjenice, prelazi se na razmatranje pravila sa not. Pošto među zadatim i dobijenim činjenicama nema činjenice t, not(t) je nova činjenica koju treba razmotriti.

Lista neupotrebljenih činjenica: not(t).
Produkcioni sistem:
1a. if t then a
2b. if not(t) then a
3a. if n(x) then m(x)
5a. if s then t
Dobijene činjenice: u, b, m(12), not(t)

Sledi se not(t); Pravilo 2b (koje se briše) dokazuje činjenicu a.

Lista neupotrebljenih činjenica: a.

Produkcioni sistem:
1a. if t then a
3a. if n(x) then m(x)
5a. if s then t
Dobijene činjenice: u, b, m(12), not(t), a

Sledi se a; ne dobijaju se nove činjenice niti pravila; zaključivanje je završeno.

Prema tome, zaključivanjem su dobijene nove činjenice ovim redom: u, b, m(12), not(t), a. Samo predikati za koje je utvrđeno da su tačni su navedeni u ovoj listi. Za sve ostale predikate, prema pretpostavki o zatvorenom svetu, važi njihova negacija; t je samo jedan od takvih predikata čiju smo negaciju i eksplicitno koristili da bismo utvrdili važenje drugih predikata.

b) Tok zaključivanja cikličkim hibridnim ulančavanjem sa datim produkcionim sistemom je sledeći:

Preduslovi pravila 1, 3 nisu ispunjeni (bar po jedan predikat iz preduslova ovih pravila ne nalazi se među činjenicama). Pravilo 2 se ignoriše jer sadrži not.

Pravilo 4 je zadovoljeno, pa se ono briše, a dobija se nova činjenica b.

Pravilo 5 nije zadovoljeno.

Poslednje, šesto pravilo je zadovoljeno (i v i r su činjenice). Pravilo se briše, a činjenicama se dodaje u.

Na kraju iteracije, pravila su:		
1. if v and t then a		
2. if b and u and not(t) then a		
3. if n(x) and b then m(x)		
5. if r and s then t		
Činjenice su:	početne	dobijene
	r, v, c, n(12)	b, u

Pri drugom prolasku kroz listu pravila, prva dva pravila ponovo nisu zadovoljena.

Treće pravilo je zadovoljeno za x=12. Činjenicama se dodaje m(12), a pravilo se ne briše jer ima generalniji zaključak.

Peto pravilo nije zadovoljeno.

Na kraju iteracije, pravila su:
1. if v and t then a
2. if b and u and not(t) then a
3. if n(x) and b then m(x)
5. if r and s then t

Činjenice su:	početne	dobijene
	r, v, c, n(12)	b, u, m(12)

8. U sledećem prolasku ne dobija se nijedna nova činjenica, pa je potrebno razmotriti negirane predikate u preduslovima pravila. U pravilu 2, not(t) je ispunjeno jer se t ne nalazi među činjenicama.
9. Pravilo 1 u sledećem prolazu nije zadovoljeno.
10. Pravilo 2 sada se razmatra i zadovoljeno je, pa se ono uklanja i utvrđuje nova činjenica a.
11. Pravilo 3 je zadovoljeno za x=12, ali to ne dovodi do dobijanja nove činjenice.
12. Pravilo 5 nije zadovoljeno.

Na kraju iteracije, pravila su:		
	1. if v and t then a	
	3. if n(x) and b then m(x)	
	5. if r and s then t	
Činjenice su:	početne	dobijene(I)
	r, v, c, n(12)	b, u, m(12), not(t), a

13. U sledećem prolazu kroz preostala pravila 1, 3 i 5 ne dobija se nijedna nova činjenica, pa je proces zaključivanja okončan.

Prethodnim postupkom dobijene su nove činjenice: b, u, m(12), not(t), a. Dobijeni su svi mogući zaključci kao i u slučaju pod a), ali je redosled dobijanja različit.

c) U algoritam 13 koji je dat u dodatku 1 već je ugrađena svojevrsna optimizacija kroz ideju o brisanju pravila koje ne sadrži promenljive po njegovoj uspešnoj primeni. Mana datog algoritma je što u svakom ciklusu razmatra sva preostala pravila. Ukoliko pravilo P u prethodnom ciklusu nije bilo zadovoljeno, a nije dobijena nijedna od činjenica koja bi se mogla upariti sa predikatima u if delu pravila P, onda ovo pravilo sigurno neće biti zadovoljeno ni u tekućem ciklusu. *Indeksiranjem pravila* prema imenima predikata u preduslovima pravila, omogućuje se da se u tekućem ciklusu pronađu i razmotre samo ona pravila koja sadrže predikat koji je u prethodnom ciklusu razmatranja pravila utvrđen kao činjenica.

Na primer, za indeksiranje pravila u bazi znanja iz ovog zadatka koristi se tabela 7. Svaka vrsta označena je jednim od predikata iz preduslova pravila. U vrsti koja odgovara predikatu P nalazi se lista rednih brojeva svih pravila koja sadrže predikat P (ili njegovu negaciju) u preduslovu. Prema tome, da bi se razmotrila pravila koja sadrže u preduslovu predikat P, dovoljno je pronaći odgovarajuću vrstu u tabeli i razmotriti sva pravila iz te vrste.

Predikat	Lista pravila
b	2, 3
c	4
n(x)	3
r	5, 6
s	5
t	1, 2
u	2
v	1, 6

Tabela 7

Takođe se mogu grupisati pravila sa negacijama predikata u preduslovu, da bi se ona u drugoj fazi algoritma (kada se razmatraju negacije predikata) mogla efikasno obraditi.

Zadatak 67: Povratno ulančavanje i pamćenje zaključaka

Dat je produkcioni sistem:

R1: if j(x) and b(x) then k(x)

R2: if a(x) and not g(x) then f(x)

R3: if b(x) and m then a(x)

R4: if i then d

R5: if e(x) and c then d

R6: if a(x) and h then g(x)

R7: if m then g(x)

R8: if c then b(x)

Pretpostavimo da su ciljevi f(x), d i k(x), u tom redosledu, a činjenice c, m, e(a) i j(b), respektivno. Zaključivanje se okončava čim se dokaže neki od ciljeva.

a) Pretpostavimo da se zaključivanje obavlja direktnim ulančavanjem. Navesti redosled razmatranja pravila.

b) Navesti redosled razmatranja pravila pri zaključivanju povratnim ulančavanjem ako nema pamćenja (engl. *caching*) zaključaka. Koji je redosled razmatranja pravila ako se upotrebljava pamćenje zaključaka?

c) Da li redosled činjenica utiče na to koji cilj će biti prvi dokazan u tački a)? Objasniti zaključak.

d) Da li redosled činjenica utiče na to koji cilj će biti prvi dokazan u tački b)? Objasniti zaključak.

Rešenje

Pamćenje zaključaka pri zaključivanju povratnim ulančavanjem znači da se zadovoljeni predikati iz zaključaka pravila dodaju u bazu znanja kao činjenice. Pamćenje zaključaka ne utiče na rezultate procesa zaključivanja, već ima za cilj da poveća performanse na taj način što se za zapamćene predikate odmah utvrđuje zadovoljenost u naknadnim razmatranjima, bez potrebe da se nanovo razmatraju pravila koja ih zadovoljavaju.

a) Pri direktnom ulančavanju zaključujemo polazeći od činjenica ka ciljevima, sledećim redom:

1. Razmatramo prvu od datih činjenica c. Uparuju je preduslovi pravila R5 i R8. R5 se uprošćava u oblik (if e(x) then d) a R8 je potpuno zadovoljeno pa se eliminiše i dobijamo novu činjenicu b(x). Produkcioni sistem sada ima sledeći izgled:

   R1: if j(x) and b(x) then k(x)

   R2: if a(x) and not g(x) then f(x)

   R3: if b(x) and m then a(x)

   R4: if i then d

   R5: if e(x) then d

   R6: if a(x) and h then g(x)

   R7: if m then g(x)

2. Sledimo b(x). Uparuju je pravila R1 i R3 koja dobijaju uprošćene oblike (if j(x) then k(x)) i (if m then a(x)) respektivno.

   R1: if j(x) then k(x)

   R2: if a(x) and not g(x) then f(x)

   R3: if m then a(x)

   R4: if i then d

   R5: if e(x) then d

   R6: if a(x) and h then g(x)

   R7: if m then g(x)

3. Razmatramo sledeću od datih činjenica a to je m. Prvo pravilo koje je uparuje je uprošćeno pravilo R3. Ono se briše i nastaje nova činjenica a(x). Drugo pravilo koje uparuje m je R7. I ono se briše, a nova činjenica g(x) dodaje se u listu činjenica za razmatranje ispred činjenica a(x).

   R1: if j(x) then k(x)

   R2: if a(x) and not g(x) then f(x)

   R4: if i then d

   R5: if e(x) then d

   R6: if a(x) and h then g(x)

4. Sledimo g(x). Pravilo R2 se ignoriše jer ima not. Nema drugih pravila koja uparuju g(x).

5. Sledimo a(x). Pravilo R2 se uprošćava u oblik (if not g(x) then f(x)) a pravilo R6 u oblik (if h then g(x)).

   R1: if j(x) then k(x)

   R2: if not g(x) then f(x)

   R4: if i then d

   R5: if e(x) then d

   R6: if h then g(x)

6. Sledimo datu činjenicu e(a). Uparuje je uprošćeno pravilo R5; ovo pravilo se briše a predikat d dodaje činjenicama. Pošto se d nalazi u listi ciljeva, zaključivanje je završeno.

   R1: if j(x) then k(x)

   R2: if not g(x) then f(x)

   R4: if i then d

   R6: if h then g(x)

Zaključivanjem su dobijene činjenice: b(x), a(x), g(x) i d, tim redosledom.

b) Prvo ćemo sprovesti zaključivanje od ciljeva ka činjenicama bez pamćenja zaključaka:

1. Prvi cilj je f(x). Među činjenicama nema ovog cilja a može ga zadovoljiti pravilo R2, pa treba razmotriti preduslov ovoga pravila.
2. Razmatramo a(x) i pravilo R3 (jer među činjenicama nemamo a(x)).
3. Razmatramo b(x) i pravilo R8.
4. Razmatramo c koje se nalazi među činjenicama. Prema tome, pravilo R8 je zadovoljeno kao i podcilj b(x).
5. Nastavljamo razmatranje pravila R3 i u skladu sa tim predikat m. Radi se o činjenici, tako da su pravilo R3 i podcilj a(x) zadovoljeni.
6. Nastavljamo sa razmatranjem pravila R2. Da bismo utvrdili zadovoljenost podcilja not g(x) razmatraćemo podcilj g(x) pa ako on nije zadovoljen, njegova negacija jeste po pretpostavci o zatvorenom svetu. Prvo od pravila koje zadovoljavaju g(x) je R6 pa ćemo razmotriti njegove preduslove.
7. Prvi preduslov pravila R6 je a(x). Pošto ne pamtimo zaključke, činjenica da je ovaj cilj već razmatran nema uticaja u ovom trenutku. To znači da će se ponoviti razmatranje pravila R3 i R8 da bi se ponovo došlo da zaključka da je a(x) zadovoljen.
8. Nastavljamo razmatranje pravila R6. Drugi preduslov je h. Ovog predikata nema među činjenicama niti ga može zadovoljiti neko od pravila. Prema tome h, kao ni pravilo R6 nisu zadovoljeni.
9. Treba pokušati zadovoljiti g(x) na alternativan način. Drugo pravilo koje ga ima u zaključku je R7.
10. Preduslov pravila R7 je predikat m i on se nalazi među činjenicama. Prema tome pravilo R7 je zadovoljeno a time i g(x). To znači da not g(x) nije zadovoljeno pa pravilo R2 nije

zadovoljeno. Pošto više nema pravila koja mogu zadovoljiti cilj f(x), zaključujemo da ovaj cilj nije zadovoljen.

11. Prelazimo na razmatranje drugog cilja d. Prvo od pravila koja zadovoljavaju ovaj cilj je R4.

12. Preduslov i pravila R4 nije činjenica niti se može zadovoljiti upotrebom pravila pa pravilo R4 ne može biti zadovoljeno. Sledeće pravilo koje može zadovoljiti cilj d je R5.

13. Oba preduslova pravila R5 se mogu upariti činjenicama tako da je R5 zadovoljeno, a time i cilj d čime se zaključivanje završava.

Zaključivanjem su dobijene činjenice: b(x), a(x), g(x) i d, tim redosledom.

Razmotrimo sada varijantu zaključivanja povratnim ulančavanjem kada postoji pamćenje zaključaka, što znači da se zadovoljeni podciljevi u toku zaključivanja dodaju u bazu znanja kao činjenice.

U konkretnom slučaju tok zaključivanja u situaciji kada se primenjuje pamćenje zaključaka je sličan varijanti bez pamćenja, s jedinom razlikom što se u tački 7 ne razmatraju pravila R3 i R8 jer se odmah detektuje da se a(x) nalazi među činjenicama.

c) Redosled činjenica ima uticaja na tok zaključivanja direktnim ulančavanjem (pogledati diskusiju u zadatku 64). Na primer, kada bismo obrnuli redosled činjenicama e(a) i j(b) tako da se prvo razmatra j(b), prvo bi bio dokazan cilj k(b).

d) U datom slučaju redosled činjenica nema uticaja na tok zaključivanja povratnim ulančavanjem. Ovaj redosled može biti od uticaja jedino u situaciji kada postoje različite činjenice s istoimenim predikatom, tako da u cilju postoji promenljiva za koju se mogu vezati različite vrednosti u zavisnosti od redosleda činjenica.

U komercijalnim ekspertskim sistemima koji primenjuju zaključivanje povratnim ulančavanjem, koriste se i takozvane *virtuelne činjenice*. To znači da se od korisnika ne traži da unese sve činjenice pre početka zaključivanja, već se u toku procesa zaključivanja korisniku postavlja pitanje u vezi sa nekim predikatom tek u trenutku kada on postane tekući cilj, čime se izbegava da se korisniku postavljaju pitanja koja nisu relevantna za konkretnu situaciju. Ovo je značajna prednost povratnog ulančavanja nad direktnim, s obzirom da je u slučaju direktnog ulančavanja neophodno imati na raspolaganju sve činjenice unapred. Kod direktnog ulančavanja, pravila je moguće razdeliti u više domena, tako da je u jednom trenutku samo jedan domen aktivan. Na osnovu zaključaka toga domena aktivira se neki od sledećih domena. Tada korisnik unapred mora da odgovori samo za pitanja vezana za konkretni domen, čime se ublažava ta mana direktnog ulančavanja.

Zadatak 68: Broj iteracija u cikličkom hibridnom ulančavanju

Pretpostavimo da vršimo cikličko hibridno ulančavanje sa R pravila. U predikatima pravila se ne pojavljuju promenljive niti se pojavljuju negirani predikati. Pretpostavimo da u preduslovima pravila ima S različitih, od ukupno T predikata. U zaključcima pravila ima L različitih predikata. Neka je dato F činjenica, gde je $F > 0$. Koliki je maksimalan broj iteracija neophodan da se dođe do svih mogućih zaključaka?

Rešenje

Pri cikličkom hibridnom ulančavanju u svakoj iteraciji prolazi se kroz sva pravila, tražeći ona čiji su preduslovi u potpunosti zadovoljeni činjenicama. Od zaključaka takvim pravila nastaju nove činjenice. U svakoj iteraciji mora se dobiti bar jedna nova činjenica; u suprotnom se zaključivanje obustavlja. Pošto ima L različitih zaključaka i nema promenljivih u predikatima, najveći broj novih činjenica koje se mogu dobiti tokom zaključivanja je L; u situaciji kada se u svakoj iteraciji zadovoljava po jedno pravilo, ne može biti više od L iteracija. Pored parametra L i parametra S (broj različitih predikata u preduslovima pravila) utiče na maksimalan broj iteracija. Broj iteracija ne može biti veći od S jer da bi se u svakoj iteraciji našao po jedan novi zaključak, mora biti zadovoljen bar po jedan novi predikat u preduslovu. Prema tome, konačni odgovor je da je maksimalan broj iteracija jednak vrednosti manjeg od parametra S i L.

Zadatak 69: Problem izbora pića uz večeru

Upravo ste se spremili za mirno veče kod kuće, kad vam se iznenada javio jedan stari poznanik - dolazi na večeru. Uzbuna! Potrebna je pomoć malog sistema za usklađivanje jelovnika i vinske karte. Pravila su sledeća:

P1:	IF	zahteva se skupo vino I danas je Đurđevdan
	THEN	Osveštano Vino
P2:	IF	zahteva se skupo vino I glavno jelo je prasetina
	THEN	Dingač
P3:	IF	dobro je jeftino vino I glavno jelo je piletina I gost nije osobito omiljen
	THEN	Banatski Rizling
P4:	IF	dobro je jeftino vino I glavno jelo se ne zna
	THEN	Jagodinska Ružica
P5:	IF	dobro je pivo I glavno jelo je kupus
	THEN	Valjevsko Pivo
P6:	IF	dobro je pivo
	THEN	BiP
P7:	IF	gost pazi na zdravu ishranu
	THEN	pahuljice od žitarica
P8:	IF	gost pazi na zdravu ishranu I ne služe se mrkve
	THEN	sok od cvekle
P9:	IF	služi se vino I treba opseniti prostotu
	THEN	zahteva se skupo vino
P10:	IF	služi se vino
	THEN	dobro je jeftino vino

P11:	IF	gost ima prefinjen ukus
	THEN	služi se vino
P12:	IF	prilog je kupus
	THEN	dobro je pivo
P13:	IF	gost nije osobito omiljen I jelo se kupuje u dragstoru
	THEN	dobro je pivo
P14:	IF	.T.
	THEN	voda

Zaključci se ispituju u navedenom redosledu:

- Osveštano Vino
- Dingač
- Jagodinska Ružica
- Banatski Rizling
- Valjevsko pivo
- BiP
- pahuljice od žitarica
- sok od cvekle
- voda

Ako treba, pretpostavlja se da su sledeći iskazi tačni:

- jelo se kupuje u dragstoru
- prilog je kupus
- gost nije osobito omiljen
- gost ima prefinjen ukus
- danas je Đurđevdan
- glavno jelo je piletina

Ako treba pretpostavlja se da sledeći iskazi nisu tačni:

- ne služe se mrkve
- glavno jelo se ne zna
- gost pazi na zdravu ishranu
- treba opseniti prostotu
- glavno jelo je prasetina

a) Simulirati zaključivanje povratnim ulančavanjem na osnovu datih pretpostavki; navesti redosled pravila i izabrano piće.

b) Sa drugim pretpostavkama, da li bi uz kupus mogao da bude preporučen Dingač, i zašto?

c) Sa drugim pretpostavkama, da li bi uz prasetinu mogao da bude preporučen sok od cvekle, i zašto?

Rešenje

a) Razmatramo ispunjenost zaključaka po zadatom redosledu:

- *Osveštano Vino*: ne nalazi se među činjenicama, samo u zaključku pravila P1 pa razmatramo njegov preduslov: *zahteva se skupo vino* I *danas je Đurđevdan.*
- *zahteva se skupo vino*: ne nalazi se među činjenicama, samo u zaključku pravila P9 pa razmatramo njegov preduslov: *služi se vino* I *treba opseniti prostotu.*
- *služi se vino*: ne nalazi se među činjenicama, samo u zaključku pravila P11 pa razmatramo njegov preduslov: *gost ima prefinjen ukus.*
- *gost ima prefinjen ukus*: uparuje ga činjenica; prema tome, pravilo P11 je zadovoljeno, a time i njegov zaključak *služi se vino.* Prelazi se na drugi stav preduslova pravila P9.
- *treba opseniti prostotu*: prema postavci zadatka, ova činjenica nije ispunjena niti se može zaključiti primenom pravila. Prema tome, pravilo P9 nije zadovoljeno. Preduslov pravila P1 *zahteva se skupo vino* ne može se zadovoljiti na drugi način osim primenom pravila P9, pa zaključujemo da ovaj preduslov kao i samo pravilo P1 i zaključak *Osveštano Vino* nisu zadovoljeni. Razmatra se sledeći zaključak sa liste zaključaka:
- *Dingač*: ne nalazi se među činjenicama, samo u zaključku pravila P2 pa razmatramo njegov preduslov: *zahteva se skupo vino* I *glavno jelo je prasetina.*
- *zahteva se skupo vino*: u ranijem razmatranju dobijeno je da se ovaj preduslov ne može ispuniti, pa zaključujemo da se pravilo P2 ne može zadovoljiti a time ni zaključak *Dingač*. Prelazimo na sledeći cilj iz liste zaključaka:
- *Jagodinska Ružica*: ne nalazi se među činjenicama, samo u zaključku pravila P4 pa razmatramo njegov preduslov: *dobro je jeftino vino* I *glavno jelo se ne zna.*
- *dobro je jeftino vino*: ne nalazi se među činjenicama, samo u zaključku pravila P10 pa razmatramo njegov preduslov: *služi se vino.*
- *služi se vino:* u ranijem razmatranju dobijeno je da je ovaj stav zadovoljen. Prema tome, zadovoljen je i zaključak pravila P10: *dobro je jeftino vino.* Razmatra se preostali preduslov pravila P4:
- *glavno jelo se ne zna*: prema postavci, ovaj stav nije zadovoljen niti se može zadovoljiti na osnovu pravila. Prema tome, pravilo P4 nije zadovoljeno a time ni zaključak *Jagodinska Ružica.* Sledeći zaključak sa liste je *Banatski Rizling*.
- *Banatski Rizling*: ne nalazi se među činjenicama, samo u zaključku pravila P3 pa razmatramo njegov preduslov: *dobro je jeftino vino* I *glavno jelo je piletina* I *gost nije osobito omiljen*
- *dobro je jeftino vino*: ranijim razmatranjem utvrđeno je da je ovaj stav ispunjen. Prelazi se na sledeći stav iz preduslova pravila P3.
- *glavno jelo je piletina*: nalazi se među činjenicama. Prelazi se na sledeći stav iz preduslova pravila P3.

- *gost nije osobito omiljen*: nalazi se među činjenicama. Prema tome, zadovoljeno je pravilo P3 kao i zaključak da za piće treba da se posluži *Banatski Rizling* čime je zaključivanje završeno.

b) Da bi za piće bio izabran Dingač, prema pravilu P2 morali bi važiti iskazi *zahteva se skupo vino* i *glavno jelo je prasetina*. Iskaz *zahteva se skupo vino* je međupredikat i proizilazi iz pravila P9, prema tome moraju važiti preduslovi *služi se vino* i *treba opseniti prostotu*. Iskaz *služi se vino* je međupredikat i proizilazi iz pravila P11, pa mora važiti njegov preduslov: *gost ima prefinjen ukus*. Prema tome, da bi se preporučio Dingač, moraju biti ispunjene sledeće pretpostavke:

- *glavno jelo je prasetina*
- *treba opseniti prostotu*
- *gost ima prefinjen ukus*

koje povlače ispunjenje međupredikata:

- *služi se vino*
- *zahteva se skupo vino*

Ove pretpostavke slažu sa pretpostavkom *prilog je kupus*, ali ne i sa pretpostavkom *glavno jelo je kupus*. Potrebno je, na kraju, proveriti da li se zaključak *Osveštano vino* može dobiti iz ovih pretpostaki s obzirom da je prioritet ovog zaključka veći od zaključka *Dingač*. Prema pravilu P1, s obzirom da je ispunjen preduslov *zahteva se skupo vino*, konstatujemo da se navedenim pretpostavkama mora dodati

- *danas nije Đurđevdan*

da bi krajnji zaključak bio *Dingač*, a ne *Osveštano vino*.

c) Sa datim redosledom ispitivanja zaključaka, ne postoji skup pretpostavki pod kojima bi se za piće preporučio sok od cvekle. Da bi se za piće doneo zaključak *sok od cvekle*, morali bi važiti iskazi *gost pazi na zdravu ishranu* i *ne služe se mrkve* iz preduslova pravila P8. Međutim, prema zadatom redosledu zaključaka, zaključak *pahuljice od žitarica* proverava se pre zaključka *sok od cvekle*, tako da se pravilo P7 razmatra pre pravila P8. Preduslov pravila P7, iskaz *gost pazi na zdravu ishranu*, je podskup preduslova pravila P8, tako da će od ova dva pravila za primenu uvek biti birano pravilo P7.

Zadatak 70: Produkcioni sistem za delovanje robota

Razmotrimo sledeći produkcioni sistem za delovanje robota. Neka su sledeće činjenice istinite, prema redosledu:

F1: Desno od robota postoji objekat sa kracima.

F2: Objekat je visok 60 cm.

F3: Objekat zauzima zapreminu od 0.5 m^3.

F4: Objekat je nepokretan.

F5: Drugi objekat se kreće ka robotu.

F6: Od drugog objekta čuje se govor (pretpostavimo da govor jeste nepravilan šum ali ne i glasan šum).

Pretpostavimo da sve ostale činjenice u pravilima nisu istinite. Pretpostavimo, takođe, da se svaka nova činjenica koja nastane zaključivanjem dodaje na čelo date liste činjenica. U produkcionom sistemu moguće su sledeće akcije, po redosledu:

A1: Okrenuti se za 180 stepeni.

A2: Stati i čekati.

A3: Okrenuti se ka nečemu.

A4: Pomeriti se unapred za malo rastojanje.

A5: Okrenuti se za 20 stepeni na desno.

A6: Pomeriti se unapred za veliko rastojanje.

Pravila su:

R1:	IF	čuje se glasan šum ispred robota
	THEN	okrenuti se za 180 stepeni i pomeriti se za veliko rastojanje.
R2:	IF	treba se sakriti a postoji žbun u blizini
	THEN	okrenuti se ka žbunu i pomeriti se za malo rastojanje
R3:	IF	treba se sakriti a robot je u žbunju
	THEN	stati i čekati
R4:	IF	objekat se kreće ka robotu i objekat je osoba ili vozilo
	THEN	treba se sakriti
R5:	IF	objekat se kreće ka robotu i radi se o životinji
	THEN	stati i čekati
R6:	IF	predmet je prepreka i robot se kreće i predmet mu blokira put
	THEN	okrenuti se za 20 stepeni i pomeriti se za kratko rastojanje
R7:		Pomeriti se unapred za veliko rastojanje (nema IF dela).
R8:	IF	objekat ima duge krake i kraci se kreću i objekat nema točkove
	THEN	radi se o životinji
R9:	IF	objekat odaje nepravilne šumove
	THEN	radi se o životinji
R10:	IF	objekat odaje pravilne šumove i kreće se
	THEN	radi se o vozilu
R11:	IF	objekat ima točkove
	THEN	radi se o vozilu
R12:	IF	objekat se ne kreće i zauzima više od 0,03 m^3 zapremine
	THEN	radi se o prepreci

R13:	IF	prepreka ima krake i niža je od 90 cm
	THEN	radi se o žbunu
R14:	IF	prepreka ima krake i viša je od 90 cm
	THEN	radi se o drvetu
R15:	IF	prepreka nema krake
	THEN	radi se o steni
R16:	IF	životinja ima četiri kraka u po dva para i drži se na jednom paru
	THEN	radi se o osobi
R17:	IF	životinja govori
	THEN	radi se o osobi

a) Navesti redosled pozivanja pravila, uspehe i neuspehe sa povratnim ulančavanjem. Usvojiti rezoluciju konflikata prema redosledu pravila. Pretpostaviti pamćenje (engl. *caching*) dokazanih činjenica.

b) Navesti redosled pozvanih pravila pri direktnom ulančavanju, ignorišući pravilo R7. Kao i u tački a), pravila uzimati prema datom redosledu.

c) Definisati drugačiju rezoluciju konflikata koja bi dobro došla za ovakav produkcioni sistem.

Rešenje

a) U slučaju povratnog ulančavanja zaključivanje se vrši od zaključaka ka činjenicama. U ovom slučaju zaključci odgovaraju akcijama koje robot može preduzeti i potrebno ih je razmotriti onim redosledom kojim su zadate.

1. Prva akcija koja se razmatra je A1, 'okrenuti se za 180 stepeni'. Jedino pravilo koje ima A1 u zaključku je R1. Pošto se pretpostavka ovog pravila 'čuje se glasan šum' ne nalazi među činjenicama niti se iz pravila može zaključiti, ne uspeva pravilo R1, a time ni akcija A1.
2. Razmatra se A2 akcija, 'stati i čekati'. Ovu akciju u THEN delu imaju pravila R3 i R5. Prema zadatoj strategiji rezolucije konflikata, za razmatranje se bira prvo od navedenih pravila, a to je R3.
3. Razmatra se preduslov pravila R3 'treba se sakriti' i pravilo R4. Preduslov 'objekat se kreće prema robotu' (vezano za drugi objekat; postojanje više objekata može se formalizovati kroz uvođenje promenljivog argumenta predikata tako da bi u ovom trenutku došlo do vezivanja promenljive za konkretnu vrednost) je zadovoljen činjenicom F5. Drugi preduslovi su 'radi se o osobi' ili 'radi se o vozilu'. Potrebno je zadovoljiti jedan od njih da bi pravilo R4 bilo zadovoljeno. Prvo od pravila koje utvrđuju da se radi o osobi je R16.
4. Razmatra se preduslov pravila R16, 'radi se o životinji'. Ovaj preduslov ne nalazi se među činjenicama, pa razmatramo prvo od pravila, pravilo R8.
5. Preduslov pravila R8 'objekat ima dugačke krake' nije zadovoljen (nema ga ni među činjenicama ni u zaključcima pravila) pa pravilo R8 nije zadovoljeno.

6. Da bismo ustanovili da li se radi o životinji biramo sledeće pravilo R9 za razmatranje. Pošto je govor vrsta nepravilnog šuma, onda je preduslov pravila R9 zadovoljen pretpostavkom F6. Prema tome novu činjenicu 'radi se o životinji' (vezano za drugi objekat) dodajemo ostalim činjenicama pošto primenjujemo pamćenje zaključaka.
7. Drugi od preduslova pravila R16, 'životinja ima četiri kraka u po dva para' nije zadovoljen, pa pravilo R16 otpada.
8. Nastavljamo razmatranje preduslova pravila R4, 'radi se o osobi' (sada vezano, izborom pravila R17. Već je ranije utvrđena činjenica 'radi se o životinji'. Činjenica F6 zadovoljava i drugi preduslov, 'životinja govori'. (Naravno ovde je uzeto da ljudi spadaju u širu klasu životinja, što je pogodno za ovaj produkcioni sistem za klasifikaciju, koji ima pravila da najpre utvrdi pripadnost široj klasi, pa zatim jednoj od njenih potklasa.).
9. Pošto je pravilo R17 zadovoljeno, nova činjenica 'radi se o osobi' dodaje se postojećim činjenicama. Pravilo R4 je takođe zadovoljeno pa se činjenicama dodaje zaključak 'treba se sakriti'.
10. Nastavljamo sa razmatranjem pravila R3. Drugi preduslov 'robot je u žbunju' nije zadovoljen (niti ga pravilima možemo zadovoljiti) pa pravilo R3 ne uspeva.
11. Alternativni način da se zadovolji postavljeni cilj, akcija 'stati i čekati' je pravilo R5. Prvi preduslov 'objekat se kreće ka robotu' je zadovoljen činjenicom F5 (koja se odnosi na drugi objekat), a drugi 'radi se o životinji' (takođe vezano za drugi objekat) je ranije zadovoljen i ubačen kao nova činjenica u bazu znanja. Prema tome, utvrđuje se da je pravilo R5 zadovoljeno, a time i postavljeni cilj, akcija 'stati i čekati'. Zato nema potrebe razmatrati druge akcije.

b) U ovom načinu zaključivanja sledimo činjenice i korišćenjem pravila iznalazimo nove činjenice koje proističu iz zadatih:

1. Činjenica F1 upariva je sa preduslovima pravila R13, R14 i R16 pa se odgovarajući predikati uklanjaju iz preduslova ovih pravila (ukoliko pretpostavimo da se u pravilima upotrebljavaju predikati sa promenljivim argumentom, zbog postojanja više objekata, pravila R13, R14 i R16 ostaće u bazi i u neizmenjenoj formi, a ispred će biti dodata odgovarajuća pojednostavljena pravila).
2. Činjenica F2 upariva je sa preduslovom iz pravila R13, 'prepreka je manja od 90 cm'. U pravilu R13 ostaje još jedan preduslov 'radi se o prepreci'.
3. Činjenica F3 uparuje drugi preduslov iz pravila R12, tako da u tom pravilu ostaje još preduslov 'objekat se ne kreće).
4. Činjenica F4 uparuje poslednji preduslov pravila R12. Iz zaključka pravila nastaje nova činjenica 'prvi objekat je prepreka'. Nazovimo je F7 i dodajmo je na početak liste činjenica za razmatranje.
5. Činjenica F7 uparuje se sa preduslovima pravila R6, R13, R14 i R15. Jedino je pravilo R13 potpuno zadovoljeno, pa novonastalu činjenicu 'prvi objekat je žbun' koju ćemo označiti sa F8 dodajemo u listu činjenica za dalje razmatranje.
6. Činjenicu F8 uparuje stav u pravilu R2, međutim ovo pravilo nije potpuno zadovoljeno.
7. Činjenicu F5 (sledeću od zadatih činjenica) uparuju stavovi iz pravila R4 i R5, međutim ne slede novi zaključci jer preduslovi pravila nisu potpuno zadovoljeni.

8. Činjenicu F6 uparuju stavovi iz pravila R9 i R17 pri čemu je R9 potpuno zadovoljeno. Zaključak 'drugi objekat je životinja' označavamo sa F9 i dodajemo u listu činjenica.
9. Činjenica F9 uparuje stavove iz pravila R16 i R17 i R17 je potpuno zadovoljeno. Prema tome, drugi objekat je osoba. Ovu činjenicu označavamo sa F10 i dodajemo u listu.
10. Činjenica F10 uparuje preduslov pravila R4, koje je potpuno zadovoljeno. Prema tome, robot treba da se sakrije, što je nova činjenica u oznaci F11.
11. Činjenica F11 kompletno zadovoljava pravilo R2. Prema tome dolazi se do zaključka da robot treba da se okrene ka žbunu (akcija A3) i pomeri unapred za malo rastojanje(akcija A4), čime je proces zaključivanja okončan.

c) U tačkama a) i b) primenom dve različite strategije zaključivanja došli smo do dva različita zaključka za isti problem. Razlog za to leži u činjenici da je uslov za završetak zaključivanja bi da se pronađe jedna akcija koju robot treba da primeni. U datoj situaciji moguće je više alternativnih akcija i da smo zaključivanje nastavili dok ne pronađemo sve akcije koje se mogu primeniti, obe strategije bi na kraju dale iste skupove mogućih akcija. U sistemima koji rade u realnom vremenu, poželjno je međutim da se zaključivanje okonča nalaženjem jedne akcije iz razloga performansi sistema za zaključivanje. Šta se može uraditi da bi prva akcija koju pronađemo bila po nekom kriterijumu najprimerenija datoj situaciji?

Pri zaključivanju povratnim ulančavanjem u tački a) iz skupa svih pravila čiji zaključak zadovoljava trenutni cilj birali smo po fiksnom redosledu pravilo za dalje razmatranje. Slično tome, pri zaključivanju direktnim ulančavanjem u tački b) fiksiran je bio redosled razmatranja činjenica (sa strategijom fokusiranja na nove činjenice) kao i redosled razmatranja pravila čiji preduslovi uparuju činjenicu koju trenutno sledimo.

Strategija rezolucije konflikata definiše način izbora jednog iz skupa pravila koja su u datom trenutku pogodna za dalje razmatranje. Postoji više različitih strategija za primenu u zaključivanju koje se mogu primeniti i pri direktnom i pri povratnom ulančavanju:

- *Uređivanje pravila po veličini preduslova* - najpre primeniti pravilo sa najvećim skupom uslovnih elemenata, to jest sa najstrožijim preduslovom
- *Uređivanje pravila po prioritetima* - svakom pravilu dodelimo određeni prioritet po nekom kriterijumu i prvo primenjujemo pravilo sa najvišim prioritetom. Kriterijum, na primer, može biti veličina preduslova pravila čime efektivno dobijamo prethodni kriterijum, ili nešto drugo. Prioriteti pravila ne moraju biti fiksni; na primer, pravila možemo urediti po ažurnosti tako da prioritet dajemo pravilu koje je najskorije (ili u najranije) korišćeno.
- *Uređivanje podataka po prioritetima* - Činjenicama možemo dodeliti prioritete po nekom kriterijumu koji će onda uticati na redosled razmatranja činjenica. Ovaj kriterijum je očigledan za direktno ulančavanje (fokusiranje pažnje na nove činjenice je jedan specijalan slučaj), a pri povratnom ulančavanju se može efektivno sprovesti tako što utiče na redosled razmatranja pojedinih stavova u složenom (pod)cilju.
- *Ograničavanje konteksta* - Deljenje pravila u grupe tako da je u određenom trenutku zaključivanja aktivna samo jedna od grupa, a druge se aktiviraju u zavisnosti od dobijenih zaključaka.

Moguće je definisanje i kombinovanih kriterijuma - na primer, ograničavanje konteksta, s tim što se u svakoj od grupa pravila primenjuje uređivanje pravila po veličini preduslova. S obzirom da se mogu zamisliti veoma složene strategije rešavanja konflikata i opšte strategije

zaključivanja, moguće je koristiti poseban produkcioni sistem - takozvana *metapravila* - za realizaciju ovih strategija.

U datom problemu moguća je primena svake od ovih strategija. Na primer, pravila možemo podeliti u više grupa od kojih bi u prvoj bila pravila za klasifikaciju okolnih objekata (da li se, na primer, radi o biljkama ,životinjama ili ljudima) a posle klasifikacije bi se aktivirala grupa pravila koja odgovaraju utvrđenoj klasi objekta.

Analizom zaključivanja iz tačaka a) i b) može se utvrditi da je direktno ulančavanje dalo rezultat primereniji situaciji jer je u obzir uzelo oba okružujuća objekata. Kod povratnog ulančavanja bi trebalo dati drugačiji prioritet ciljevima (dati prednost kretanju u odnosu na akciju 'stati i čekati') da bi se dobio isti rezultat.

Zadatak 71: Latisa odlučivanja i I-ILI-NE latisa

Dati produkcioni sistem predstaviti u obliku:

a) I-ILI-NE (engl. *AND-OR-NOT*) latise

b) latise odlučivanja

R1: if a and d and not e then r

R2: if not a and not c and q then s

R3: if not a and p then t

R4: if a and d and e then u

R5: if a and q then u

R6: if not a and not b and c then v

R7: if b and c then p

R8: if not c and d then p

R9: if not d then q

Rešenje

Latisa je skraćeni naziv za orijentisani aciklički graf. Latise odlučivanja i AND-OR-NOT latise spadaju u prevedene (kompilovane) načine predstavljanja produkcionih sistema. Prednost ovakvih struktura je u većoj brzini donošenja zaključka, dok je najveća mana to što se gubi na generalnosti produkcionog sistema; naime, ova vrsta prevođenja ne može se primeniti na produkcione sisteme koji imaju predikate sa promenljivim argumentima.

a) Da bismo za dati produkcioni sistem odredili AND-OR-NOT latisu, interpretiraćemo pravila kao logičke funkcije. Predikati koji se pojavljuju u pretpostavkama pravila predstavljaju nezavisno promenljive ovih funkcija, a predikati iz zaključaka zavisno promenljive. Upotrebljavajući notaciju za logičke funkcije, zadati produkcioni sistem opisan je sledećim skupom funkcija:

$r = a \wedge d \wedge \neg e$

$s = \neg a \wedge \neg c \wedge q$

$t = \neg a \wedge p$

$u = (a \wedge d \wedge e) \vee (a \wedge q)$

$v = \neg a \wedge \neg b \wedge c$

$p = (b \wedge c) \vee (\neg c \wedge d)$

$q = \neg d$

Treba primetiti da se predikat u pojavljuje kao zaključak u dva pravila, R4 i R5. Isto to važi i za predikat p. Jedinstvena funkcija za svaki od ovih predikata dobijena je objedinjavanjem funkcija za svako od pravila korišćenjem logičke operacije ILI, s obzirom na to da se istinitost predikata može utvrditi ili jednim ili drugim pravilom.

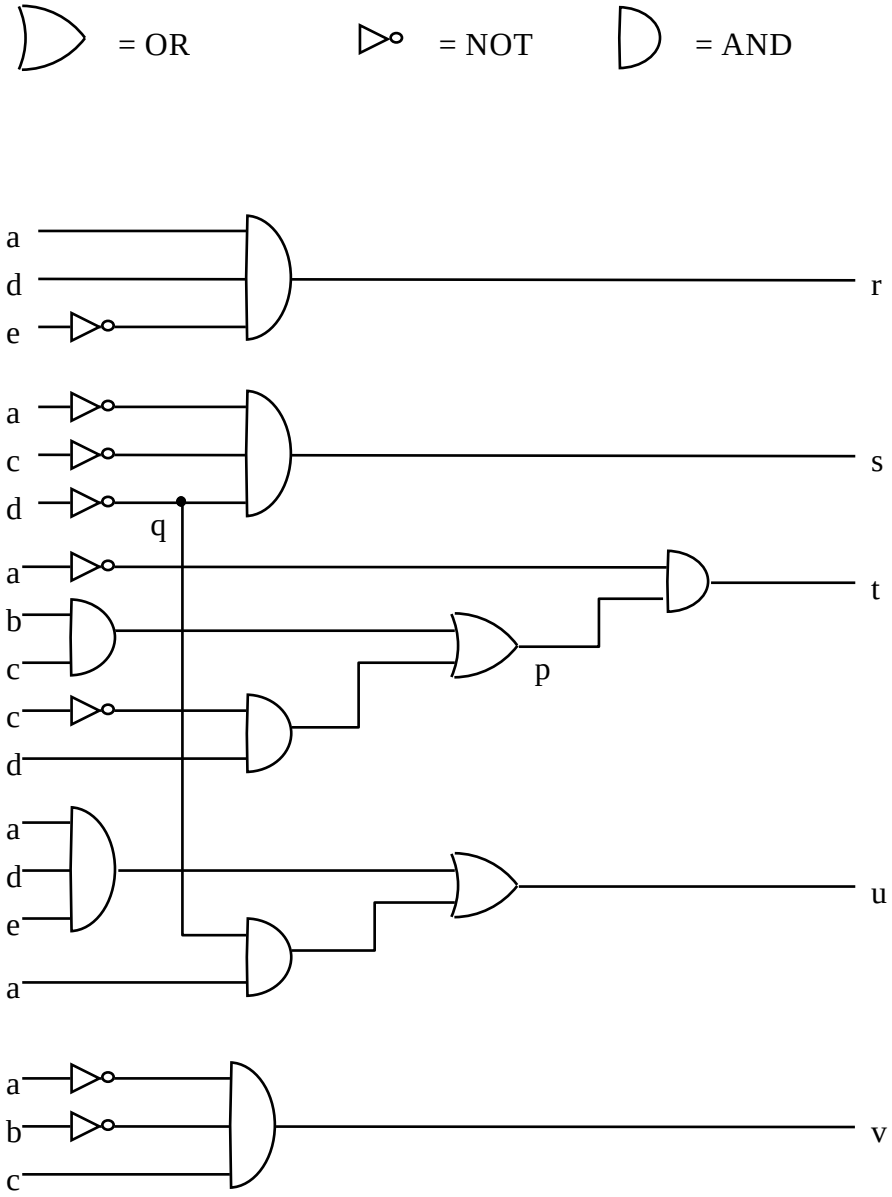
AND-OR-NOT latisa za dati produkcioni sistem predstavlja se u vidu kombinacione mreže (prikazane na slici 102) koja realizuje napred navedeni skup logičkih funkcija. Ulazi mreže su predikati-pretpostavke (na način kako su oni definisani u zadatku 62), a izlazi mreže označeni su ciljnim predikatima. Upotrebljene su standardne oznake za logičke elemente koji realizuju pojedine logičke funkcije. Međupredikati p i q su označeni na unutrašnjim linijama mreže, na onim mestima gde je realizovana njihova funkcija. Bez dodatnog elementa koji se odnosi na logičku funkciju not, AND-OR-NOT (i uz različitu grafičko predstavljanje čvorova) latise se poklapaju sa AND-OR acikličkim grafovima iz poglavlja o pretraživanju. Svaki konektor AND-OR grafa ovde se predstavlja logičkim I elementom, a logičkim ILI elementom predstavljeni su čvorovi AND-OR grafa u kojima se stiče više konektora. Postoji i razlika u orijentaciji grafičke predstave: kod AND-OR grafa cilj je na vrhu slike, a listovi su dole, dok se kombinaciona mreža po konvenciji predstavlja tako da levo budu ulazi (dakle pretpostavke), a desno izlazi (ciljevi).

Predstava produkcionog sistema putem AND-OR-NOT latise omogućava maksimalnu paralelizaciju postupka zaključivanja; mogu se zamisliti i realizacije ovako predstavljenog produkcionog sistema u integrisanim kolima za primenu u sistemi za odlučivanje u realnom vremenu.

b) Latisa odlučivanja je vrsta usmerenog acikličkog grafa koji najviše podseća na dijagram toka programa bez petlji. Pri zaključivanju, vrši se kretanje kroz graf. Svaki unutrašnji čvor latise odlučivanja ima po dve izlazne grane. U svakom čvoru ispituje se istinitosna vrednost nekog od predikata-pretpostavki (postavljanjem upita korisniku, na primer) i na osnovu toga donosi odluka kojom od izlaznih grana će se dalje nastaviti kretanje. Listovi grafa (čvorovi bez naslednika), sadrže ciljne predikate. Kada se u procesu zaključivanja dođe do nekog lista, znači da je zadovoljen cilj koji odgovara tom listu i zaključivanje se okončava.

Latisa odlučivanja može se konstruisati za dati produkcioni sistem primenom sledećeg algoritma:

1. Za svako pravilo koje u zaključku ima ciljni predikat (takozvano dijagnostičko pravilo), zameniti u preduslovu toga pravila sve pojave međupredikata preduslovima pravila koja u zaključcima imaju te međupredikate. Ako za neki međupredikat postoji više pravila koja ga imaju u zaključku, za svako od takvih pravila napraviti po jednu verziju dijagnostičkog pravila. Ovaj postupak se zove *sažimanje pravila* (engl. *rule collapsing*) i sam za sebe predstavlja jedan od postupaka kompilacije produkcionog sistema.



Slika 102

2. Izabrati predikat P koji najbolje zadovoljava sledeće uslove:

- predikat P ili njegova negacija not P pojavljuju se u preduslovima što većeg broja pravila

negacija predikata pojavljuje se u pravilima približno isti broj puta koliko i sam predikat

3. Podeliti pravila u dve grupe. U prvu grupu idu sva pravila u kojima se pojavljuje predikat P, a u drugu sva pravila u kojima se pojavljuje not P. Pravila u kojima se ne pojavljuje ni P ni not P moraju se iskopirati u obe grupe. Posle ove podele, iz svih pravila u obe grupe ukloniti iz preduslova P i not P.

4. Polazni produkcioni sistem pridružen je korenu latise odlučivanja. Ovom čvoru pridružuje se i pitanje (koje se postavlja korisniku prilikom zaključivanja uz korišćenje latise) o istinitosnoj vrednosti predikata P. Čvoru nasledniku korenog čvora za istinito P pridružena je prva grupa pravila iz tačke 3., a nasledniku za neistinito P druga grupa pravila.

5. Za svaku od dobijenih grupa ponaosob primeniti korake 2. do 4., zatim isto uraditi sa novodobijenim grupama itd. Postupak se okončava kada se iz pravila potpuno eliminišu preduslovi i ostanu samo zaključci. Ovi zaključci odgovaraju listovima (čvorovima bez naslednika) latise odlučivanja. Ako se u toku postupka deljenja dobije grupa pravila G identična sa grupom G' u nekom od već postojećih čvorova latise odlučivanja, za grupu G se ne pravi poseban čvor već se uzima čvor grupe G' (na osnovu ovoga pravila konačna struktura predstavlja aciklički graf, a ne stablo).

U datom produkcionom sistemu predikati a, b, c, d i e su pretpostavke (prema tome, u toku zaključivanja biće postavljana pitanja o njihovoj istinitosnoj vrednosti), predikati p i q su međupredikati, a predikati r, s, t, u i v predstavljaju ciljne predikate.

Produkcioni sistem koji se dobija sažimanjem datog sistema eliminacijom predikata p i q (1. koraka algoritma) odgovara korenu n_1 latise odlučivanja (slika 103).

Produkcioni sistem n_1:

if a and d and not e then r

if not a and not c and not d then s

if not a and b and c then t

if not a and not c and d then t

if a and d and e then u

if a and not d then u

if not a and not b and c then v

Dobijena su dva nova pravila koja odgovaraju originalnom pravilu R3. U prvom od njih zamenjen je predikat p preduslovom originalnog pravila R7, a u drugom predikat p zamenjen je preduslovom originalnog pravila R8. Pošto je pretpostavljeno da međupredikati p i q nisu od značaja korisniku, uklonjena su posle sažimanja pravila R7, R8 i R9.

Za prvu deobu pravila (korak 2. algoritma) izabran je predikat a koji se posle deobe uklanja iz svih pravila (korak 3. algoritma). Dobijaju se sledeće grupe koje odgovaraju istoimenim čvorovima latise sa slike 103:

Grupa n_2 odgovara pravilima u kojima se pojavljivalo a:

if d and not e then r

if d and e then u

if not d then u

Grupa n_3 odgovara pravilima u kojima se pojavljivalo not a:

if not c and not d then s

if b and c then t

if not c and d then t

if not b and c then v

Grupu n_2 najzgodnije je podeliti na osnovu predikata d čime se (posle uklanjanja d i not d) dobijaju grupe n_4 i u (koja sadrži samo zaključak u kao činjenicu pa je po njemu i nazvana), a grupu n_3 na osnovu predikata c čime se dobijaju n_5 i n_6:

Grupa n_4 (pravila sa d):

if not e then r

if e then u

Grupa u (pravila sa not d):

u

Grupa n_5 (pravila sa c):

if b then t

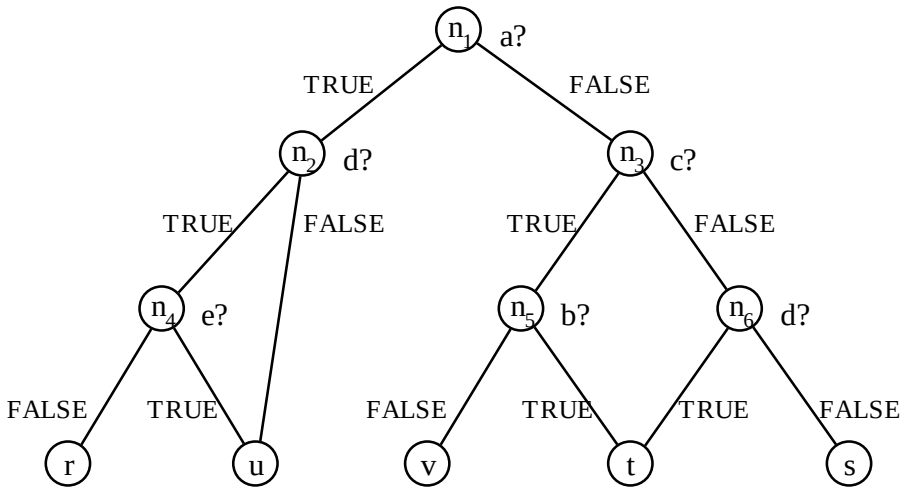
if not b then v

Grupa n_6 (pravila sa not c):

if not d then s

if d then t

U grupi u nema više pravila sa preduslovima, ostao je samo zaključak u tako da tu grupu više nije potrebno deliti. Grupe n_4, n_5 i n_6 dele se na osnovu predikata e, b i d respektivno pri čemu se dobijaju konačne grupe koje sadrže samo zaključke - listovi grafa sa slike 103.



Slika 103

Zadatak 72: Problem vraćanja u zaključivanju

Posmatrajmo upit u bazu znanja:

n(x) and n(y) and n(z) and x>y and not f(x,x) and g(x,y,z)?

Neka je sadržaj baze znanja sledeći:

n(1)

n(2)

n(3)

n(4)

n(5)

f(2,2)

f(2,4)

f(3,1)

g(1,3,2)

g(2,3,3)

g(3,3,4)

g(4,3,5)

a) Koliko se puta pri ovom upitu obavi vraćanje na prethodni predikat da bi se dobio prvi odgovor povratnim ulančavanjem? Računati svako pomeranje od desna na levo od predikata do predikata u upitu kao jedno vraćanje.

b) Preurediti upit tada da se smanji broj vraćanja. Koliki je taj broj sada?

c) Predložiti način zavisno zasnovanog vraćanja (engl. *dependency based backtracking*) da bi se dobio odgovor na početni upit. Koliki je sada broj vraćanja?

d) Dati odgovor na početni upit uz relaksaciju. Usvojiti standardan oblik relaksacije kojom se pokušava zadovoljiti svako složeno ograničenje (sa više promenljivih) zasebno, uz zanemarivanje prethodnih dodela. Prikazati tok postupka.

Rešenje

Algoritam zaključivanja povratnim ulančavanjem iz dodatka 1 primenjuje takozvano *hronološko vraćanje pri zaključivanju*, čija je karakteristika da se vraćanje sa tekućeg predikata vrši na predikat koji je poslednji bio zadovoljen pre tekućeg.

Zavisno zasnovano vraćanje pri zaključivanju formuliše se na sledeći način: Sa predikata P vraćanje se vrši na poslednji zadovoljeni predikat kojim je vezan neki od promenljivih argumenata predikata P. Ako takvih predikata nema, vraćanje se vrši na predikat zadovoljen neposredno pre razmatranja predikata P.

Relaksacija predstavlja jedan od metoda u okviru strategije rešavanja problema zadovoljavanjem ograničenja (engl. *constraint satisfaction*). Ova strategija ide na to da se smanji skup mogućih vrednosti za svaku od promenljivih u upitu razmatranjem postavljenih ograničenja.

a) Potrebno je ukupno 162 vraćanja da bi se dobio odgovor x = 4, y = 3, z = 5:

- 3 vraćanja od n(y) do n(x), pošto se moraju isprobati četiri različite vrednosti za x dok se ne dobije odgovor
- 17 vraćanja od n(z) do n(y). Mora se tri puta proći kroz pet mogućih vrednosti za y, i šesti put upit uspeva za treću po redu vrednost y (znači dodatna dva vraćanja šesti put).
- 89 vraćanja od x > y do n(z). Sedamnaest puta se razmatra x > y za svaku od 5 mogućih vrednosti za z. Osamnaesti put upit uspeva za petu po redu vrednost z.
- 29 vraćanja od not f(x,x) do x > y. Stav x > y 'prolazi' 6 parova (x,y) a to su: (2,1), (3,1), (3,2), (4,1), (4,2) i (4.3) za svaku od 5 mogućih vrednosti za z što je ukupno 30 pokušaja od kojih poslednji uspeva. Od tih 30 pokušaja, 5 puta (za x=2 i z=1 do z=5) ne uspeva not

f(x,x). 24 puta ne uspeva poslednji stav g(x,y,z). Kada god poslednji stav ne uspe, vrši se vraćanje do stava not f(x,x) što automatski povlači vraćanje do x > y jer ne postoji drugi način da not f(x,x) zadovolji kada jednom uspe.

- 24 vraćanja od g(x,y,z) do not f(x,x) kao što je u prethodnoj tački zaključeno.

b) Upit treba preurediti tako da najpre dođu oni stavovi koje je najteže zadovoljiti. U ovom slučaju radi se prvo u predikatu g, a zatim o stavu x > y. Prema tome, zadati upit se može preformulisati u:

g(x,y,z) and x > y and not f(x,x) and n(x) and n(y) and n(z)

Ovaj upit u zadatu bazu znanja daje isti odgovor kao pod a) sa samo 3 vraćanja; sva tri puta vraćanje se vrši od stava x > y do stava g(x,y,z).

c) Primenjujući zavisno zasnovano vraćanje pri razmatranju datog upita, zaključujemo da će se sa predikata n(y) vraćanje vršiti na predikat n(x), sa n(z) na n(y) sa x > y na n(y) (a ne n(z) kao kod hronološkog vraćanja jer se u stavu x > y ne pojavljuje promenljiva z), sa not f(x,x) na n(x) i sa g(x,y,z) na n(z). Razmotrimo sada tok zaključivanja uz zavisno zasnovano vraćanje za zadati upit: n(x) and n(y) and n(z) and x>y and not f(x,x) and g(x,y,z).

- Prva tri predikata vezuju promenljive: x = 1, y = 1 i z = 1.
- Stav x > y nije zadovoljen; vraćamo se na n(y).
- Bira se y = 2 i ponovo razmatra x > y koji ponovo nije zadovoljen; vraćamo se na n(y). Isto se ponavlja za vrednosti y od 3 do 5 (još tri vraćanja).
- Stav n(y) nije zadovoljen jer nema neisprobanih vrednosti za y; vraćamo se na n(x) i biramo x = 2.
- Razmatraju se n(y) i n(z), bira se y=1, z=1 i x > y sada uspeva.
- Pošto f(2,2) uspeva, not f(x,x) nije zadovoljeno i vraćamo se na n(x) uzimajući sada x = 3. Razmatranjem n(y) i n(z) uzima se y =1 i z=1.
- Stav x>y uspeva kao i not f(x,x) ali stav g(x,y,z) nije zadovoljen. Vraćamo se na n(z) i uzimamo z=2.
- Stavovi x>y, not f(x,x) uspevaju ali g(x,y,z) ponovo nije zadovoljen pa se vraćamo na n(z). Situacija se ponavlja za vrednosti z od 3 do 5 (još tri vraćanja).
- Pošto n(z) nije uspelo, vraćamo se na n(y) (neposredno prethodni stav, jer nema prethodnika koji vezuju z) i biramo y=2. Razmatranjem stava n(z) usvaja se z=1.
- Ponavlja se neispunjenje g(x,y,z) i vraćanje na n(z) za vrednosti z od 1 do 5 (još 5 vraćanja).
- Stav n(z) nije zadovoljen, vraćamo se na n(y) i biramo y=3. Razmatranjem stava n(z) usvaja se z=1.
- Stav x > y nije zadovoljen. Vraćamo se na n(y) i biramo y=4. Ponovo nije zadovoljen x > y, vraćamo se na n(y), biramo y=5. Ponovo nije zadovoljen x > y, vraćamo se na stav n(y) koji nije zadovoljen pa se dalje vraćamo na n(x) i biramo x=4, a razmatranjem stavova n(y) i n(z) usvaja se y=1 i z=1.
- Za vrednosti y=1 i y=2 ponavlja se neispunjenje g(x,y,z) i vraćanje na n(z) za vrednosti z od 1 do 5 (još 10 vraćanja sa g(x,y,z) na n(z) i 2 vraćanja sa n(z) na n(y)).

- Za y=3 ponavlja se vraćanje sa g(x,y,z) na n(z) za vrednosti z od 1 do 4 (još 4 vraćanja)
- Za z=5 uspeva kompletan upit.

Ukupno je bilo 39 vraćanja što je znatno manje od verzije sa hronološkim vraćanjem, ali i znatno više nego u slučaju optimalno uređenog upita.

d) Algoritam relaksacije naveden je u dodatku 1 (algoritam 16). Inicijalno, promenljive x, y i z su aktivne i imaju sledeće moguće vrednosti:

$$x, y, z \in \{ 1, 2, 3, 4, 5\}$$

- Razmatramo prvo promenljivu x i stavove upita koji sadrže x. Za x=1 ne postoji vrednost y tako da bude zadovoljen stav x>y, znači 1 se briše iz skupa mogućih vrednosti x. Stav not f(x,x) nije zadovoljen za x=2 pa se i ova vrednost uklanja iz skupa. Vrednost 5 se takođe uklanja jer se g(5,y,z) ne može zadovoljiti. Ostaju samo vrednost x=3 ili x=4. Promenljivu x proglašavamo neaktivnom. Napomena: ni vrednost x=3 nije moguća jer tada iz g(3,y,z) sledi y=3 a to ne zadovoljava stav x > y. Međutim, ovo razmatranje uključuje povezivanje vrednosti promenljivih iz dva različita stava, tako da se algoritmom proste relaksacije to ne može utvrditi, jer ovaj algoritam razmatra svaki stav nezavisno od drugih stavova.
- Razmatramo promenljivu y. Stav g(x,y,z) zadovoljen je samo za y=3. Promenljivu y proglašavamo neaktivnom, a promenljivu x ponovo aktiviramo jer se pominje u ograničenjima za y a nije joj utvrđena konačna vrednost.
- Ponovo razmatramo promenljivu x. Pošto je y=3, mora biti x=4 da bi važilo x > y. Promenljivu x proglašavamo neaktivnom.
- Razmatramo promenljivu z. Mora biti z=5 da bi važilo g(4,3,z). Promenljivu z proglašavamo neaktivnom.

Prema tome, u skupu mogućih vrednosti za svaku od promenljivih ostala je samo po jedna vrednost čime je problem rešen: x=4, y=3, z=5.

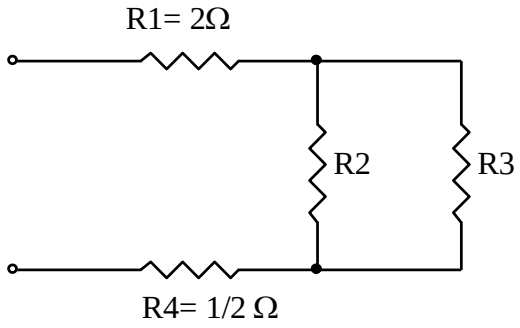
Zadatak 73: Problem električnog kola

Posmatrajmo električno kolo prikazano na slici 104. Činjenicu da su otpornici R1 i R4 serijski vezani možemo predstaviti predikatom SERIJSKI(R1,R4); činjenicu da je struja kroz otpornik R1 2A zadajemo predikatom STRUJA(R1,2) a da R1 ima 2Ω predikatom OTPORNOST(R1,2), i tako dalje.

a) Napisati pravilo koje izražava činjenicu da, ako struja I teče kroz otpor R, tada ista struja teče i kroz otpor serijski vezan sa R.

b) Napisati pravilo koje izražava činjenicu da je pad napona na otporniku jednak proizvodu struje kroz otpornik i njegove otpornosti.

c) Ispratiti put do postizanja sledećeg cilja: određivanje pada napona na otporniku R4, ako se primenjuje povratno ulančavanje. Pretpostaviti da je prethodno utvrđena činjenica da je struja kroz R1 jednaka 2A.



Slika 104

Rešenje

a) Pravilo glasi:

if (SERIJSKI(r1,r2) or SERIJSKI(r2, r1)) and STRUJA(r1,x) then STRUJA(r2,x)

b) Pravilo glasi:

if OTPORNOST(r,x) and STRUJA(r,y) and JEDNAKO(z,PROIZVOD(x,y))

then NAPON(r,z)

Funkcija PROIZVOD(x,y) vraća proizvod brojeva x i y (u trenutku izračunavanja, x i y moraju biti vezani za konkretne vrednosti). Za uvedeni predikat JEDNAKO potrebno je u bazu znanja dodati činjenicu JEDNAKO(x,x).

c) Deo baze znanja ima sledeći izgled (navedene su samo one činjenice koje su opisane predikatima zadatim u postavci):

1. if SERIJSKI(r1,r2) and STRUJA(r1,x) then STRUJA(r2,x)
2. if OTPORNOST(r,x) and STRUJA(r,y) and JEDNAKO(z,PROIZVOD(x,y))

   then NAPON(r,z)
3. OTPORNOST(R1, 2)
4. OTPORNOST(R4, 0.5)
5. SERIJSKI(R1, R4)
6. STRUJA(R1, 2)
7. JEDNAKO(x,x)

- Cilj je predstavljen sa NAPON(R4,u). Cilj ne može upariti činjenicom, već samo zaključkom pravila 2, pri čemu je r=R4, z=u. Potrebno je razmotriti preduslove ovoga pravila.
- Preduslov OTPORNOST(R4,x) zadovoljen je činjenicom pa se ima da je x = 0.5. Prelazi se na razmatranje drugog preduslova pravila 2.
- Predikat STRUJA(R4,y) ne može se upariti činjenicom već samo zaključkom pravila 1 uz vezivanje promenljivih r2 = R4, x=y. Potrebno je razmotriti preduslove pravila 1.
- Predikat SERIJSKI(r1,R4) zadovoljen je činjenicom uz vezivanje r1=R1.

- Drugi preduslov pravila 1, STRUJA(R1,x) zadovoljen je činjenicom pa je x=2. Prema tome zadovoljeno je pravilo 1 pa važi i njegov zaključak STRUJA(R4,2). Ovo zadovoljava drugi predikat iz pravila 2 za y=2.
- Sada se razmatra treći preduslov pravila 2 a to je predikat JEDNAKO(z,PROIZVOD(0.5,2)), odnosno JEDNAKO(z,1). Ovaj predikat zadovoljen je činjenicom pri čemu je z = x = 1. Pošto je potpuno zadovoljen preduslov pravila 2, važi i njegov zaključak NAPON(R4,1).

Nužno je da se novi predikat postavi krajnje desno u preduslovu. U trenutku razmatranja ovog predikata pri zaključivanju, promenljive x i y moraju biti vezane za konkretne vrednosti, inače se ne bi mogao izračunati njihov proizvod.

Redosled predikata u preduslovima pravila 1 i 2 nije proizvoljan. Razmotrimo, na primer, pravilo 1. Ovo pravilo tipično se pri zaključivanju povratnim ulančavanjem aktivira upitom STRUJA(R_i,x) (znači tražimo vrednost struje za poznati otpor R_i). Ukoliko bi se u preduslovu pravila 1 obrnuo redosled predikata, prvo bi se išlo na zadovoljavanje predikata STRUJA(r1,x) pri čemu su obe promenljive ovoga predikata slobodne. Ukoliko pretpostavimo da u bazi nema činjenica koje mogu da upare ovaj predikat, u razmatranje bi se ponovo uzelo pravilo 1 i njegov preduslov. Time se opet kao tekući cilj pojavljuje predikat STRUJA(r1,x), to jest zaključivanje upada u mrtvu petlju.

Zadatak 74: Članovi planinarskog društva

Razmotrimo sledeću situaciju: Toša, Mika i Jova članovi su planinarskog društva. Svaki član planinarskog društva koji nije skijaš je planinar. Planinari ne vole kišu, a svako ko ne voli sneg ne voli ni skijanje. Mika ne voli ništa što Toša voli i voli sve što Toša ne voli. Toša voli kišu i sneg.

a) Predstaviti ovu situaciju produkcionim sistemom pogodnim za zaključivanje ulančavanjem unazad.

b) Kakav je odgovor na pitanje: *Da li postoji neki član planinarskog kluba koji je planinar a nije skijaš?*

Rešenje

a) Koristićemo sledeće predikate:

- Član(x) važi ako je osoba x član planinarskog društva
- Skijaš(x) označava da osoba x skija
- Planinar(x) označava da je osoba x planinar
- Voli(x,y) označava da osoba x voli y, gde y može biti Kiša ili Sneg.

U postavci su navedene sledeće činjenice:

Član(Toša)

Član(Mika)

Član(Jova)

Voli(Toša,Kiša)

Voli(Toša,Sneg)

Postavkom su definisana sledeća pravila:

- Svaki član društva koji nije skijaš je planinar (podsetimo se da su promenljive u pravilima univerzalno kvantifikovane):

  P1. if Član(x) and not Skijaš(x) then Planinar(x)

- Planinari ne vole kišu. U ovom slučaju u zaključku pravila nalaziće se negacija predikata. Negacija se, prema tome, utvrđuje eksplicitno ne oslanjajući se na pretpostavku o zatvorenom svetu.

  P2. if Planinar(y) then not Voli(y,Kiša)

- Svako ko ne voli sneg, ne voli ni skijanje. U prevodu ćemo formulaciju 'voleti skijanje' prevesti predikatom Skijaš jer to odgovara smislu iskaza.

  P3. if not Voli(z,Sneg) then not Skijaš(z)

- Mika ne voli ništa što Toša voli.

  P4. if Voli(Toša,v) then not Voli(Mika,v)

- Mika voli sve što Toša ne voli.

  P5. if not Voli(Toša,w) then Voli(Mika,w)

b) Upit glasi (promenljiva t je egzistencijalno kvantifikovana):

Član(t) and Planinar(t) and not Skijaš(t).

Polazimo od datog cilja pokušavajući da ga zadovoljimo činjenicama:

1. Predikat Član(t) se prvi razmatra i zadovoljava prvom činjenicom pri čemu je t = Toša.

2. Razmatra se predikat Planinar(Toša). Nijedna činjenica ga ne zadovoljava pa se bira pravilo P1 koje u zaključku ima ovaj predikat pri čemu je x= Toša.

3. Prvi predikat iz preduslova pravila P1 je Član(Toša) i zadovoljen je istoimenom činjenicom.

4. Drugi stav preduslova pravila P1 koji glasi not Skijaš(Toša) ne nalazi se među činjenicama pa se Razmatra pravilo P3 za z=Toša.

5. Preduslov pravila P3, koji glasi not Voli(Toša,Sneg) nije ispunjen jer se među činjenicama nalazi Voli(Toša,Sneg). Prema tome, pravilo P3 nije zadovoljeno, kao ni pravilo P1 pa ne važi Planinar(Toša). Moramo se dakle vratiti na prvi stav upita da bismo razmotrili alternativni način zadovoljavanja cilja.

6. Ciljni predikat Član(t) zadovoljava se za t = Mika postojanjem istoimene činjenice.

7. Razmatra se sledeći ciljni predikat Planinar(Mika) i preduslov pravila P1.

8. Važi da je Član(Mika) pa ostaje da se razmotri predikat not Skijaš(Mika) i pravilo P3.

9. Razmatra not Voli (Mika, Sneg). Odgovarajuće činjenice nema, pa se razmatra pravilo P4 za v= Sneg.

10. Razmatra se preduslov pravila P4, predikat Voli(Toša,Sneg). Baza znanja poseduje odgovarajuću činjenicu, pa zaključujemo da je ovaj predikat zadovoljen a time i pravila P4, P3 i P1 respektivno, kao i ciljni predikat Planinar(Mika).

11. Razmatra se poslednji ciljni predikat not Skijaš(Mika). Ukoliko se pri zaključivanju primenjuje pamćenje zaključaka, odmah bi se pronašao odgovarajući predikat među činjenicama jer je to bio zaključak zadovoljenog pravila P3. Ukoliko nema pamćenja zaključaka, ponovilo bi se razmatranje pravila P3 i ponovo zaključilo da je ono zadovoljeno. Prema tome, polazni upit zadovoljen je za t = Mika.

Zadatak 75: Efikasnost zaključivanja sa pamćenjem zaključaka

Pri zaključivanju povratnim ulančavanjem pamćenje zaključaka ne mora u svakoj situaciji doneti poboljšanje performansi. Ovaj zadatak ima za cilj da utvrdi pod kojim uslovima se isplati primenjivati pamćenje zaključaka.

Pretpostavimo da su data pravila koja utvrđuju istinitosnu vrednost predikata a(x) za neku vrednost promenljive x i da je za razmatranje tih pravila potrebno u proseku R vremenskih jedinica. Pretpostavimo da u cilju ubrzavanja zaključivanja pamtimo K vrednosti promenljive x za koju je predikat a(x) tačan. Pri razmatraju upita sa predikatom a, prvo se moraju sekvencijalno ispitati sve zapamćene činjenice sa predikatom a i u slučaju da se nijedna činjenica ne može upariti sa upitom, koristimo pravila za a.

a) Neka je P verovatnoća da bilo koja zapamćena činjenica uparuje upit pri čemu su verovatnoće za sve zapamćene činjenice međusobno nezavisne. Pod kojim uslovima se pamćenje isplati? Pretpostaviti da je K <0.1/P i da svaki pristup kešu traje jednu vremensku jedinicu.

b) Ponoviti analizu iz tačke a), ukoliko verovatnoća uparivanja upita nije ista za svaku činjenicu nego ima sledeću raspodelu: za najčešće korišćenu činjenicu iznosi P, za drugu po učestalosti korišćenja iznosi P/2, za treću P/3 itd. Ostali uslovi su isti kao u tački a).

Rešenje

a) Ukoliko pri razmatranju upita prva činjenica u kešu zadovolji upit vreme obrade upita je1, ukoliko upit zadovolji druga činjenica vreme obrade je 2, za treću činjenicu je 3 itd. Pošto je rečeno da su verovatnoće međusobno nezavisne za svaku činjenicu, to znači da verovatnoća da nijedna činjenica iz keša ne zadovoljava upit iznosi P_r = 1-KP, i tada je vreme obrade upita *K+R* (vreme potrošeno na ispitivanje zapamćenih činjenica + vreme razmatranja pravila).

Prema tome prosečno vreme obrade upita izraženo je formulom:

$$T = P*1 + P*2 + P*3 +...+P*K + (1-KP)*(K+R)$$

Da bi se pamćenje isplatilo, ovo vreme mora biti manje od vremena obrade upita bez pamćenja koje se svodi na vreme razmatranja pravila R

$$T < R$$

Pošto je poznato da je 1+2+...+K = K(K+1)/2 nejednačinu možemo transformisati u sledeći oblik:

$$PK(K+1)/2+(1-KP)K < KPR, \text{ to jest}$$

$$(K+1)/2 + (1/P) -K < R$$

$$(2+P-PK)/2P<R$$

Pošto je dato da je KP<0.1 važi takođe i da je P <0.1 pa možemo aproksimirati nejednakost sa

$$PR>1$$

što je konačna formula za uslov pod kojim se isplati pamćenje zaključaka.

b) Izračunajmo prvo verovatnoću P_r da nijedna od K činjenica u kešu ne zadovoljava upit:

$$P_r = 1 - P - P/2 - P/3 - .. - P/K$$

Upotrebljavajući aproksimativnu formulu $1 + 1/2 + 1/3 + ... + 1/K \approx \log_2(K+1)$ imamo da je

$$P_r = 1 - P\log_2(K+1)$$

Ovu vrednost uvrstićemo u formulu za srednje vreme pretrage u slučaju pamćenja zaključaka:

$$T = P*1+P/2*2+P/3*3+...+P/K*K + (1 - P\log_2(K+1))*(R+K)$$

odnosno

$$T = KP + (1 - P\log_2(K+1))*(R+K)$$

Kriterijum za uspešnost pamćenja je, kao i u slučaju a):

$$T < R$$

odnosno, kada se zameni formula za T,

$$KP + (1 - P\log_2(K+1))*(R+K) < R$$

Transformacijom dobijamo

$$K(P+1) < P(R+K)\log_2(K+1)$$

Pošto je $P < PK < 0.1$ možemo zanemariti P u odnosu na 1 pa dobijamo konačan izraz:

$$1 < P(R/K+1)\log_2(K+1)$$

2.3. Semantičke mreže

Zadatak 76: Štrumfovi

Celokupno Štrumfetino znanje o štrumfovima Ljutku, Srećku, Kefalu i Luftiki predstavljeno je semantičkom mrežom na slici 105.

a) Predstaviti sledeća fakta

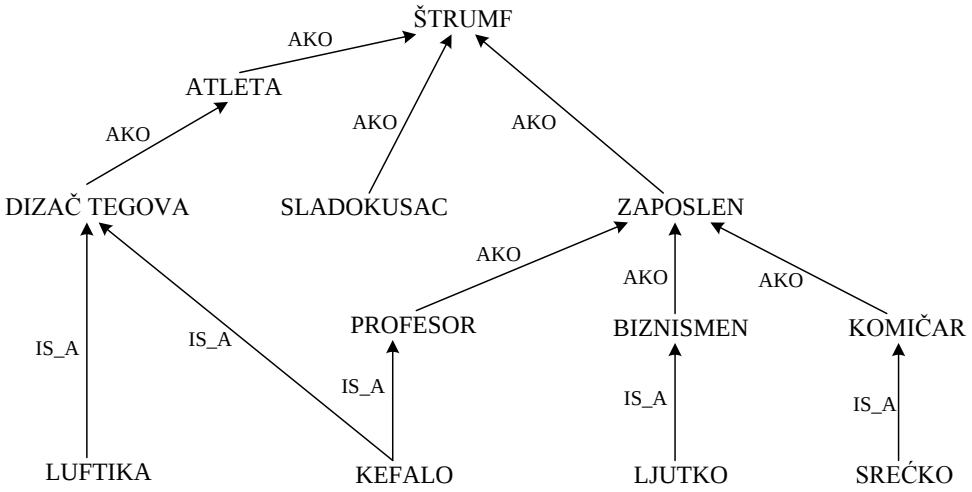
1. Štrumfovi su sitni.
2. Sladokusci su krupni.
3. Biznismen ima neprijatan karakter.
4. Štrumf ima prijatan karakter jedino ako je (taj štrumf) sitan.

dodavanjem IS-A, AKO, drugih veza i if-needed procedura u pomenutu mrežu.

b) Navesti redosled pretraživanja čvorova u odgovarajućoj proceduri nasleđivanja pri odgovoru na pitanje: Da li Luftika ima prijatan karakter?

Rešenje

U zadatoj mreži pojavljuju se veze tipa JE (engl. IS_A) i JE_VRSTA_DO (engl. A_KIND_OF skr. AKO). Veza IS_A povezuje objekat (kao konkretnu instancu neke klase objekata) sa tom klasom. Objekat može biti istovremeno specijalizacija više različitih klasa. Na primer, Luftika je i dizač tegova i sladokusac. Veza tipa AKO naznačuje da je neka klasa specifična instanca neke generalnije klase objekata.



Slika 105

Nasleđivanje u semantičkim mrežama predstavlja prenošenje zajedničkih osobina sa generalnijih klasa na specifične klase ili objekte. U konkretnom slučaju, s obzirom za štrumfa Luftiku nije eksplicitno naveden karakter, zaključak o tome mora se izvesti na osnovu saznanja kom tipu štrumfova Luftika pripada i na osnovu zajedničkih osobina svih štrumfova.

Za odgovor na pitanje mora se koristiti varijanta procedure nasleđivanja koja uzima u obzir postojanje podrazumevanih vrednosti i if-needed procedura (algoritam 15 u dodatku 1).

Procedura vrši pretragu po širini u delu mreže koji se sastoji od čvora F i njegovih sledbenika prema IS_A i AKO vezama. Data je Z varijanta, u kojoj se u istom čvoru ispituje postojanje prave, podrazumevane ili vrednosti po potrebi pre prelaska na sledeći čvor.

U N varijanti procedure nasleđivanja, podmreža se ispituje tri puta: prvi put se zanemaruje postojanje if-needed i podrazumevanih vrednosti. Ako se u ovom prolazu ne nađe rezultat, vrši se ponovno pretraživanje cele podmreže pri čemu se u ispitivanje uključuju i if-needed procedure. Ako ni ovo ispitivanje ne da rezultat, u trećoj pretrazi se uzimaju u obzir i podrazumevane vrednosti kada se konačno ili dobija rezultat ili pretraga proglašava neuspešnom.

a) Kompletna mreža sa dodatnim znanjem prikazana je na slici 106. VELIČINA i KARAKTER predstavljaju osobine štrumfova pa se radi predstavljanja ovih osobina odgovarajućim čvorovima dodaju pregratci (engl. *slot*) koje mogu uzeti jednu iz skupa vrednosti. Na primer, vrednosti u pregratka za veličinu mogu biti KRUPAN ili SITAN, a za karakter PRIJATAN ili NEPRIJATAN. Pregratci se u mreži predstavljaju na taj način što se od čvora kome dodajemo pregradak nacrta usmerena strelica ka novom čvoru koji označava vrednost pregratka. Strelica se označi imenom pregratka. Činjenice 1. i 4. definišu pregratke čvora ŠTRUMF, činjenica 2. definiše pregradak čvora SLADOKUSAC, a činjenica 3. pregradak čvora biznismen. Pregratci čvora ŠTRUMF imaju posebne osobine:

- VELIČINA je pregradak sa *podrazumevanom* (engl. *default*) vrednošću SITAN. Drugim rečima, za nekog konkretnog štrumfa, u nedostatku dodatnih informacija možemo pretpostaviti da je sitan. Na taj način iskaz "Štrumfovi su sitni" tumačimo kao "Štrumfovi su uglavnom sitni". Podrazumevane vrednosti povezane su sa nasleđivanjem osobina - videti objašnjenje pod b).

- KARAKTER je pregradak čija se vrednost izračunava *po potrebi* (engl. *if-needed*) na osnovu vrednosti drugih pregradaka, u ovom slučaju na osnovu veličine štrumfa. Za izračunavanje služi procedura K definisana pravilom 4.:

PROCEDURA K(VELIČINA): if VELIČINA = SITAN then return PRIJATAN

else return NEPRIJATAN

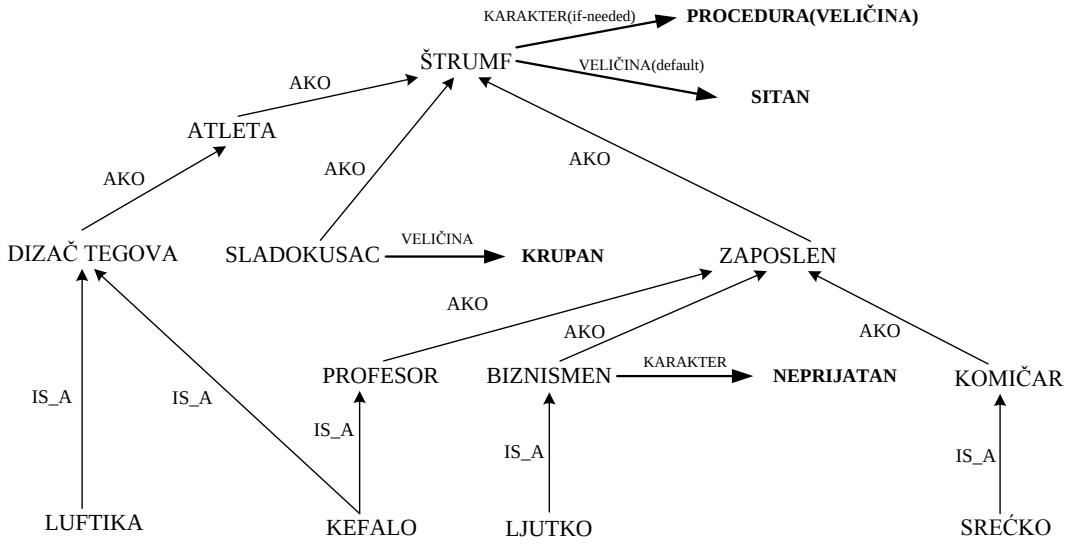
b) Procedura nasleđivanja posećuje čvorove zadate mreže sledećim redom:

LUFTIKA → DIZAČ TEGOVA → SLADOKUSAC → ATLETA → ŠTRUMF

U čvoru ŠTRUMF nalazi se if-needed procedura K kojoj je ulazni argument VELIČINA. Da bi se odredila vrednost ovog atributa, rekurzivno se poziva procedura nasleđivanja pri čemu pretraga kreće od čvora LUFTIKA:

LUFTIKA → DIZAČ TEGOVA → SLADOKUSAC

Na ovom mestu određuje se da atribut VELIČINA ima vrednost KRUPAN, pa se izvršava if-needed procedura K koja daje odgovor NEPRIJATAN za vrednost atributa KARAKTER.



Slika 106

Zadatak 77: Šerlok Holms i gospodin Vilson

Koristeći semantičku mrežu predstaviti činjenice koje se mogu izvesti iz sledećeg teksta:

Holms: Pored očiglednih činjenica da je gospodin Vilson neko vreme obavljao fizičke poslove, da je bio u Kini i da se kasnije dosta bavio pisanjem, ne mogu zaključiti ništa drugo...

Vilson: Ali kako ??

Holms: Vaše ruke, dragi moj gospodine!

Vaša desna ruka je dosta veća od leve.

Njome ste sigurno radili pa su mišići bolje razvijeni.

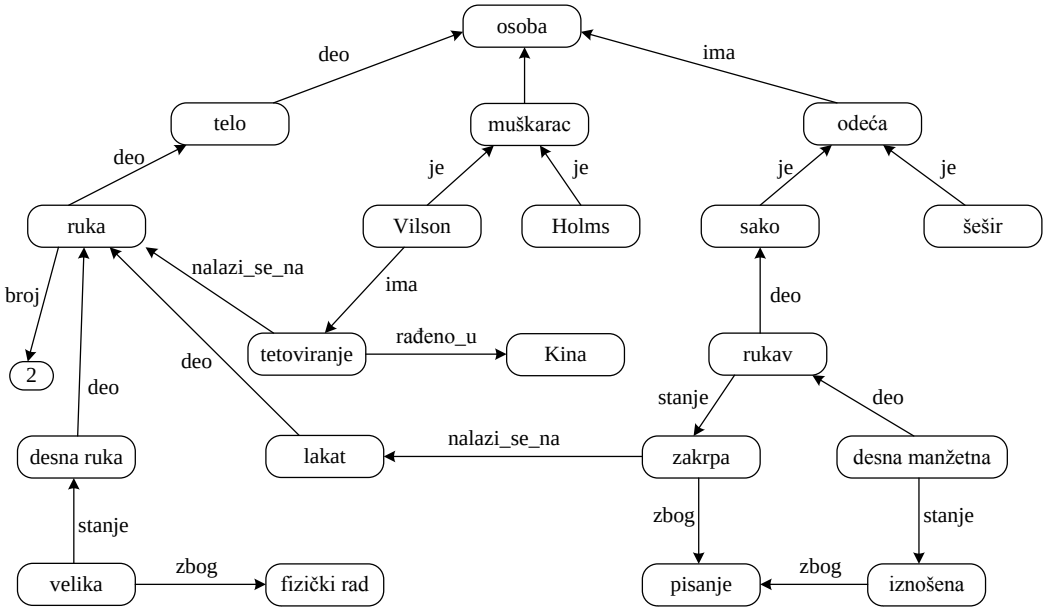
Riba koju imate istetoviranu neposredno iznad članka na ruci jedino u Kini može biti urađena.

Trik bojenja ribljih krljušti nežno ružičasto je svojstven za Kinu.

Šta još možemo zaključiti po veoma uglačanoj desnoj manžeti i po zakrpi na levom rukavu pored lakta na koji se, odmarajući se, oslanjate na sto.

Rešenje

Jedno od mogućih rešenja dato je na slici 107.



Slika 107

Zadatak 78: Alat

Nacrtati semantičku mrežu koja predstavlja sledeće činjenice. Vrednosti svojstva predstaviti takođe čvorovima. Predstaviti ono što one znače a ne ono što bukvalno kažu.

1. Ključevi su alat.
2. Čekići su alat.
3. Alat ima ručke.
4. Ručka čekića je čvrsta.
5. Ključevi su čvrsti.
6. Većina čekića ima čeličnu ručku.
7. Ključevi su u proseku dugi 25 santimetara.

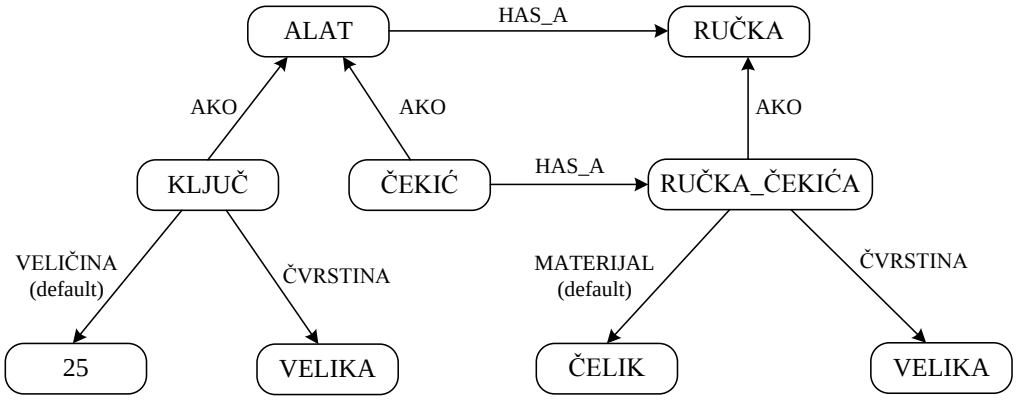
Rešenje

Rešenje je prikazano na slici 108. U predstavljanju prve dve rečenice korišćena je relacija AKO da se naznači da su klase čekića i ključeva specijalizacija šire klase alata. Konkretni primerci čekića i ključeva (na primer ključ broj 12) koji se ovde ne pominju, bili bi povezani sa odgovarajućim klasama IS_A vezama.

Treća rečenica izražava svojstvo alata predstavljeno relacijom HAS_A. Za predstavljanje četvrte rečenice u mrežu je uveden poseban čvor RUČKA ČEKIĆA koji je podklasa opštije klase RUČKA. Pogrešno bi bilo svojstvo čvrstine pripisivati čvoru RUČKA jer bi se, zbog nasleđivanja, to odnosilo i na ručku ključeva. Uočiti da bi bukvalan prevod predikata četvrte rečenice relacijom IS_A bio potpuno pogrešan jer bi to značilo da je ručka čekića specifična klasa opštije klase 'čvrsta' što je besmisleno. Čvrstina je svojstvo ručke čekića, a svojstva se prikazuju istoimenim vezama (pregratcima) u mreži između čvora koji ima određeno svojstvo

(u ovom slučaju ručka čekića) i čvora koji predstavlja vrednost tog svojstva (u ovom slučaju velika).

Od specifičnijih problema predstavljanja treba pomenuti da 6. i 7. rečenica opisuju podrazumevane (engl. *default*) vrednosti određenih svojstava što je i naznačeno na odgovarajućim vezama a povezano je sa nasleđivanjem vrednosti svojstava.



Slika 108

Zadatak 79: Zaključivanje u semantičkim mrežama

Za svaki od sledećih skupova sentenci dati najbolji način za predstavljanje znanja putem semantičke mreže da bi se mogao dati odgovor na postavljeno pitanje:

a) Sentence:

1. Jovan voli voće.
2. Banane su voće.
3. Ljudi jedu ono što vole.

Da li Jovan jede banane?

b) Sentence:

1. Većina ljudi voli bombone.
2. Većina ljudi koja priređuje zabave voli da služi hranu koju njihovi gosti vole.
3. Toma priređuje zabavu.

Šta bi Toma želeo da posluži?

Rešenje

a) Slika 109 prikazuje semantičku mrežu koja odgovara datim činjenicama. Prvi iskaz predstavljen je relacijom VOLI nad objektom JOVAN (instanca klase ČOVEK) i klasom VOĆE. Drugi iskaz označava da je objekat BANANA instanca klase VOĆE. Treći iskaz definiše osobinu klase ČOVEK pod nazivom JEDE. Vrednost ove osobine dobija se kao rezultat if-needed procedure P koja glasi:

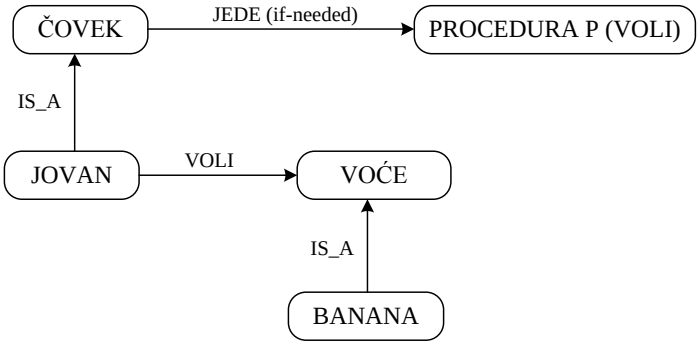
PROCEDURA P(VOLI):

AKO je vrednost pregratka VOLI jednaka x,

ONDA je vrednost pregratka JEDE takođe jednaka x.

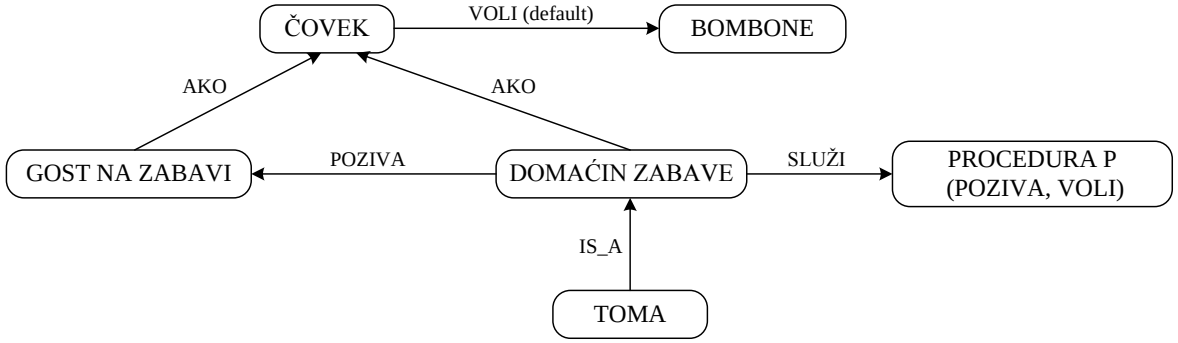
Radi odgovora na pitanje 'Da li Jovan jede banane?' potrebno je utvrditi vrednost pregratka JEDE čvora JOVAN. Vrednost ovoga pregratka dobija se nasleđivanjem: JOVAN nasleđuje vrednost pregratka JEDE od opštije klase ČOVEK. Pošto se radi o if-needed proceduri njenim razmatranjem nalazi se da JOVAN JEDE VOĆE jer se u pregratku VOLI objekta JOVAN nalazi vrednost VOĆE.

Ovime je iscrpljen skup vrednosti koje čvor JOVAN nasleđuje za pregradak JEDE. S obzirom da se u početnom upitu pominje objekat BANANA, nadalje se razmatra istoimeni čvor mreže. Ovaj čvor nasleđuje vezu JEDE (usmerenu od čvora JOVAN) od generalnije klase VOĆE, čime je konačno dobijen pozitivan odgovor na pitanje da li Jovan jede banane.



Slika 109

b) Date činjenice mogu se predstaviti mrežom na slici 110.



Slika 110

Procedura P glasi:

PROCEDURA P(POZIVA, VOLI)

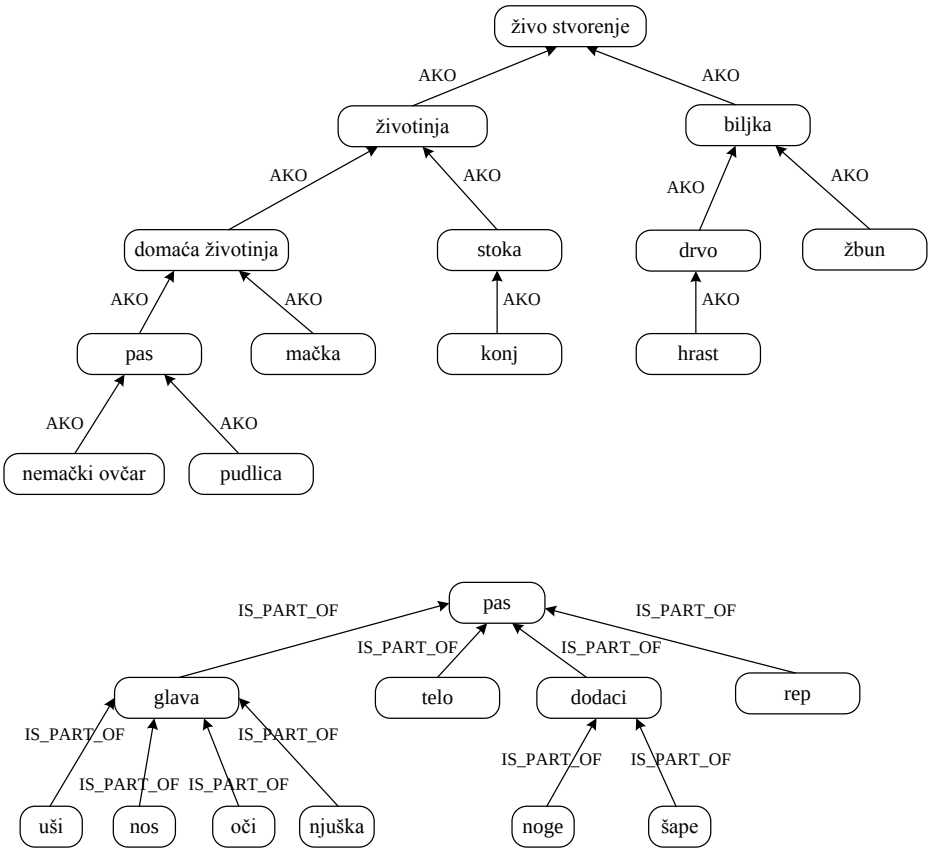
1. Odrediti čvor mreže definisan vrednošću pregratka POZIVA.
2. Odrediti vrednost pregratka VOLI čvora nađenog u koraku 1.
3. Pregradak SLUŽI dobija vrednost određenu u koraku 2.

Pitanje 'Šta bi Toma želeo da posluži?' sada formulišemo u obliku 'Koju vrednost ima pregradak SLUŽI objekta TOMA?'. Odgovor na ovo pitanje dobijamo primenjujući proceduru nasleđivanja:

- TOMA nasleđuje vrednost pregratka SLUŽI od čvora DOMAĆIN ZABAVE. Vrednost se određuje if-needed procedurom P.
- Izvršavanjem procedure P nastaje potreba za utvrđivanjem vrednosti u pregratku POZIVA čvora TOMA.
- Nasleđivanjem od opštijeg čvora DOMAĆIN ZABAVE utvrđuje se da je reč o čvoru GOST NA ZABAVI. U proceduri P potrebno je odrediti vrednost pregratka VOLI čvora GOST NA ZABAVI.
- Vrednost pregratka VOLI čvora GOST NA ZABAVI nasleđuje se od čvora ČOVEK koji predstavlja opštiju klasu. Radi se o podrazumevanoj vrednosti BOMBONE. Prema tome, procedura P vraća vrednost BOMBONE kao vrednost pregratka SLUŽI za čvor TOMA.

Zadatak 80: Problem nemačkih ovčara

Na slici 111 data je jedna semantička mreža.



Slika 111

Pretpostavimo da raspolažemo sistemom za predikatsku logiku u kojem su predstavljene informacije na slici. Koje bi dodatno znanje trebalo uključiti da obezbedimo nasleđivanje naniže u hijerarhiji?

Na primer, kako bi se moglo odgovoriti na pitanje da li nemački ovčari imaju njuške?

Rešenje

Relacija IS_PART_OF je tranzitivna: Ako je x deo od y, a y je deo od z, onda je i x deo od z. Na primer, uši su deo glave, a glava je deo psa; prema tome, uši su deo psa. Izrazimo ovo pravilo predikatskom formulom:

1. $\forall x \forall y \forall z$ [IS_PART_OF (x,y) $\wedge$ IS_PART_OF(y,z) $\Rightarrow$ IS_PART_OF(x,z)]

Nasleđivanje osobina sa opštije klase na specifičniju klasu ili konkretnu instancu obezbeđuje se sa sledeća dva pravila:

- Ako je x deo od y (IS_PART_OF) a z je podklasa šire klase y (A_KIND_OF), onda važi da je x deo i od z. Na primer, rep je deo psa, a nemački ovčar je objekat klase pas iz čega proizilazi da je rep deo i od nemačkog ovčara.

  2. $\forall x \forall y \forall z$ [IS_PART_OF (x,y) $\wedge$ AKO(z,y) $\Rightarrow$ IS_PART_OF (x,z)]

- klasa x je podklasa klase y ako su u semantičkoj mreži čvorovi x i y povezani sa jednom ili više AKO veza od čvora x ka čvoru y. Na primer, pas je vrsta domaće životinje; domaća životinja je vrsta životinje; prema tome, zaključak je da je pas vrsta životinje.

  3. $\forall x \forall y \forall z$ [AKO (x,y) $\wedge$ AKO(y,z) $\Rightarrow$ AKO(x,z)]

Deo semantičke mreže koji se odnosi na pitanje da li nemački ovčari imaju njuške opisuje konjunkcijom sledećih predikata:

AKO(nemački ovčar, pas)

IS_PART_OF(njuška, glava)

IS_PART_OF(glava, nemački ovčar)

Primenom pravila 1. za x = njuška, y = glava, z = pas, primenom modus ponensa (iz A i A $\Rightarrow$ B zaključuje se B), možemo utvrditi da važi

IS_PART_OF(njuška, pas).

Primenom pravila 2. za x = njuška, y = pas i z = nemački ovčar, primenom modus ponensa, možemo utvrditi da važi

IS_PART_OF (njuška, pas)

čime je ustanovljen pozitivan odgovor na postavljeno pitanje.

2.4. Okviri

Zadatak 81: Pismo Pobesnelog Programera

Predstaviti značenje sledećeg pisma grupom međusobno povezanih okvira.

Dragi Magic Software,

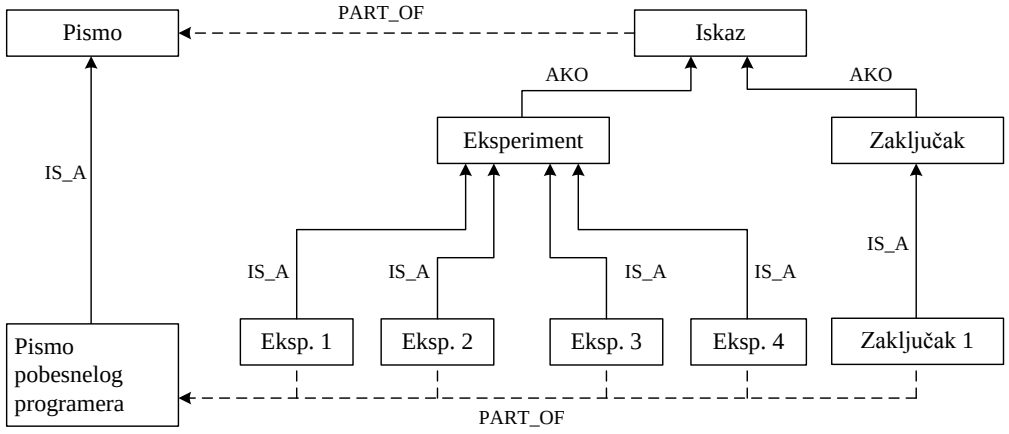
Isprobao sam Vaš "Zadivljujući inteligenti softver" i on ne radi. Probao sam Primer 3 iz Vašeg priručnika i on se slupao pri pokušaju deljenja nulom. Zatim sam probao Primer 8 i linijski štampač mi je odštampao 10000 LF. Kada sam izašao iz Vašeg programa, utvrdio sam da su sve moje datoteke uništene. Hoću da mi vratite novac.

Pobesneli Programer

Rešenje

Potrebno je identifikovati klase objekata, kao i konkretne objekte pojedinih klasa. Za svaki od objekata i za njihove klase potrebno je oformiti poseban okvir sa pregratcima koji opisuju osobine ovih objekata. Pismo se može predstaviti hijerarhijom okvira prikazanom na slici 112.

Punim linijama date su veze među okvirima koje se odnose na hijerarhiju nasleđivanja, a isprekidanim linijama ostale veze koje se uspostavljaju vrednostima pregradaka pojedinih okvira. Na primer, okvir Eksperiment 1 je jedan konkretan objekat koji pripada klasi Eksperiment pa je povezan sa odgovarajućim okvirom vezom IS_A. Klasa Eksperiment predstavlja jednu specijalizaciju šire klase Iskaz pa je veza među njima tipa AKO (engl. *A Kind Of*, sa značenjem *Vrsta Od*). Iskaz je deo (engl. PART_OF) pisma, jer pismo treba da sadrži pregradak Tekst koji predstavlja listu iskaza.



Slika 112

Sledi opis svakog od okvira; za svaki pregradak naveden je njegov naziv, vrednost iza dvotačke (crtica označava da je pregradak prazan) i kod (nekih okvira) tip vrednosti u zagradi.

Okvir: Pismo

Pošiljalac: -

Primalac:-

Tekst: - (lista iskaza)

Okvir: Pismo Pobesnelog Programera

Pošiljalac: Pobesneli Programer

Primalac: Magic Software

Tekst: Eksperiment 1, Eksperiment 2, Eksperiment 3, Eksperiment 4, Zaključak 1

Okvir: Iskaz

Tip: -

Okvir: Eksperiment

Tip: konstatacija

Akcija: -

Rezultat: -

Okvir: Zaključak

Tip: zahtev

Sadržaj: -

Okvir: Eksperiment 1

Tip: konstatacija

Akcija: ispitivanje 'Zadivljujućeg inteligentnog softvera'

Rezultat: ne radi

Okvir: Eksperiment 2

Tip: konstatacija

Akcija: proba Primera 3 iz priručnika

Rezultat: pokušaj deljenja nulom

Okvir: Eksperiment 3

Tip: konstatacija

Akcija: proba Primera 8 iz priručnika

Rezultat: odštampano 10000 LF na štampaču

Okvir: Eksperiment 4

Tip: konstatacija

Akcija: Izlazak iz programa

Rezultat: uništene korisničke datoteke

Okvir: Zaključak

Tip: zahtev

Sadržaj: vratite mi moj novac

Izloženo rešenje koncentriše se na predstavljanje strukture pisma. Rešenje je moguće dalje razraditi tako što bi se program 'Zadivljujući inteligentni softver' tretirao kao poseban objekat klase Program, koji poseduje objekat klase Priručnik koji opet poseduje objekte klase Primer. Sada bi pojedini objekti klase Eksperiment bili povezani (pregradak Akcija) sa objektima klase Primer. Za svaki od ovih objekata, kao i za klase kojima oni pripadaju, uveo bi se poseban okvir sa odgovarajućih pregratcima. Na primer, okvir za program opisan u pismu mogao bi se opisati na sledeći način:

Okvir: Program 'Zadivljujući inteligentni softver'

Naziv: Zadivljujući inteligentni softver

Proizvođač: Magic Software

Korisnik: Pobesneli Programer

Prilog: Priručnik za 'Zadivljujući inteligentni softver'

Testiran sa: Eksperiment 1, Eksperiment 2, Eksperiment 3, Eksperiment 4

U daljoj razradi postojali bi i okviri za Pobesnelog programera i firmu Magic Software.

Zadatak 82: Predavanje na fakultetu (hijerarhija okvira)

Posmatrajmo okvir koji predstavlja bilo koji čas (predavanje) na Elektrotehničkom fakultetu.

a) Dati dva primera pregrada koje će biti uvek popunjene vrednostima i odrediti koje će to vrednosti biti.

b) Izvršiti generalizaciju i specijalizaciju okvira.

c) Naznačiti jednu pregradu koja se nasleđuje od nadređenog okvira i jednu koja se ne nasleđuje.

Rešenje

a) Okvir za predavanje mora sadržati pregratke koji karakterišu čas: mesto, vreme, predavač, predmet, slušaoci, trajanje časa itd. Samo oni pregratci koji čije su vrednosti zajedničke za sve časove na Elektrotehničkom fakultetu mogu biti popunjeni. Na primer, to je pregradak naziv fakulteta koji ima vrednost Elektrotehnički fakultet i pregradak trajanje časa koji ima vrednost 45 minuta. Drugi pregratci, na primer oznaka učionice, postoje ali nisu popunjeni. Oni će biti popunjeni u okvirima koji predstavljaju specijalizaciju razmatranog okvira i koji se odnose na jedan konkretan čas na Elektrotehničkom fakultetu.

b) Generalniji od posmatranog okvira koji predstavlja bilo koje predavanje na Elektrotehničkom fakultetu je okvir koji predstavlja bilo koje predavanje na bilo kojem fakultetu, generalniji od ovog poslednjeg je okvir koji predstavlja bilo koje predavanje u bilo kojoj obrazovnoj instituciji itd.

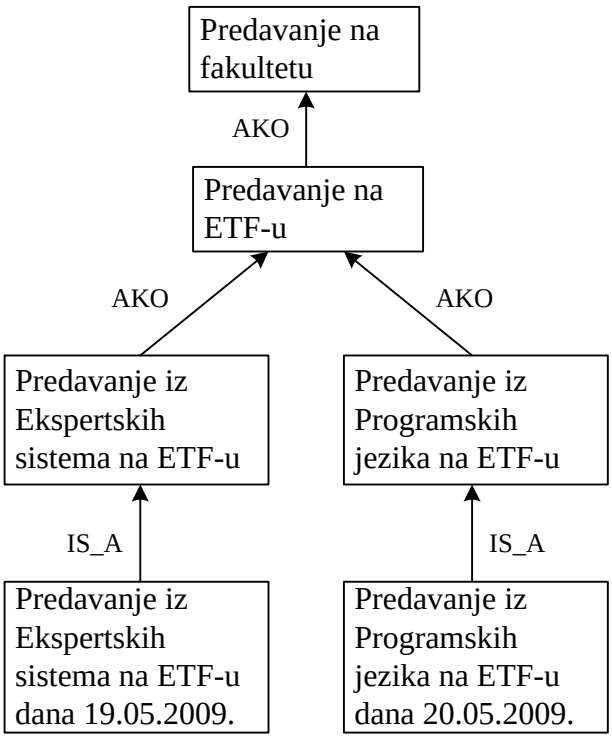
Okvir koji predstavlja bilo koje predavanje iz predmeta 'ekspertski sistemi' na Elektrotehničkom fakultetu je specijalan slučaj posmatranog okvira za bilo koje predavanje na Elektrotehničkom fakultetu. Za konkretan čas iz predmeta 'ekspertski sistemi' koji je održan 20. maja školske 1994/95 godine može se definisati okvir koji je specijalizacija prethodno definisanog okvira. Ovaj poslednji okvir odgovara (u terminologiji objektno orijentisanog projektovanja) konkretnom objektu, dok svi prethodni okviri definišu neku klasu objekata pri

čemu se definiše hijerarhija klasa međusobno povezanih vezama tipa JE_VRSTA_OD (engl. *A_KIND_OF* skraćeno *AKO*) i JE (engl. *IS_A*). , kako je prikazano na slici 113. Veza IS_A odražava pripadnost konkretnog objekta nekoj klasi, dok veza tipa AKO odražava pripadnost neke klase generalnijoj klasi.

c) Postoje dva tipa nasleđivanja kod okvira:

- nasleđivanje pregradaka i
- nasledjivanje vrednosti pregradaka

Pod nasleđivanjem pregradaka podrazumeva se činjenica da okvir mora imati sve pregratke kao i okvir koji je generalniji od njega, bez obzira da li su oni popunjeni vrednostima ili nisu. Tako, na primer, okvir koji predstavlja bilo koje predavanje na Elektrotehničkom fakultetu ima pregradak za trajanje časa kao i generalniji okvir za bilo koje predavanje na bilo kojem fakultetu.



Slika 113

Okvir koji predstavlja specijalizaciju, pored pregradaka nasleđenih od generalnijeg okvira ima u opštem slučaju i pregratke koji nisu nasleđeni nego su karakteristični za specijalizovani okvir, a ne nalaze se u generalnijem okviru. Na primer, okvir koji predstavlja čas iz 'ekspertskih sistema' može da ima pregradak 'materijal koji se deli na času'. Ovaj pregradak ne poseduje okvir za bilo koji čas na fakultetu, jer postoje drugi predmeti u kojima se na času nikada ne dele materijali.

Nasleđivanje vrednosti pregradaka podrazumeva da se vrednost nekog pregratka u posmatranom okviru, ako nije data, može utvrditi na osnovu postojeće vrednosti istoimenog pregratka nekog od generalnijih okvira. Na primer, ako bilo koji čas na Elektrotehničkom fakultetu traje 45 minuta, tada i bilo koji čas iz 'ekspertskih sistema' traje 45 minuta.

Zadatak 83: Računar (nasleđivanje pregradaka i vrednosti pregradaka)

Svaki računar ima proizvođača i identifikacioni broj. Centralni procesor je deo svakog računara. Razmotrimo okvir koji predstavlja bilo koji računar prodat od strane preduzeća Ata Mata doo.

a) Navesti primer nasleđivanja vrednosti pregratka iz ovog okvira naniže u hijerarhiji.

b) Navesti primer nasleđivanja pregratka iz ovog okvira naniže u hijerarhiji.

Rešenje

a) i b) Okvir koji predstavlja bilo koji računar proizveden u Ata Mata doo preduzeću ima sledeća dva pregratka:

- *naziv proizvođača* koji ima vrednost 'Ata Mata doo'. Ovaj pregradak i njegovu vrednost nasleđuju svi specifičniji okviri (koji se odnose na pojedine tipove računara koje proizvodi ovo preduzeće)
- *identifikacioni broj* koji je prazan (nije definisana vrednost pregratka). Ovaj pregradak nasleđuju svi specifičniji okviri u hijerarhiji. Okviri koji predstavljaju pojedinačne konkretne računare imaju vrednost u ovom pregratku (vrednost se, dakle, ne nasleđuje od generalnijih okvira).

Zadatak 84: Ustavni sud (relacije među okvirima)

Razmotrimo sledeća dva primera zaključivanja:

Aksioma 1: Državne institucije nalaze se u više gradova Srbije.

Aksioma 2: Sudovi su državne institucije.

Zaključak: Sudovi se nalaze u više gradova Srbije.

Ovaj zaključak ima smisla.

Aksioma 1: Sudovi se nalaze u više gradova Srbije.

Aksioma 2: Ustavni sud Srbije je sud.

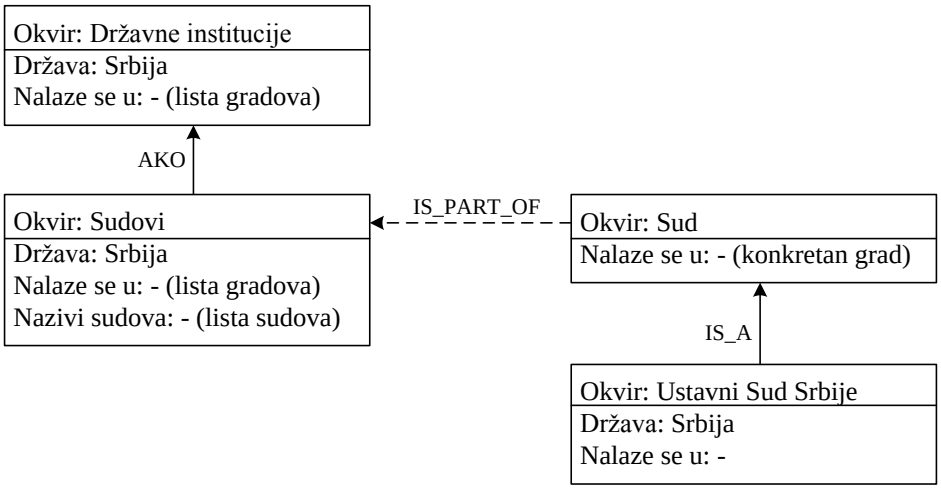
Zaključak: Ustavni sud Srbije se nalazi u više gradova Srbije.

Ovo je pogrešan zaključak na osnovu tačnih pretpostavki. Predstaviti aksiome iz prethodna dva primera jedinstvenom hijerarhijom okvira, ali na taj način da se nasleđivanjem ne može dobiti pogrešan zaključak iz drugog primera. Pri tome se ne smeju dodavati nove informacije, na primer da je Ustavni sud Srbije u Beogradu.

Rešenje

Za pravilno predstavljanje datih iskaza bitno je uočiti da se skup svih sudova posmatran kao celina nalazi se u skupu gradova, ali se svaki konkretan sud nalazi u jednom gradu. Tražena hijerarhija okvira prikazana je na slici 114 i sastoji se od tri klase (Državne institucije, Sudovi i Sud) i jednog konkretnog objekta (Ustavni sud Srbije). Odnos klase Sudovi (koja se odnosi na skup svih sudova Srbije) i klase Sud (koja generalizuje jedan sud) je odnos tipa IS_PART_OF. Ukoliko bi se (greškom) uzelo da je klasa Sud jedna specijalizacija klase

Sudovi (veza tipa AKO), okvir Ustavni sud Srbije nasledio bi pregradak 'nalaze se u:' i vrednost ovog pregratka od okvira Sudovi. U datom rešenju, veza tipa IS_PART_OF prekida lanac nasleđivanja.



Slika 114

Zadatak 85: Narudžba (nasleđivanje kvalifikujućih pregradaka)

Posmatrajmo okvir F1 koji predstavlja narudžbu (obrazac kojim se naručuje kupovina nečega).

a) Navesti primer nasleđivanja vrednosti iz ovog okvira u neki drugi okvir.

b) Posmatrajmo kvalifikujući pregradak *valuta* vezan za pregradak *cena koštanja* u okviru narudžbe. Pretpostavimo da pregradak *cena koštanja* nasleđuje svoju vrednost od istog pregratka nekog drugog okvira F2. Odakle se pregradak *valuta* nasleđuje?

Rešenje

Svaki pregradak definisan je vrednošću koju sadrži i tipom te vrednosti. Na primer, pregradak *cena koštanja* može imati vrednost 100 i tip *dinari*. Tip pregratka može biti eksplicitno naveden u definiciji pregratka ili može biti određen vrednošću nekog drugog pregratka K. Pregradak K tada se naziva *kvalifikujući pregradak* jer određuje tip vrednosti drugih pregradaka.

Na primer, za pregradak *valuta* tip može biti eksplicitno definisan kao jedna od vrednosti *dinari, marke, dolari*. Pošto je pregradak *valuta* kvalifikujući za pregradak *cena koštanja*, to znači da će vrednost ovog pregratka (na primer *dinari*) određivati tip vrednosti pregratka *cena koštanja*.

a) Pošto okvir F1 predstavlja prilično apstraktnu klasu njegova struktura odgovara praznom formularu narudžbenice. Drugim rečima, većina pregradaka ovog okvira (koji definišu, na primer, ko naručuje, šta se naručuje i od koga) je prazna pa specifičniji okviri ne mogu od F1 naslediti vrednosti. Pregratci koji su popunjeni u okviru F1 mogli bi biti: *naslov* popunjen vrednošću *narudžba* ili, eventualno pregradak *gde se dobij*a sa vrednošću: *kancelarija magacionera* koji opisuje gde se mogu uzeti prazni obrazci narudžbenice. Nezavisno od

nasleđivanja vrednosti, specifičniji okviri od okvira F1 u hijerarhiji nasleđuju sve pregratke okvira F1.

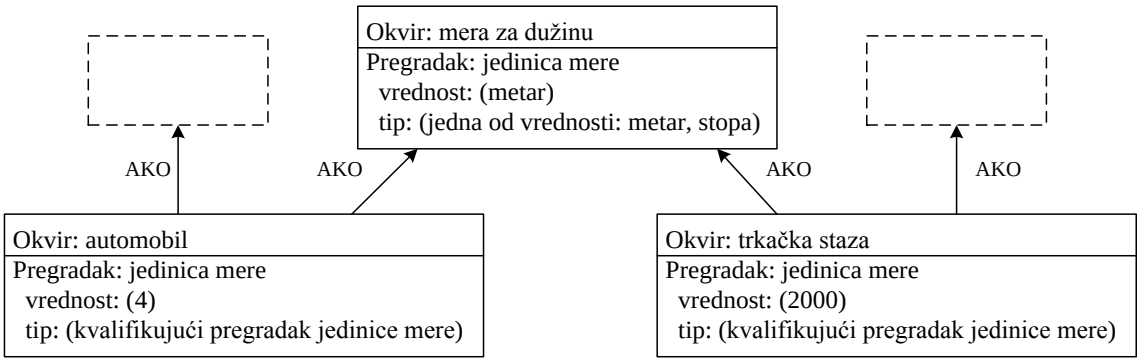
b) Pregradak *valuta* se nasleđuje na isti način kao i pregradak *cena koštanja* od generalnijih okvira u hijerarhiji. Prema tome, pregradak *valuta* i njegova vrednost biće nasleđeni od okvira F2 ili od nekog generalnijeg okvira.

Zadatak 86: Dužina (višestruko nasleđivanje)

Pretpostavimo da su pri pravljenju nekog ekspertskog sistema definisani okviri *automobil* i *trkačka staza*. I jedan i drugi okvir imaju pregradak *dužina* i želi se da tip ova dva okvira bude uvek određen istim kvalifikujućim pregratkom *jedinica mere*. Drugim rečima, ako se ekspertski sistem bude prodavao na evropskom kontinentu da obe dužine budu izražene u metrima, a ako se sistem bude prodavao u Engleskoj da dužine budu izražene u stopama, a da se promena izvrši jednostavnom promenom vrednosti kvalifikujućeg pregratka *jedinice mere*. Međutim, ova dva okvira u hijerarhiji okvira nemaju zajedničkog prethodnika (to jest, ne postoji generalniji okvir koji bi kao specijalne slučajeve imao navedena dva okvira), tako da ne mogu naslediti kvalifikujući pregradak. Predložiti rešenje ovoga problema.

Rešenje

Rešenje je prikazano na slici 115. Uveden je novi okvir *mera za dužinu* koji poseduje pregradak *jedinica mere.* U hijerarhiji okvira novi okvir predstavlja zajedničkog prethodnika okvirima *trkačka staza* i *automobil*. Prema tome, okviri *trkačka staza* i *automobil* nasleđuju pregradak *jedinica mere* (i njegovu vrednost) iz okvira *mera za dužinu.* Svaki od okvira *trkačka staza* i *automobil* može, u hijerahiji okvira, da ima i druge okvire prethodnike osim okvira *mera za dužinu.* Tada se radi o takozvanom višestrukom nasleđivanju - od drugih prethodnika ovi okviri nasleđuju sve njihove pregratke.



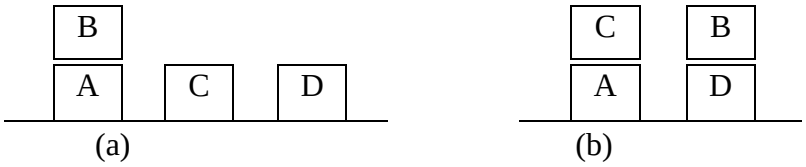
Slika 115

3. Strategije rešavanja problema

3.1. Planiranje

Zadatak 87: Svet blokova i STRIPS

Primenom STRIPS mehanizma generisati niz operatora kojima bi se niz blokova uz pomoć mehaničke hvataljke preveo iz rasporeda prikazanog na slici 116a u raspored prikazan na slici 116b.



Slika 116

Rešenje

STRIPS algoritam služi za rešavanje problema strategijom planiranja. Problem je potrebno predstaviti u formalnoj logici. Cilj STRIPS-a je da se nađe sekvenca operatora koja sistem prevodi iz početnog u ciljno stanje. STRIPS koristi sledeće strukture podataka:

- tekuće stanje problema u toku rada algoritma opisano u predikatskoj logici; inicijalno je to startno stanje, a ažurira se primenom operatora.
- ciljni stek koji sadrži stavove koji odgovaraju trenutnom (pod)cilju u toku rada algoritma i operatore izabrane za primenu za koje uslovi primene nisu još zadovoljeni.

Operatori promene stanja definišu se zadavanjem tri liste stavova:

- PREDUSLOV je lista stavova koji moraju biti ispunjeni u tekućem stanju da bi operator mogao biti primenjen
- UKLONI je lista stavova koji se uklanjaju iz tekućeg stanja u trenutku primene operatora
- DODAJ je lista stavova koji se dodaju tekućem stanju u trenutku primene operatora.

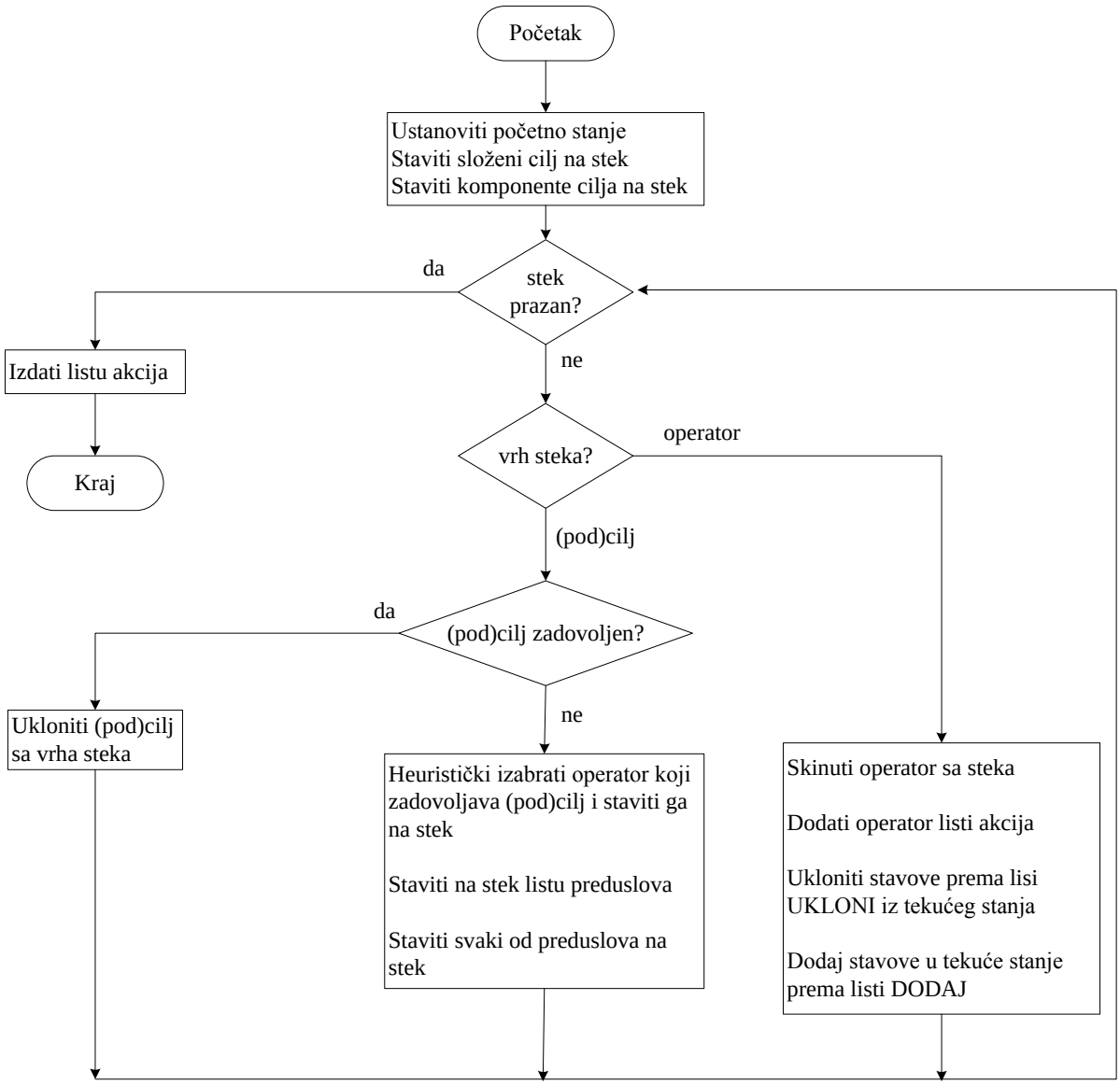
STRIPS algoritam prikazan je na slici 117.

Za opis stanja problema uvode se sledeći predikati za opisivanje međusobnog položaja blokova:

- Na(B,A) označava da se blok B nalazi neposredno iznad bloka A.
- NaStolu(C) označava da se blok C nalazi neposredno na stolu.
- NaVrhu(B) označava da se iznad bloka B ne nalazi nijedan drugi blok.

Status mehaničke hvataljke opisuje se predikatima:

- RukaPrazna označava da nema blokova u hvataljci.
- URuci(A) označava da se u hvataljci nalazi blok A.



Slika 117

Početno stanje može se predstaviti sledećim skupom literala:

Na(B,A)	NaVrhu(B)	NaStolu(A)	NaVrhu(C)
NaStolu(C)	NaVrhu(D)	NaStolu(D)	RukaPrazna

Ciljno stanje predstavlja se sa:

Na(C,A)	NaVrhu(C)	NaStolu(A)	Na(B,D)
NaVrhu(B)	NaStolu(D)	RukaPrazna	

Akcije mehaničke hvataljke (ruke) možemo modelirati sledećim operatorima:

- operator **UZMI_SA_STOLA**(x):

  PREDUSLOV: RukaPrazna $\wedge$ NaVrhu(x) $\wedge$ NaStolu(x)

  UKLONI: RukaPrazna; NaVrhu(x); NaStolu(x)

DODAJ: URuci(x)

- operator **SPUSTI_NA_STO**(y):

  PREDUSLOV: URuci(y)

  UKLONI: URuci(y)

  DODAJ: RukaPrazna; NaVrhu(y); NaStolu(y)

- operator **SKINI_SA_BLOKA**(u,z):

  PREDUSLOV: RukaPrazna $\wedge$ NaVrhu(u) $\wedge$ Na(u,z)

  UKLONI: RukaPrazna; NaVrhu(u); Na(u,z)

  DODAJ: URuci(u); NaVrhu(z)

- operator **STAVI_NA_BLOK**(v,w):

  PREDUSLOV: URuci(v) $\wedge$ NaVrhu(w)

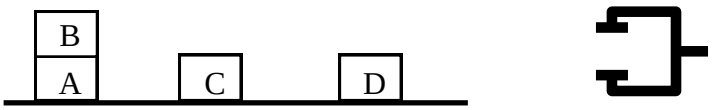
  UKLONI: URuci(v); NaVrhu(w)

  DODAJ: RukaPrazna; NaVrhu(v); Na(v,w)

STRIPS algoritam nalaže da se tekuće stanje problema inicijalizuje formalnim opisom početnog stanja i da se na ciljni stek stavi kako složeni cilj (jednak formalnom opisu ciljnog stanja) tako i svaka od komponenata složenog cilja pojedinačno. Posle ove inicijalizacije izgled struktura podataka je sledeći:

TEKUĆE STANJE:

Na(B,A)	NaVrhu(B)	NaStolu(A)	NaVrhu(C)
NaStolu(C)	NaVrhu(D)	NaStolu(D)	RukaPrazna



CILJNI STEK (raste naniže):

Na(C,A)$\wedge$NaVrhu(C)$\wedge$NaStolu(A)$\wedge$Na(B,D)$\wedge$NaVrhu(B)$\wedge$NaStolu(D)$\wedge$RukaPrazna

Na(C,A)

NaVrhu(C)

NaStolu(A)

Na(B,D)

NaVrhu(B)

NaStolu(D)

RukaPrazna

LISTA AKCIJA:

-

Grafički prikaz tekućeg stanja nije deo algoritma već je dat radi lakšeg praćenja promena tekućeg stanja. Prva tri stava sa vrha steka RukaPrazna, NaStolu(D) i NaVrhu(B) su već

zadovoljena u tekućem stanju, pa se uklanjaju sa steka. Na vrhu steka ostaje stav Na(B,D) koji trenutno ne važi. Algoritam nalaže izbor operatora koji u svom DODAJ delu ima navedeni stav, a to je jedino operator **STAVI_NA_BLOK**(v,w), pri čemu je izvršena unifikacija v ⇔ B, w ⇔ D. Zatim se na stek stavljaju sam operator, njegov preduslov i svaka komponenta preduslova posebno, posle čega je izgled steka:

CILJNI STEK (raste naniže):

Na(C,A)∧NaVrhu(C)∧NaStolu(A)∧Na(B,D)∧NaVrhu(B)∧NaStolu(D)∧RukaPrazna

Na(C,A)

NaVrhu(C)

NaStolu(A)

Na(B,D)

STAVI_NA_BLOK(B,D)

URuci(B) ∧ NaVrhu(D)

URuci(B)

NaVrhu(D)

Vršni stav je zadovoljen u tekućem stanju dok stav URuci(B) nije zadovoljen. Ponovo, dakle, treba pronaći operator koji ima ovakav stav u svom DODAJ delu. U ovom slučaju postoje dva operatora **UZMI_SA_STOLA** i **SKINI_SA_BLOKA** pa treba izvršiti izbor jednog od njih. STRIPS algoritam ne propisuje kako se vrši izbor, već se koristi poseban algoritam vezan za konkretan problem. U našem slučaju, pošto se blok B u tekućem stanju nalazi na bloku A pogodniji je operator **SKINI_SA_BLOKA**(u,z) uz unifikacije promenljivih u ⇔ B, z ⇔ A, pa se na stek stavlja operator, njegov preduslov i svaki stav preduslova:

CILJNI STEK (raste naniže):

Na(C,A)∧NaVrhu(C)∧NaStolu(A)∧Na(B,D)∧NaVrhu(B)∧NaStolu(D)∧RukaPrazna

Na(C,A)

NaVrhu(C)

NaStolu(A)

Na(B,D)

STAVI_NA_BLOK(B,D)

URuci(B) ∧ NaVrhu(D)

URuci(B)

SKINI_SA_BLOKA(B,A)

RukaPrazna ∧ NaVrhu(B) ∧ Na(B,A)

RukaPrazna

NaVrhu(B)

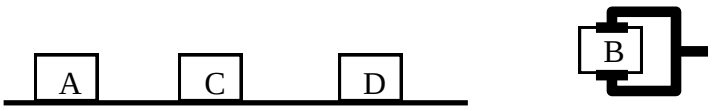
Na(B,A)

U tekućem stanju svi preduslovi važe pa ih sve skidamo sa steka, na čijem vrhu ostaje operator **SKINI_SA_BLOKA**(B,A). Prema algoritmu, operator se skida sa steka i upisuje u listu akcija; vrši se modifikacija tekućeg stanja: uklanjaju se stavovi koje operator **SKINI_SA_BLOKA** ima u svojoj UKLONI listi, a dodaju se stavovi iz njegove DODAJ liste. Strukture podataka STRIPS-a posle ovoga izgledaju ovako:

TEKUĆE STANJE

NaStolu(A) NaVrhu(A) NaVrhu(C) NaStolu(C)

NaVrhu(D) NaStolu(D) URuci(B)



CILJNI STEK (raste naniže):

Na(C,A)∧NaVrhu(C)∧NaStolu(A)∧Na(B,D)∧NaVrhu(B)∧NaStolu(D)∧RukaPrazna

Na(C,A)

NaVrhu(C)

NaStolu(A)

Na(B,D)

STAVI_NA_BLOK(B,D)

URuci(B) ∧ NaVrhu(D)

URuci(B)

LISTA AKCIJA:

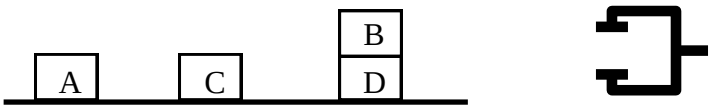
1. **SKINI_SA_BLOKA**(B,A)

Vršni stav na steku URuci(B) sada važi, kao i složeni preduslov, pa se uklanjaju sa steka na čijem se vrhu sada pojavljuje operator **STAVI_NA_BLOK**(B,D). Prema algoritmu, operator se skida sa steka i dopisuje u listu akcija a tekuće stanje se ažurira u skladu sa UKLONI i DODAJ listama operatora **STAVI_NA_BLOK**. U tom trenutku imamo sledeći izgled struktura podataka:

TEKUĆE STANJE

NaStolu(A) NaVrhu(A) NaVrhu(C) NaStolu(C)

Na(B,D) NaVrhu(B) NaStolu(D) RukaPrazna



CILJNI STEK (raste naniže):

Na(C,A)∧NaVrhu(C)∧NaStolu(A)∧Na(B,D)∧NaVrhu(B)∧NaStolu(D)∧RukaPrazna

Na(C,A)

NaVrhu(C)

NaStolu(A)

Na(B,D)

LISTA AKCIJA:

1. **SKINI_SA_BLOKA**(B,A)

2. **STAVI_NA_BLOK**(B,D)

Vršna tri stava steka zadovoljena su u tekućem stanju, pa se uklanjaju sa steka. Izgled ciljnog steka posle uklanjanja stavova je:

CILJNI STEK (raste naniže):

Na(C,A)∧NaVrhu(C)∧NaStolu(A)∧Na(B,D)∧NaVrhu(B)∧NaStolu(D)∧RukaPrazna

Na(C,A)

Potrebno je naći operator koji zadovoljava stav Na(C,A) a to je slučaj jedino sa operatorom **STAVI_NA_BLOK**(v,w), uz unifikacije v ⇔ C, w ⇔ A, posle čega na stek ide prvo operator a zatim i njegovi preduslovi pa je izgled steka:

CILJNI STEK (raste naniže):

Na(C,A)∧NaVrhu(C)∧NaStolu(A)∧Na(B,D)∧NaVrhu(B)∧NaStolu(D)∧RukaPrazna

Na(C,A)

STAVI_NA_BLOK(C,A)

URuci(C) ∧ NaVrhu(A)

URuci(C)

NaVrhu(A)

Stav NaVrhu(A) je zadovoljen, pa se skida sa steka; na vrhu steka ostaje stav URuci(C) koji nije zadovoljen. Operatori koji ga zadovoljavaju su **UZMI_SA_STOLA** i **SKINI_SA_BLOKA**. Koristeći znanje o tekućem stanju i značenju pojedinih operatora može se zaključiti da je operator **UZMI_SA_STOLA**(x) pogodniji. Ovaj operator, nakon unifikacije x ⇔ C, stavlja se na stek kao i njegovi preduslovi nakon čega stek izgleda ovako:

CILJNI STEK (raste naniže):

Na(C,A)∧NaVrhu(C)∧NaStolu(A)∧Na(B,D)∧NaVrhu(B)∧NaStolu(D)∧RukaPrazna

Na(C,A)

STAVI_NA_BLOK(C,A)

URuci(C) ∧ NaVrhu(A)

URuci(C)

NaVrhu(A)

UZMI_SA_STOLA(C)

RukaPrazna ∧ NaVrhu(C) ∧ NaStolu(C)

RukaPrazna

NaVrhu(C)

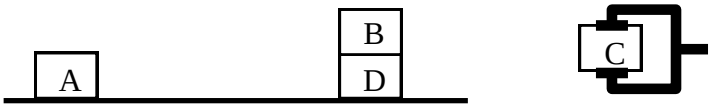
NaStolu(C)

Pošto su sve komponente složenog podcilja zadovoljene, uklanjamo ih sa steka kao i sam podcilj te na vrhu ostaje operator **UZMI_SA_STOLA**. Uklanjanjem ovog operatora sa steka, ažuriranjem liste akcija i tekućeg stanja, izgled struktura podataka je sledeći:

TEKUĆE STANJE

NaStolu(A) NaVrhu(A) Na(B,D)

NaVrhu(B) NaStolu(D) URuci(C)



CILJNI STEK (raste naniže):

Na(C,A)∧NaVrhu(C)∧NaStolu(A)∧Na(B,D)∧NaVrhu(B)∧NaStolu(D)∧RukaPrazna

Na(C,A)

STAVI_NA_BLOK(C,A)

URuci(C) ∧ NaVrhu(A)

URuci(C)

LISTA AKCIJA:

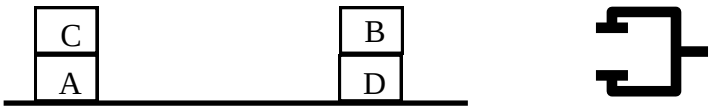
1. **SKINI_SA_BLOKA**(B,A)
2. **STAVI_NA_BLOK**(B,D)
3. **UZMI_SA_STOLA**(C)

Stav URuci(C) je u tekućem stanju zadovoljen pa se uklanja sa steka; kako su obe komponente složenog podcilja koji ostaje na vrhu steka zadovoljene, i taj podcilj se uklanja. Zatim se operator **STAVI_NA_BLOK**(C,A) skida sa steka i upisuje u plan operacija, a tekuće stanje se ažurira preko UKLONI i DODAJ lista ovog operatora. Prema tome, stanje je:

TEKUĆE STANJE

NaStolu(A) Na(C,A) NaVrhu(C) Na(B,D)

NaVrhu(B) NaStolu(D) RukaPrazna



CILJNI STEK (raste naniže):

Na(C,A)∧NaVrhu(C)∧NaStolu(A)∧Na(B,D)∧NaVrhu(B)∧NaStolu(D)∧ RukaPrazna

Na(C,A)

LISTA AKCIJA:

1. **SKINI_SA_BLOKA**(B,A)
2. **STAVI_NA_BLOK**(B,D)
3. **UZMI_SA_STOLA**(C)
4. **STAVI_NA_BLOK**(C,A)

Sada se vidi da tekuće stanje odgovara ciljnom stanju, pa se sa steka skida kako stav Na(C,A) tako i složeni cilj; stek ostaje prazan što znači da je problem rešen. Izdavanjem liste akcija koje sistem prevode iz početnog u ciljno stanje završava se rad STRIPS algoritma.

Zadatak 88: Svet blokova (problem izbora operatora)

Na slici 118a prikazana je jedna situacija iz sveta cigala. Pronaći koje operatore i kojim redom treba primeniti, da bi se prešlo u situaciju pokazanu na slici 118b, koristeći STRIPS algoritam.

Slika 118

Rešenje

Predstava stanja i operatori mogu se definisati na isti način kao u zadatku 87. Početno stanje u ovom slučaju opisano je stavovima:

NaStolu(A) Na(B,A) NaVrhu(B) RukaPrazna

dok je ciljno stanje opisano sa:

NaStolu(B) Na(A,B) NaVrhu(B) RukaPrazna

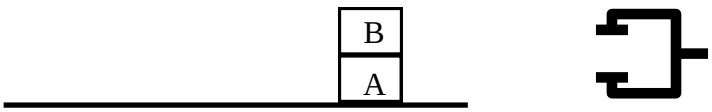
Nije lako definisati algoritam koji obezbeđuje optimalan redosled izbora operatora u opštem slučaju. Razmotrimo rad STRIPS algoritma ako za izbor operatora usvojimo jednostavno pravilo da se operatori, u situacijama kada postoji mogućnost izbora više od jednog od njih, biraju po redosledu kojim su definisani, naime:

1. **UZMI_SA_STOLA**(x)
2. **SPUSTI_NA_STO**(y)
3. **SKINI_SA_BLOKA**(u,z)
4. **STAVI_NA_BLOK**(v,w).

Inicijalno imamo sledeću situaciju:

TEKUĆE STANJE

NaStolu(A) Na(B,A) NaVrhu(B) RukaPrazna



CILJNI STEK (raste naniže):

NaStolu(B) $\wedge$ Na(A,B) $\wedge$ NaVrhu(A) $\wedge$ RukaPrazna

NaStolu(B)

Na(A,B)

NaVrhu(A)

RukaPrazna

LISTA AKCIJA:

-

Stav RukaPrazna zadovoljen je u početnom stanju pa se ovaj stav skida sa steka. Da bi se zadovoljio stav NaVrhu(A) potrebno je izabrati operator. Mogući operatori, koji imaju stav NaVrhu(A) u svom DODAJ delu, su:

1-1 **SPUSTI_NA_STO**(A)

1-2 **SKINI_SA_BLOKA**(u,A)

1-3 **STAVI_NA_BLOK**(A,w)

pa biramo prvi od njih, **SPUSTI_NA_STO**(A). Izgled steka posle ažuriranja je:

CILJNI STEK (raste naniže):

NaStolu(B) ∧ Na(A,B) ∧ NaVrhu(A) ∧ RukaPrazna

NaStolu(B)

Na(A,B)

NaVrhu(A)

SPUSTI_NA_STO(A)

URuci(A)

Da bismo zadovoljili stav URuci(A) na izboru su sledeći operatori:

2-1 **UZMI_SA_STOLA**(A)

2-2 **SKINI_SA_BLOKA**(A,z)

Biramo operator **UZMI_SA_STOLA**(A) i stavljamo ga na stek zajedno sa preduslovima:

CILJNI STEK (raste naniže):

NaStolu(B) ∧ Na(A,B) ∧ NaVrhu(A) ∧ RukaPrazna

NaStolu(B)

Na(A,B)

NaVrhu(A)

SPUSTI_NA_STO(A)

URuci(A)

UZMI_SA_STOLA(A)

RukaPrazna ∧ NaVrhu(A) ∧ NaStolu(A)

RukaPrazna

NaVrhu(A)

NaStolu(A)

Stav NaStolu(A) zadovoljen je u tekućem stanju, pa ga skidamo sa steka. Na vrhu steka ostaje nezadovoljeni stav NaVrhu(A) koji se već pojavljivao kao tekući cilj u tekućem stanju.

Algoritam u ovoj situaciji nalaže ponovni izbor operatora **SPUSTI_NA_STO**(A), a za zadovoljavanje preduslova primene ovog operatora bio bi ponovo izabran operator **UZMI_SA_STOLA**(A) i tako dalje. Ova situacija predstavlja, prema tome, 'ćorsokak' za algoritam izbora operatora.

Algoritam izbora operatora treba dakle da obezbedi vraćanje unatrag (engl. *backtracking*), tako što će obnoviti stanje steka pre izbora operatora **UZMI_SA_STOLA**(A) i izabrati alternativni operator, u ovom slučaju operator **SKINI_SA_BLOKA**(A,z).

CILJNI STEK (raste naniže):

NaStolu(B) ∧ Na(A,B) ∧ NaVrhu(A) ∧ RukaPrazna

NaStolu(B)

Na(A,B)

NaVrhu(A)

SPUSTI_NA_STO(A)

URuci(A)

SKINI_SA_BLOKA(A,z)

RukaPrazna ∧ Na(A,z) ∧ NaVrhu(A)

RukaPrazna

Na(A,z)

NaVrhu(A)

Treba primetiti da pri prethodnom izboru operatora promenljiva z nije vezana ni za jednu od konstanti A, B, C jer za tim nije postojala potreba. Na vrhu steka opet se pojavio nezadovoljeni stav NaVrhu(A) koji signalizuje pogrešan izbor operatora i nalaže vraćanje na situaciju pre izbora poslednjeg operatora:

CILJNI STEK (raste naniže):

NaStolu(B) ∧ Na(A,B) ∧ NaVrhu(A) ∧ RukaPrazna

NaStolu(B)

Na(A,B)

NaVrhu(A)

SPUSTI_NA_STO(A)

URuci(A)

S obzirom da su već razmotrene sve mogućnosti za zadovoljavanje stava URuci(A), zaključujemo da je operator **SPUSTI_NA_STO**(A) pogrešno izabran pa se vraćamo na razmatranje situacije pre izbora ovoga operatora:

CILJNI STEK (raste naniže):

NaStolu(B) ∧ Na(A,B) ∧ NaVrhu(A) ∧ RukaPrazna

NaStolu(B)

Na(A,B)

NaVrhu(A)

U ovoj situaciji biramo sledeći operator po redosledu primene, a to je **SKINI_SA_BLOKA**(u,A). Stavljamo ga na stek zajedno sa preduslovima primene:

CILJNI STEK (raste naniže):

NaStolu(B) ∧ Na(A,B) ∧ NaVrhu(A) ∧ RukaPrazna

NaStolu(B)

Na(A,B)

NaVrhu(A)

SKINI_SA_BLOKA(u,A)

RukaPrazna ∧ Na(u,A) ∧ NaVrhu(u)

RukaPrazna

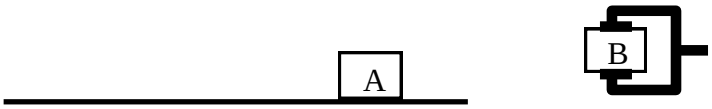
Na(u,A)

NaVrhu(u)

Izvršavanje algoritma nastavlja se unifikacijom u ⇔ B da bi se utvrdilo da se stav NaVrhu(B) nalazi u tekućem stanju. Unifikacijom je određena vrednost i za ostale pojave promenljive *u* na ciljnom steku tako da je sad na vrhu steka stav Na(B,A). Ovaj stav je zadovoljen, pa se uklanja sa steka kao i stav RukaPrazna. Zadovoljen je i složeni preduslov za primenu operatora **SKINI_SA_BLOKA**(B,A) pa se preduslov i operator uklanjaju sa steka i operator se primenjuje čime se dobija novo tekuće stanje:

TEKUĆE STANJE

NaStolu(A) NaVrhu(A) URuci(B)



CILJNI STEK (raste naniže):

NaStolu(B) ∧ Na(A,B) ∧ NaVrhu(A) ∧ RukaPrazna

NaStolu(B)

Na(A,B)

NaVrhu(A)

LISTA AKCIJA:

1. **SKINI_SA_BLOKA**(B,A)

Stav na vrhu steka, NaVrhu(A) zadovoljen u ovom stanju pa se uklanja sa steka. Stav Na(A,B) nije zadovoljen u tekućem stanju. Jedini operator koji može da zadovolji ovaj stav je **STAVI_NA_BLOK**(v,w) uz unifikaciju v ⇔ A, w ⇔ B pa se ovaj operator stavlja na stek zajedno sa preduslovima.

CILJNI STEK (raste naniže):

NaStolu(B) ∧ Na(A,B) ∧ NaVrhu(A) ∧ RukaPrazna

NaStolu(B)

Na(A,B)

STAVI_NA_BLOK(A,B)

URuci(A) ∧ NaVrhu(B)

URuci(A)

NaVrhu(B)

Stav na vrhu steka, NaVrhu(B), nije zadovoljen u tekućem stanju. Mogući operatori za izbor su:

3-1 **SPUSTI_NA_STO**(B)

3-2 **STAVI_NA_BLOK**(B,w)

3-3 **SKINI_SA_BLOKA**(u,B)

Bira se operator **SPUSTI_NA_STO**(y) i ažurira ciljni stek stavljanjem ovoga operatora i njegovih preduslova:

CILJNI STEK (raste naniže):

NaStolu(B) ∧ Na(A,B) ∧ NaVrhu(A) ∧ RukaPrazna

NaStolu(B)

Na(A,B)

STAVI_NA_BLOK(A,B)

URuci(A) ∧ NaVrhu(B)

URuci(A)

NaVrhu(B)

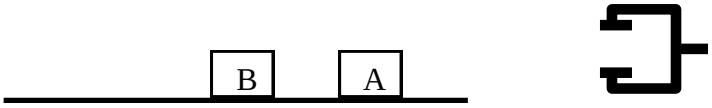
SPUSTI_NA_STO(B)

URuci(B)

Preduslov URuci(B) je zadovoljen, pa je moguće primeniti operator **SPUSTI_NA_STO**(B) čime se dobija novo tekuće stanje:

TEKUĆE STANJE

NaStolu(A) NaVrhu(A) NaStolu(B) NaVrhu(B) RukaPrazna



CILJNI STEK (raste naniže):

NaStolu(B) ∧ Na(A,B) ∧ NaVrhu(A) ∧ RukaPrazna

NaStolu(B)

Na(A,B)

STAVI_NA_BLOK(A,B)

URuci(A) ∧ NaVrhu(B)

URuci(A)

NaVrhu(B)

LISTA AKCIJA:

1. **SKINI_SA_BLOKA**(B,A)

2. **SPUSTI_NA_STO**(B)

Stav NaVrhu(B) zadovoljen je u ovom stanju, pa se uklanja sa vrha steka. Za zadovoljavanje stava URuci(A) na raspolaganju su operatori:

4-1 **UZMI_SA_STOLA**(A)

4-2 **SKINI_SA_BLOKA**(A,z)

Bira se prvi od ovih operatora i stavlja na stek zajedno sa preduslovima. Izgled steka je:

CILJNI STEK (raste naniže):

NaStolu(B) ∧ Na(A,B) ∧ NaVrhu(A) ∧ RukaPrazna

NaStolu(B)

Na(A,B)

STAVI_NA_BLOK(A,B)

URuci(A) ∧ NaVrhu(B)

URuci(A)

UZMI_SA_STOLA(A)

RukaPrazna ∧ NaVrhu(A) ∧ NaStolu(A)

RukaPrazna

NaVrhu(A)

NaStolu(A)

Svi preduslovi za primenu operatora **UZMI_SA_STOLA**(A) zadovoljeni su u tekućem stanju, pa se zajedno sa operatorom uklanjaju sa steka. Primenom ovoga operatora dobija se novo stanje:

TEKUĆE STANJE

NaStolu(B) NaVrhu(B) URuci(A)

CILJNI STEK (raste naniže):

NaStolu(B) ∧ Na(A,B) ∧ NaVrhu(A) ∧ RukaPrazna

NaStolu(B)

Na(A,B)

STAVI_NA_BLOK(A,B)

URuci(A) ∧ NaVrhu(B)

URuci(A)

LISTA AKCIJA:

1. **SKINI_SA_BLOKA**(B,A)
2. **SPUSTI_NA_STO**(B)
3. **UZMI_SA_STOLA**(A)

Preduslovi za primenu operatora **STAVI_NA_BLOK**(A,B) su u potpunosti zadovoljeni pa se redom skidaju sa steka. Ovaj operator takođe se uklanja sa steka i primenjuje na tekuće stanje čime se dobija novo tekuće stanje.

TEKUĆE STANJE

NaStolu(B) Na(A,B) NaVrhu(A) RukaPrazna

CILJNI STEK (raste naniže):

NaStolu(B) ∧ Na(A,B) ∧ NaVrhu(A) ∧ RukaPrazna

NaStolu(B)

Na(A,B)

LISTA AKCIJA:

1. **SKINI_SA_BLOKA**(B,A)
2. **SPUSTI_NA_STO**(B)
3. **UZMI_SA_STOLA**(A)
4. **STAVI_NA_BLOK**(A,B)

Svi stavovi na steku su zadovoljeni pa se redom skidaju sa steka dok se stek ne isprazni čime se okončava algoritam. Rešenje problema dato je listom akcija.

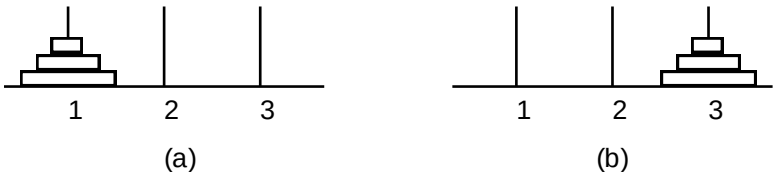
Čitaocu ostaje da proveri da li se može definisati bolji redosled operatora pri izboru koji bi smanjio broj vraćanja unatrag.

Zadatak 89: Hanojske kule

Početno stanje u igri Hanojskih kula, prikazano na slici 119a, opisano je sledećim stavovima:

NA(S1,MD) NA(S1,SD) NA(S1,VD)

Napomena: S_i označava stub *i*, MD označava mali disk, SD srednji a VD veliki disk.



Slika 119

a) Opisati ciljno stanje prikazano na slici 119b. Definisati operatore:

- POMERI_MALI_DISK
- POMERI_SREDNJI_DISK
- POMERI_VELIKI_DISK

poštujući pri tome sledeća ograničenja:

- u datom trenutku može se pomeriti samo jedan disk koji mora biti slobodan i
- veći disk ne može da dođe iznad manjeg.

b) Navesti redosled biranja operatora tokom rada STRIPS algoritma i redosled operatora u listi akcija.

Rešenje

Treba primetiti da se u modelu stanja ne vidi eksplicitan redosled diskova na istom stubu. U dozvoljenim stanjima ovaj redosled je jednoznačno određen pa to omogućava da se ova informacija ne uključi u predstavu stanja. Pri definiciji operatora promene stanja treba obezbediti da se prelazi iz legalnih stanja vrše samo u druga legalna stanja.

a) Ciljno stanje se opisuje koristeći isti predikat NA kojim je predstavljeno početno stanje:

NA(S3,MD) NA(S3,SD) NA(S3,VD)

Operatori se mogu definisati na sledeći način:

- operator **POMERI_MALI_DISK**(x,y):

PREDUSLOV: NA(x, MD)

ODUZMI: NA(x, MD)

DODAJ: NA(y, MD)

- operator **POMERI_SREDNJI_DISK**(z,u):

PREDUSLOV: $NA(z, SD) \wedge \neg NA(z, MD) \wedge \neg NA(u, MD)$

ODUZMI: NA(z, SD)

DODAJ: NA(u, SD)

- operator **POMERI_VELIKI_DISK**(v,w):

PREDUSLOV: $NA(v,VD) \wedge \neg NA(v,MD) \wedge \neg NA(v,SD) \wedge \neg NA(w,MD) \wedge \neg NA(w,SD)$

ODUZMI: NA(v, VD)

DODAJ: NA(w, VD)

Prvi argument operatora predstavlja izvorišni, a drugi argument odredišni stub. U radu STRIPS algoritma podrazumeva se pretpostavka o zatvorenom svetu (engl. *closed world assumption*) da negacija stava važi pod uslovom da se taj stav ne nalazi u tekućem stanju.

b) Inicijalno stanje opisano je na sledeći način.

TEKUĆE STANJE:

NA(S1,MD) NA(S1,SD) NA(S1,VD)

CILJNI STEK:

NA(S3,MD) ∧ NA(S3,SD) ∧ NA(S3,VD)

NA(S3,MD)

NA(S3,SD)

NA(S3,VD)

LISTA AKCIJA:

-

Da bismo zadovoljili stav NA(S3,VD) biramo operator **POMERI_VELIKI_DISK** uz unifikaciju v ⇔S1 w ⇔ S3 pa je izgled ciljnog steka:

CILJNI STEK:

NA(S3,MD) ∧ NA(S3,SD) ∧ NA(S3,VD)

NA(S3,MD)

NA(S3,SD)

NA(S3,VD)

POMERI_VELIKI_DISK(S1,S3)

NA(S1, VD) ∧ ¬NA(S1, MD) ∧ ¬NA(S1, SD) ∧ ¬NA(S3, MD) ∧ ¬NA(S3, SD)

NA(S1, VD)

¬NA(S1, MD)

¬NA(S1, SD)

¬NA(S3, MD)

¬NA(S3, SD)

S obzirom da se stav NA(S3,SD) ne nalazi u tekućem stanju, zadovoljena je negacija ovoga stava prema pretpostavci o zatvorenom svetu, pa se ovaj stav uklanja sa steka; isto važi i za stav ¬NA(S3, MD). Stav ¬NA(S1, SD) nije zadovoljen u tekućem stanju pa biramo operator **POMERI_SREDNJI_DISK**(S1, S2) jer je to jedini operator koji uklanja stav NA(S1,SD). Između mogućnosti da izaberemo S2 ili S3 za odredišni disk, izbor je pao na S2 jer vodi do rešenja u manjem broju poteza. Sada je ciljni stek

CILJNI STEK:

NA(S3,MD) ∧ NA(S3,SD) ∧ NA(S3,VD)

NA(S3,MD)

NA(S3,SD)

NA(S3,VD)

POMERI_VELIKI_DISK(S1,S3)

NA(S1, VD) ∧ ¬NA(S1, MD) ∧ ¬NA(S3, MD) ∧ ¬NA(S1, SD) ∧ ¬NA(S3, SD)

NA(S1, VD)

¬NA(S1, MD)

¬NA(S1, SD)

POMERI_SREDNJI_DISK(S1, S2)

NA(S1, SD) ∧ ¬NA(S1, MD) ∧ ¬NA(S2, MD)

NA(S1, SD)

¬NA(S1, MD)

¬NA(S2, MD)

U tekućem stanju zadovoljeno je ¬NA(S2, MD). Da bismo zadovoljili ¬NA(S1, MD) potrebno je ukloniti stav NA(S1,MD) iz tekućeg stanja, pa biramo operator **POMERI_MALI_DISK**(S1,S3). Odredišni stub i ovog puta je izabran tako da se dobije rešenje u najmanjem broju poteza. Sadržaj steka sada je:

CILJNI STEK:

NA(S3,MD) ∧ NA(S3,SD) ∧ NA(S3,VD)

NA(S3,MD)

NA(S3,SD)

NA(S3,VD)

POMERI_VELIKI_DISK(S1,S3)

NA(S1, VD) ∧ ¬NA(S1, MD) ∧ ¬NA(S3, MD) ∧ ¬NA(S1, SD) ∧ ¬NA(S3, SD)

NA(S1, VD)

¬NA(S1, MD)

¬NA(S1, SD)

POMERI_SREDNJI_DISK(S1, S2)

NA(S1, SD) ∧ ¬NA(S1, MD) ∧ ¬NA(S2, MD)

NA(S1, SD)

¬NA(S1, MD)

POMERI_MALI_DISK(S1,S3)

NA(S1, MD)

Pošto je preduslov za primenu poslednjeg izabranog operatora ispunjen, preduslov i operator skidaju se sa steka i ažurira se stanje u skladu sa UKLONI i DODAJ listama operatora pa je stanje sledeće:

TEKUĆE STANJE:

NA(S3,MD) NA(S1,SD) NA(S1,VD)

CILJNI STEK:

NA(S3,MD) ∧ NA(S3,SD) ∧ NA(S3,VD)

NA(S3,MD)

NA(S3,SD)

NA(S3,VD)

POMERI_VELIKI_DISK(S1,S3)

NA(S1, VD) ∧ ¬NA(S1, MD) ∧ ¬NA(S3, MD) ∧ ¬NA(S1, SD) ∧ ¬NA(S3, SD)

NA(S1, VD)

¬NA(S1, MD)

¬NA(S1, SD)

POMERI_SREDNJI_DISK(S1, S2)

NA(S1, SD) ∧ ¬NA(S1, MD) ∧ ¬NA(S2, MD)

NA(S1, SD)

¬NA(S1, MD)

LISTA AKCIJA:

1. **POMERI_MALI_DISK**(S1,S3)

U tekućem stanju zadovoljeni su stavovi ¬NA(S1, MD), NA(S1, SD) kao i složeni uslov za primenu operatora **POMERI_SREDNJI_DISK**(S1, S2) pa se ovi stavovi i pomenuti operator skidaju sa steka. Primenom operatora dobija se:

TEKUĆE STANJE:

NA(S3,MD) NA(S2,SD) NA(S1,VD)

CILJNI STEK:

NA(S3,MD) ∧ NA(S3,SD) ∧ NA(S3,VD)

NA(S3,MD)

NA(S3,SD)

NA(S3,VD)

POMERI_VELIKI_DISK(S1,S3)

NA(S1, VD) ∧ ¬NA(S1, MD) ∧ ¬NA(S3, MD) ∧ ¬NA(S1, SD) ∧ ¬NA(S3, SD)

NA(S1, VD)

¬NA(S1, MD)

¬NA(S1, SD)

LISTA AKCIJA:

1. **POMERI_MALI_DISK**(S1,S3)

2. **POMERI_SREDNJI_DISK**(S1, S2)

U tekućem stanju zadovoljena su vršna tri stava sa steka pa se oni uklanjaju sa steka. Složeni preduslov, NA(S1, VD) ∧ ¬NA(S1, MD) ∧ ¬NA(S3, MD) ∧ ¬NA(S1, SD) ∧ ¬NA(S3, SD), nije međutim zadovoljen jer ne važi stav ¬NA(S3, MD). Ovaj stav bio je zadovoljen u početnom stanju, ali je izvršeno pomeranje malog diska na stub 3; prema tome, zadovoljavanjem jednog podcilja složenog cilja 'pokvarilo' je zadovoljenost drugog podcilja.

Upravo iz ovog razloga pored podciljeva na stek se stavlja i složeni cilj. Rad algoritma nastavlja se tako što ponovo na stek stavljamo stav ¬NA(S3, MD) i biramo operator **POMERI_MALI_DISK**(S3,S2) da bismo zadovoljili ovaj stav. Primetimo da bi pomeranje malog diska sa stuba 3 na stub 1 ponovo poremetilo jedan od uslova za primenu operatora **POMERI_VELIKI_DISK**. Algoritam izbora operatora, treba dakle da uključi i eliminaciju mogućnosti 'klackanja' između dva podcilja tako da zadovoljavanje jednog od njih uvek kvari drugi i obrnuto.

Uslovi za primenu izabranog operatora **POMERI_MALI_DISK**(S3,S2) su zadovoljeni, pa se uslovi i sam operator skidaju sa steka odmah po njihovom stavljanju; primenom operatora dobija se:

TEKUĆE STANJE:

NA(S2,MD) NA(S2,SD) NA(S1,VD)

CILJNI STEK:

NA(S3,MD) ∧ NA(S3,SD) ∧ NA(S3,VD)

NA(S3,MD)

NA(S3,SD)

NA(S3,VD)

POMERI_VELIKI_DISK(S1,S3)

NA(S1, VD) ∧ ¬NA(S1, MD) ∧ ¬NA(S3, MD) ∧ ¬NA(S1, SD) ∧ ¬NA(S3, SD)

¬NA(S3, MD)

LISTA AKCIJA:

1. **POMERI_MALI_DISK**(S1,S3)

2. **POMERI_SREDNJI_DISK**(S1, S2)

3. **POMERI_MALI_DISK**(S3,S2)

U narednim koracima utvrđuje se da su zadovoljeni uslovi za primenu operatora **POMERI_VELIKI_DISK**(S1,S3). Posle primene ovog operatora imamo:

TEKUĆE STANJE:

NA(S2,MD) NA(S2,SD) NA(S3,VD)

CILJNI STEK:

NA(S3,MD) ∧ NA(S3,SD) ∧ NA(S3,VD)

NA(S3,MD)

NA(S3,SD)

NA(S3,VD)

LISTA AKCIJA:

1. **POMERI_MALI_DISK**(S1,S3)
2. **POMERI_SREDNJI_DISK**(S1, S2)
3. **POMERI_MALI_DISK**(S3,S2)
4. **POMERI_VELIKI_DISK**(S1,S3)

Stav NA(S3,VD) je zadovoljen pa se skida sa steka. Da bi se zadovoljio stav NA(S3,SD) bira se operator **POMERI_SREDNJI_DISK**(S2,S3). U narednim koracima algoritma, da bi se zadovoljio uslov za primenu navedenog operatora mali disk uklanja se sa stuba 2 izborom operatora **POMERI_MALI_DISK**(S2,S1). Ovaj operator odmah se primenjuje a posle toga i operator **POMERI_SREDNJI_DISK**(S2,S3). Na kraju se izborom i primenom operatora **POMERI_MALI_DISK**(S1,S3) zadovoljava složeni cilj i svi preostali stavovi bivaju skinuti sa steka tako da je konačno stanje:

TEKUĆE STANJE:

NA(S3,MD) NA(S3,SD) NA(S3,VD)

CILJNI STEK:

-

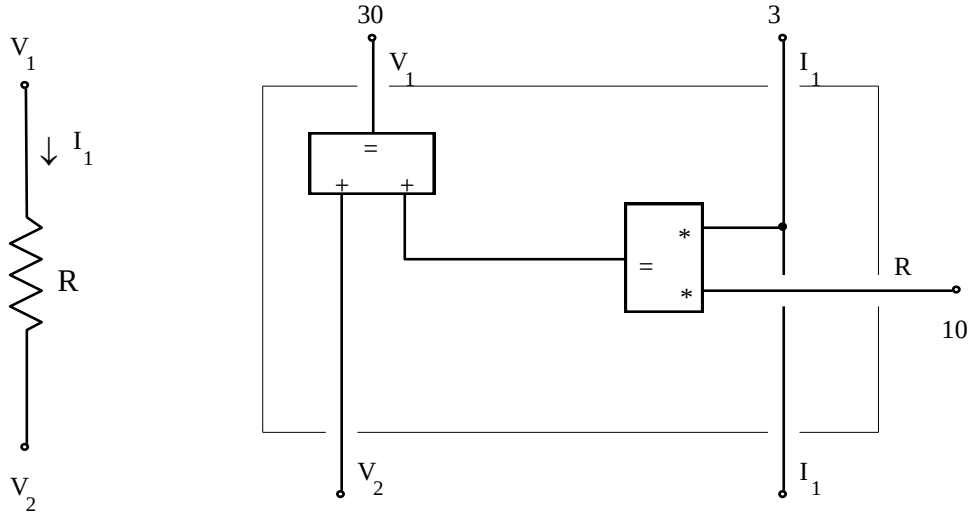
LISTA AKCIJA:

1. **POMERI_MALI_DISK**(S1,S3)
2. **POMERI_SREDNJI_DISK**(S1, S2)
3. **POMERI_MALI_DISK**(S3,S2)
4. **POMERI_VELIKI_DISK**(S1,S3)
5. **POMERI_MALI_DISK**(S2,S1)
6. **POMERI_SREDNJI_DISK**(S2,S3)
7. **POMERI_MALI_DISK**(S1,S3)

3.2. Metod zadovoljenja ograničenja

Zadatak 90: Mreže tipa konstanta-sabirač-množač

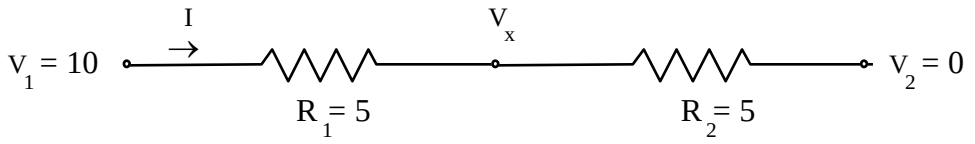
Mreže tipa konstanta-sabirač-množač se mogu iskoristiti za modeliranje izvesnih elektronskih komponenata. Mreža na slici 120 je model otpornika.



Slika 120

a) Izračunati vrednosti nepoznatih veličina na slici 120.

b) Modelirati kolo na slici 121 i pokušati odrediti veličine koje nisu poznate.



Slika 121

c) Kako treba uopštiti proceduru propagacije numeričkih ograničenja da bi se situacija u tački b) mogla obraditi?

Rešenje

d) U slučaju mreža tipa konstanta-sabirač-množač nepoznate veličine mogu se odrediti primenom sledeće procedure:

1. Sve dok postoji neka nepoznata veličina u mreži:

   1.1. Odabrati element (sabirač ili množač) kod koga se na dva od tri izvoda nalaze poznate veličine.

   1.2. Ako takav element ne postoji u mreži procedura se završava neuspehom.

1.3. Ako je element pronađen, izračunati nepoznatu vrednost na trećem izvodu. Ako se radi o izlazu elementa, vrednost se računa kao proizvod (za množač) ili zbir (za sabirač) poznatih vrednosti na ulazima. Ako se radi o jednom od ulaza, vrednost se računa kao količnik vrednosti na izlazu i vrednosti na drugom ulazu (za množač) ili kao razlika vrednosti na izlazu i vrednosti na drugom ulazu (za sabirač).

Primenimo proceduru na mrežu sa slike 120. Potrebno je izračunati veličinu V_2. U prvom koraku procedure biramo množač i računamo vrednost na izlazu 3 * 10 = 30. U drugom koraku može se izabrati sabirač jer su sada poznate vrednosti na izlazu i na desnom ulazu sabirača, pa se tražena vrednost V_2 na preostalom ulazu računa kao razlika tih vrednosti: 30 - 30 = 0. Pošto u mreži više nema nepoznatih vrednosti, procedura se uspešno završava.

b) Koristeći dati model otpornika, kolo sa slike 121 se modelira mrežom prikazanom na slici 122. Nepoznate veličine su struja I i napon V_x. Lako se može utvrditi da nijedan od elemenata na slici 122 nema poznate vrednosti na dva od tri svoja priključka, tako da procedurom primenjenom u tački a) nije moguće odrediti nepoznate veličine. Pri tome, zadat je dovoljan broj poznatih veličina da se problem može rešiti matematički, na primer sistemom jednačina:

$$V_1 - R_1 * I = V_x$$

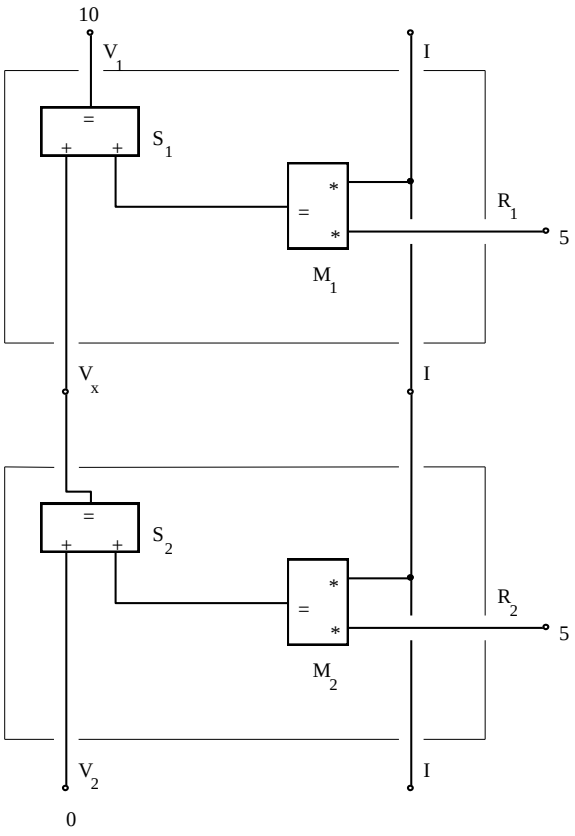
$$V_2 + R_2 * I = V_x$$

uz zamenu poznatih vrednosti:

$$10 - 5 * I = V_x$$

$$0 + 5 * I = V_x$$

Sabiranjem jednačina dobijamo $10 = 2V_x$ to jest $V_x = 5$, pa je $I = V_x / 5 = 1$.



Slika 122

c) Pretpostavimo da je u proceduri u tački a) dozvoljeno, da se pored poznatih veličina koristi i nepoznata veličina jedne od promenljivih, na primer I. Sada možemo odrediti vrednosti na izlazima množača M1 i M2 koje su obe jednake 5I. Vrednost na izlazu sabirača S2 je 0 + 5I = IX. Sada na svakom od ulaza sabirača S1 imamo vrednost 5I, a na izlazu je vrednost 10 pa možemo napisati 5I + 5I = 10 odnosno I = 1. Sada nije teško pronaći preostalu nepoznatu vrednost V_x; na izlazu množača M2 je vrednost 5, pa je na izlazu iz sabirača S2 vrednost 5 + 0 = 5 to jest V_x =5.

Zadatak 91: Problem tri muzičara

Petar, Jovan i Pavle su muzičari. Jedan od njih svira saksofon, drugi gitaru, a treći doboše. Jedan od njih se plaši broja 13, drugi mačaka, a treći se boji visine. Poznato je da su Petar i gitarista paraglajdisti; da Pavle i svirač saksofona vole mačke; i da dobošar živi u stanu broj 13 na trinaestom spratu.

Potrebno je za Petra, Jovana i Pavla utvrditi koji instrument svira svaki od njih i čega se koji od njih boji.

Rešenje

Svaka od osoba u datom problemu opisana je skupom svoje tri osobine: imenom, zanimanjem i strahom. Pri tome svaka od ovih osobina može imati jednu od tri konkretne vrednosti i svaka od ovih vrednosti karakteristična je za jednu osobu (na primer, nemamo dvojicu dobošara). Možemo definisati tri relacije identiteta, po jednu za svaki par osobina. Neka NOT(Petar, gitarista) znači da osoba sa imenom Petar nije gitarista. Činjenice date u postavci iskazuju se na sledeći način:

1. NOT(Petar, gitarista)
2. NOT(Petar, plaši se visine)
3. NOT(gitarista, plaši se visine)
4. NOT(Pavle, plaši se mačaka)
5. NOT(Pavle, saksofonista)
6. NOT(saksofonista, plaši se mačaka)
7. NOT(dobošar, plaši se broja 13)
8. NOT(dobošar, plaši se visine)

Relacije možemo da predstavimo tabelarno. Svaki red i kolona su označeni konkretnim vrednostima neke osobine, NOT ulaz u tabeli označava da je relacija identiteta isključena, a YES označava da relacija identiteta važi. Brojevi u ulazima označavaju činjenicu koja je korišćena da se uspostavi ulaz u tabeli.

	gitarista	saksofonista	dobošar
Petar	NOT,1		
Pavle		NOT,5	
Jovan			

Očigledno da su nam potrebne još dve tabele da bismo predstavili moguće identitete između ljudi i strahova i moguće veze između svirača instrumenata i strahova.

	plaši se broja 13	plaši se mačaka	plaši se visine
Petar			NOT,2
Pavle		NOT,4	
Jovan			

	plaši se broja 13	plaši se mačaka	plaši se visine
gitarista			NOT,3
saksofonista		NOT,6	
dobošar	NOT,7		NOT,8

Sledeća pravila nam kazuju kako se preostali nepopunjeni ulazi u ovim dodatnim tabelama mogu popuniti. Ova pravila izražavaju ograničenje da svaka osoba poseduje svoju karakterističnu vrednost svake od osobina.

1. IF svi ulazi u jednoj vrsti su NOT izuzev jednog THEN preostali je YES
2. IF jedan ulaz u vrsti je YES THEN svi ostali u toj vrsti su NOT
3. IF svi ulazi u jednoj koloni su NOT izuzev jednog THEN preostali je YES
4. IF jedan ulaz u koloni je YES THEN svi ostali u toj koloni su NOT
5. IF važi YES(x,y) i NOT(y,z) THEN može se zaključiti NOT(x,z).

Uz data pravila nije teško popuniti tabele. U svakom koraku rešavanja problema, popunjavanjem nekog ulaza smanjuje se broj mogućih vrednosti za neku osobinu neke od osoba, sve dok ne utvrdimo tačnu vrednost. Pravila popunjavanja izraz su ograničenja koja su svojstvena problemu.

Rešavanje počinjemo popunjavanjem treće tabele. Prema pravilu 3, utvrđujemo da se saksofonista plaši visine. Prema pravilu 2 imamo NOT(saksofonista, plaši se broja 13). Prema pravilu 3 imamo YES(gitarista, plaši se broja 13). Prema pravilu 1 imamo YES(dobošar, plaši se mačaka). Prema pravilu 2, imamo NOT(gitarista, plaši se mačaka). Ovim je treća tabela potpuno popunjena.

	plaši se broja 13	plaši se mačaka	plaši se visine
gitarista	YES	NOT	NOT,3
saksofonista	NOT	NOT,6	YES
dobošar	NOT,7	YES	NOT,8

Sada ćemo popunjavati prvu tabelu. Pošto važi YES(plaši se mačaka, dobošar) i NOT(plaši se mačaka, Pavle) prema pravilu 5 imamo da je NOT(dobošar, Pavle) pa popunjavamo odgovarajući ulaz prve tabele. Prema pravilu 1 imamo YES(Pavle, gitarista), a prema pravilu 4 imamo NOT(Jovan, gitarista). Prema pravilu 5, pošto važi YES(plaši se visine, saksofonista) i NOT(plaši se visine, Petar) zaključujemo NOT(Petar, saksofonista). Prema

pravilu 3, zaključujemo YES(Jovan, saksofonista), a prema pravilu 1, imamo da je YES(Petar, dobošar). Prema pravilu 2 je NOT(Jovan, dobošar) čime je i tabela 1 potpuno popunjena.

	gitarista	saksofonista	dobošar
Petar	NOT,1	NOT	YES
Pavle	YES	NOT,5	NOT
Jovan	NOT	YES	NOT

Ostala je još druga tabela da se popuni. Prema pravilu 5, iz YES(plaši se broja 13, gitarista) i NOT(gitarista, Petar) sledi NOT(Petar, plaši se broja 13). Prema pravilu 1, sada imamo YES(Petar, plaši se mačaka). Prema pravilu 4, imamo NOT(Jovan, plaši se mačaka). Prema pravilu 5, pošto je YES(plaši se visine, saksofonista) i NOT(saksofonista, Pavle) imamo NOT(Pavle, plaši se visine). Prema pravilu 1, imamo YES(Pavle, plaši se broja 13), a prema pravilu 3 imamo YES(Jovan, plaši se visine). Ostalo je još samo da se prema pravilu 4 utvrdi NOT(Jovan, plaši se broja 13) i druga tabela je popunjena.

	plaši se broja 13	plaši se mačaka	plaši se visine
Petar	NOT	YES	NOT,2
Pavle	YES	NOT,4	NOT
Jovan	NOT	NOT	YES

Sada možemo da utvrdimo da je:

- Petar dobošar i plaši se mačaka
- Pavle gitarista i plaši se broja 13 i
- Jovan saksofonista i plaši se visine.

Zadatak 92: Problem interpretacije snimaka

Računari se mogu upotrebiti za analiziranje snimaka izviđačkih aviona ili satelita da bi od velikog broja snimaka izdvojili mali broj snimaka koji mogu biti zanimljivi. Preliminarno video procesiranje grupiše delove slike u regione (preciznije, određuje ivice regiona) na osnovu sličnosti u boji, osvetljenosti i teksturi delova slike. Drugi deo problema je interpretacija slike, to jest određivanje šta svaki od dobijenih regiona može da predstavlja. Problem interpretacije slike pogodan je za rešavanje strategijom zadovoljavanja ograničenja.

Na slici 123 prikazane su granice 5 regiona. Ovu sliku potrebno je interpretirati poštujući pri tome sledeća ograničenja:

C1. Svaki od regiona predstavlja ili travu ili vodu ili cestu ili kuću ili vozilo.

C2. Region se ne može graničiti ili biti unutar regiona iste vrste (u suprotnom ne bi se videla ivica).

C3. Kuće i vozila ne mogu se graničiti niti biti unutar regiona vode (pretpostavljamo da voda nije dovoljno duboka za čamce).

C4. Vozila se moraju graničiti ili biti unutar regiona ceste, ali cesta ne može biti unutar regiona vozila.

C5. Vozila ne mogu biti unutar kuća.

C6. Trava ne može biti unutar vozila ili kuća.

C7. Cesta ne može biti potpuno unutar drugog regiona (pošto su ceste skoro uvek povezane s drugim cestama).

C8. Kuće, vozila i ceste su pravilni (s pravolinijskim ivicama) regioni.

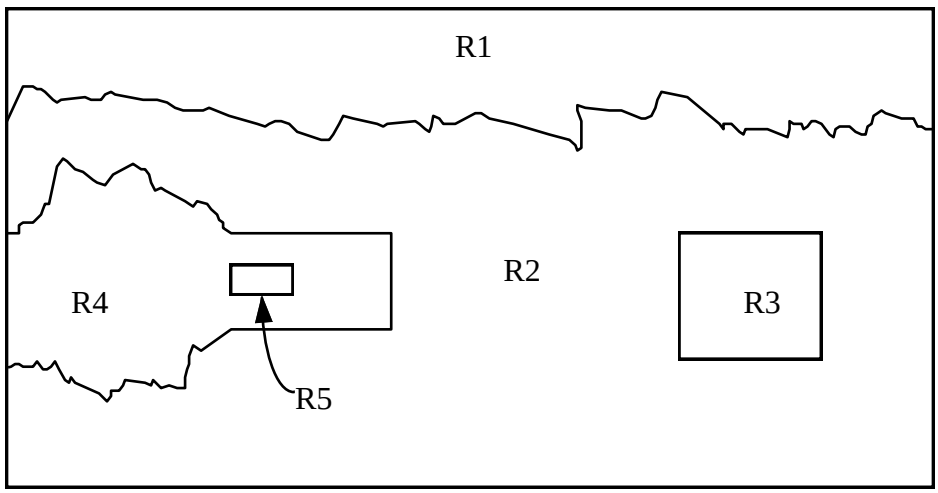
C9. Voda i trava su nepravilni regioni.

C10. Vozilo je region male veličine.

a) Napisati produkcioni sistem koji odgovara datim ograničenjima.

b) Napisati upit u produkcioni sistem pod a) za interpretaciju slike 123.

c) Metodom proste relaksacije pronaći odgovor na upit pod b).



Slika 123

Rešenje

a) Za interpretaciju slike dovoljni su nam sledeći predikati:

- Susedni(x,y) i Nesusedni(x,y) označavaju da se vrste regiona x i y mogu, odnosno ne mogu graničiti;
- Unutar(x,y) i NijeUnutar(x,y) označavaju da region vrste x jeste, odnosno nije, unutar regiona vrste y;
- Pravilan(x) i Nepravilan(x) označavaju da je region vrste x pravilnog, odnosno nepravilnog oblika;
- Mali(x) i Veliki(x) označavaju da je region vrste x mali, odnosno veliki.

Ograničenje C1 opisuje skupove mogućih vrednosti za promenljive R1 do R5 koje predstavljaju pojedine regione na slici:

R1 = {Trava, Voda, Cesta, Kuća, Vozilo}

R2 = {Trava, Voda, Cesta, Kuća, Vozilo}

R3 = {Trava, Voda, Cesta, Kuća, Vozilo}

R4 = {Trava, Voda, Cesta, Kuća, Vozilo}

R5 = {Trava, Voda, Cesta, Kuća, Vozilo}

Ograničenja C2 do C10 data u postavci mogu se iskazati sledećim činjenicama:

C2:

Nesusedni(x,x)

NijeUnutar(x,x)

Napomena: U gornjim predikatima x predstavlja proizvoljnu vrstu regiona (univerzalno kvantifikovanu promenljivu).

C3:

Nesusedni(Kuća,Voda)

Nesusedni(Voda,Kuća)

Nesusedni(Vozilo,Voda)

Nesusedni(Voda,Vozilo)

C4:

Susedni(Vozilo,Cesta)

Unutar(Vozilo,Cesta)

Susedni(Cesta,Vozilo)

NijeUnutar(Cesta,Vozilo)

C5:

NijeUnutar(Vozilo,Kuća)

C6:

NijeUnutar(Trava,Vozilo)

NijeUnutar(Trava,Kuća)

C7:

NijeUnutar(Cesta,y), gde je y region proizvoljne vrste

C8:

Pravilan(Kuća)

Pravilan(Vozilo)

Pravilan(Cesta)

C9:

Nepravilan(Voda)

Nepravilan(Trava)

C10:

Mali(Vozilo)

Prethodnim činjenicama treba dodati sledeća pravila:

P1: if not Nesusedni(x,y) then Susedni(x,y)

P2: if not NijeUnutar(x,y) then Unutar(x,y)

sa značenjem da su predikati Susedni, odnosno Unutar zadovoljeni (za regione x i y proizvoljne vrste) ako ne podležu nekom od prethodnih ograničenja.

b) Slika 123 opisana je predikatskom formulom

Veliki(R1) and Veliki(R2) and Veliki(R3) and Veliki(R4) and Mali(R5) and

Pravilan(R3) and Pravilan(R5) and Nepravilan(R1) and Nepravilan(R2) and

Susedni(R1,R2) and Susedni(R2,R4) and Unutar(R3,R2) and Unutar(R5,R4)

koja predstavlja upit u bazu znanja iz tačke a) sa ciljem da se dobiju konkretne vrednosti za promenljive R1 do R5. Baza znanja sadrži opšta ograničenja za interpretaciju proizvoljne slike, a upit sadrži ograničenja vezana za samu sliku.

Treba primetiti da u gornjoj formuli nismo klasifikovali region R4 ni kao pravilan ni kao nepravilan zbog njegovog neobičnog oblika.

c) Rešavanje ovakvog tipa problema klasičnim metodima zaključivanja sa direktnim ili povratnim ulančavanjem nije efikasno jer, zbog složenog upita u koji je ugrađeno mnoštvo ograničenja, dolazi mnogo puta do vraćanja pri zaključivanju. Za rešavanje ovog problema na efikasniji način koristićemo algoritam proste relaksacije naveden u dodatku 1 (algoritam 16) koji ima za cilj da smanji skup mogućih vrednosti za svaku od promenljivih čije nas vrednosti interesuju.

Primenimo algoritam na postavljeni problem. Inicijalni skupovi vrednosti i statusi promenljivih R1 do R5 iz upita su:

R1 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

R2 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

R3 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

R4 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

R5 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

Izaberimo, prema koraku 3. algoritma, promenljivu R1 za razmatranje. Razmotrimo da li je V=Trava moguća vrednost. Prvi predikat iz upita koji pominje R1 je Veliki(R1) i ispunjen je. Razmatramo drugi predikat, Nepravilan(R1); i ovaj predikat je ispunjen, pa razmatramo treći predikat, Susedni(R1,R2) je ispunjeno za R2 = Voda.

Pošto su svi predikati upita za R1=Trava ispunjeni, ovu mogućnost ne možemo udaljiti iz liste za R1. Razmatranjem sledeće mogućnosti R1=Voda, ustanovljavamo da su i za nju svi predikati upita ispunjeni, pa je ne možemo izbaciti.

Razmotrimo R1=Cesta. Veliki(R1) je ispunjeno, ali nije i Nepravilan(R1), pa izbacujemo Cesta iz skupa za R1.

Razmatranjem R1=Kuća i R1=Vozilo ustanovljavamo da i ove mogućnosti treba izbaciti iz skupa za R1. Završeno je razmatranje promenljive R1, obeležavamo je kao neaktivnu. Nema potrebe obeležiti neku drugu promenljivu kao aktivnu. Situacija je sada sledeća:

R1 = {Trava, Voda}, neaktivna

R2 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

R3 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

R4 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

R5 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

Razmotrimo promenljivu R2 sledeću. Predikati iz upita koji se odnose na R2 su:

Veliki(R2), Nepravilan(R2), Susedni(R1,R2), Susedni(R2,R4) i Unutar(R3,R2).

Za R2=Trava svi predikati su zadovoljeni (svaki od njih se razmatra nezavisno od drugih), pošto važi Susedni(Trava,Voda), Susedni(Voda,Trava) i Unutar (Voda, Trava). Isto važi i za R2=Voda (važi, na primer Unutar(Trava, Voda)). Dalje ustanovljavamo da R2 ne može biti ni Cesta ni Kuća ni Vozilo jer nisu ispunjeni predikati Nepravilan(R2), ili u slučaju vozila, Veliki(R2). Posle ažuriranja statusa promenljivih situacija je:

R1 = {Trava, Voda}, aktivna

R2 = {Trava, Voda}, neaktivna

R3 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

R4 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

R5 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

Promenljiva R2 je proglašena neaktivnom a R1 aktivnom prema koraku 3.3. Izaberimo ponovo promenljivu R1 (strategija je, na primer da se od aktivnih promenljivih R_i uzme ona sa najmanjim indeksom i). Razmatranjem ove promenljive ne dovodi do eliminacije neke od njenih vrednosti, pa je proglašavamo neaktivnom i prelazimo na sledeću aktivnu promenljivu R3.

Potrebno je zadovoljiti predikate Veliki(R3), Pravilan(R3) i Unutar(R3,R2). Prvi predikat eliminiše vrednost Vozilo, drugi vrednosti Trava i Voda, a treći vrednost Cesta. Ostaje samo vrednost R3=Kuća (zadovoljeno je, na primer Unutar(Kuća, Trava)). Promenljivu R3 označavamo kao neaktivnu, a R2 kao aktivnu pa imamo da je:

R1 = {Trava, Voda}, neaktivna

R2 = {Trava, Voda}, aktivna

R3 = {Kuća}, neaktivna

R4 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

R5 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

Zamenom vrednosti za R3 u upitu on dobija oblik:

Veliki(R1) and Veliki(R2) and Veliki(R4) and Mali(R5) and

Pravilan(R5) and Nepravilan(R1) and Nepravilan(R2) and

Susedni(R1,R2) and Susedni(R2,R4) and Unutar(Kuća,R2) and Unutar(R5,R4)

Eliminisani su predikati koji ne sadrže promenljivu.

Ponovnim razmatranjem promenljive R2 možemo iz njenog skupa vrednosti eliminisati Voda jer važi predikat Unutar(Kuća,R2). R2 postaje neaktivna, a R1 aktivna promenljiva pa imamo:

R1 = {Trava, Voda}, aktivna

R2 = {Trava}, neaktivna

R3 = {Kuća}, neaktivna

R4 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

R5 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

Izgled upita je:

Veliki(R1)and Veliki(R4) and Mali(R5) and

Pravilan(R5) and Nepravilan(R1) and

Susedni(R1,Trava) and Susedni(Trava,R4) and Unutar(R5,R4)

Razmatranjem R1 eliminiše se vrednost R1 = Trava jer ne važi Susedni(Trava,Trava). Sada su vrednosti promenljivih:

R1 = {Voda}, neaktivna

R2 = {Trava}, neaktivna

R3 = {Kuća}, neaktivna

R4 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

R5 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

Izgled upita je:

Veliki(R4) and Mali(R5) and Pravilan(R5) and Susedni(Trava,R4) and Unutar(R5,R4)

Razmatramo promenljivu R4. Vrednost Trava se eliminiše jer ne važi Susedni(Trava,R4). Vrednost R4=Voda ostaje jer važi svaki od predikata Veliki(R4), Susedni(Trava,R4) i Unutar(R5,R4) (za R5=Trava). Vrednost R4=Cesta takođe ostaje (na primer važi Susedni(Trava,Cesta) i Unutar(Vozilo,Cesta)). Vrednost R4=Kuća ostaje (važi Susedni(Trava, Kuća) i Unutar(Voda,Kuća)). Vrednost R4=Vozilo se eliminiše jer nije Veliki(Vozilo). Sada je:

R1 = {Voda}, neaktivna

R2 = {Trava}, neaktivna

R3 = {Kuća}, neaktivna

R4 = {Voda, Cesta, Kuća}, neaktivna

R5 = {Trava, Voda, Cesta, Kuća, Vozilo}, aktivna

Upit je neizmenjen:

Veliki(R4) and Mali(R5) and Pravilan(R5) and Susedni(Trava,R4) and Unutar(R5,R4)

Razmatramo R5. Vredosti Trava, Voda, Cesta i Kuća se eliminišu iz skupa mogućih vrednosti jer za njih ne važi Mali(R5). Vrednost Vozilo ostaje jer važi Mali(Vozilo), Pravilan(Vozilo) i Unutar(Vozilo,Cesta). Situacija je sada sledeća:

R1 = {Voda}, neaktivna

R2 = {Trava}, neaktivna

R3 = {Kuća}, neaktivna

R4 = {Voda, Cesta, Kuća}, aktivna

R5 = {Vozilo}, neaktivna

Upit glasi:

Veliki(R4) and Susedni(Trava,R4) and Unutar(Vozilo,R4)

Razmatra se jedina aktivna promenljiva R4. Predikat Unutar(Vozilo,R4) zadovoljen je samo za R4= Cesta čime je nađena jedinstvena interpretacija za svaki od regiona: R1 je Voda, R2 Trava, R3 Kuća, R4 Cesta a R5 Vozilo.

Relaksacija je pojam preuzet iz matematike. U gornjem algoritmu koristi se diskretna relaksacija pri čemu se za svaku promenljivu razmatra konačan skup mogućih vrednosti i u svakoj iteraciji se, korišćenjem postavljenih ograničenja (relaksaciona formula u matematici), smanjuje taj skup vrednosti. Radi se o čisto iterativnom metodu koji prethodi klasičnom pretraživanju prostora mogućih vrednosti promenljivih).

U prethodnom primeru nije bilo potrebno primeniti pretraživanje jer su relaksacijom dobijene jednoznačne interpretacije regiona. Ukoliko bi se desilo da se primenom relaksacije eliminišu SVE vrednosti iz skupa mogućih vrednosti za neku od promenljivih, treba razmotriti i preraditi polazna ograničenja jer su prestrogo postavljena.

Zadatak 93: Kriptoaritmetički problem

U sledećoj operaciji sabiranja dva dekadna broja cifre su zamenjene slovima tako da različitim slovima odgovaraju različite cifre. Metodom proste relaksacije pronaći koji broj predstavlja svaka od cifara. Usvojiti da je E = 5.

```
  SEND
+ MORE
 MONEY
```

Rešenje

Za rešavanje problema biće upotrebljen algoritam proste relaksacije naveden u dodatku 1. Problem se može opisati na sledeći način:

Aktivne: M S O C100 N C10 R C1 D Y	Neaktivne: E
M ∈ {1, 2, 3, 4, 5, 6, 7, 8, 9}	
S ∈ {1, 2, 3, 4, 5, 6, 7, 8, 9}	Ograničenja:
O ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	
E ∈ {5}	O1: D + 5 = Y + 10C1
N ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O2: N + R + C1 = 5 + 10C10

R ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O3: 5 + O + C10 = N + 10C100
D ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O4: S + M + C100 = O + 10M
Y ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	
C100 ∈ {0, 1}	O5: M, S, O, N, 5, R, D, Y su svi
C10 ∈ {0, 1}	međusobno različiti
C1 ∈ {0, 1}	

Skupovi mogućih vrednosti za pojedine promenljive kao i ograničenje O5 slede iz uslova zadatka, a ograničenja O1 do O4 se dobijaju primenom elementarne matematike. Promenljive C1, C10 i C100 predstavljaju prenose iz razreda jedinica, desetica i stotina respektivno. U ograničenjima je zamenjena promenljiva E vrednošću 5 prema uslovu zadatka. Inicijalno sve promenljive imaju status aktivnih promenljivih. Usvojićemo heuristiku da od aktivnih promenljivih za razmatranje biramo onu koja predstavlja cifru ili prenos najveće težine u zadatoj sumi.

- Od aktivnih promenljivih, heuristički biramo M za razmatranje; iz ograničenja O4 sledi M=1 jer ako je M > 1 onda je desna strana O4 veća do jednaka 20, a ne postoji kombinacija za S + M + C100 čija je vrednost veća ili jednaka 20. M se označava neaktivnim.

Aktivne: S O C100 N C10 R C1 D Y	Neaktivne: E M
M ∈ {1}	
S ∈ {1, 2, 3, 4, 5, 6, 7, 8, 9}	Ograničenja:
O ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	
N ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O1: D + 5 = Y + 10C1
E ∈ {5}	O2: N + R + C1 = 5 + 10C10
R ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O3: 5 + O + C10 = N + 10C100
D ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O4: S + 1 + C100 = O + 10
Y ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	
C100 ∈ {0, 1}	O5: 1, S, O, N, 5, R, D, Y su svi
C10 ∈ {0, 1}	međusobno različiti
C1 ∈ {0, 1}	

- Bira se S za razmatranje; da bi O4 bilo zadovoljeno, S mora imati vrednost 8 ili 9 jer je desna strana O4 veća ili jednaka 10. Drugo ograničenje koje pominje O5 je S i ono ne doprinosi daljoj redukciji mogućih vrednosti. S se označava neaktivnim.

Aktivne: O C100 N C10 R C1 D Y	Neaktivne: E M S
M ∈ {1}	
S ∈ {8, 9}	Ograničenja:
O ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	

$N \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$	O1: D + 5 = Y + 10C1
$E \in \{5\}$	O2: N + R + C1 = 5 + 10C10
$R \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$	O3: 5 + O + C10 = N + 10C100
$D \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$	O4: S + 1 + C100 = O + 10
$Y \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$	
$C100 \in \{0, 1\}$	O5: 1, S, O, N, 5, R, D, Y su svi
$C10 \in \{0, 1\}$	međusobno različiti
$C1 \in \{0, 1\}$	

- Bira se O za razmatranje; na osnovu O3 ne može se eliminisati nijedna od mogućih vrednosti za O. Iz O4 sledi da je O jednako ili 0 ili 1 jer je S ili 8 ili 9 a C100 je 0 ili 1. Iz O5 sledi sa O ne može biti 1 (jer je M=1). Znači da je O = 0. Obeležavamo O neaktivnim, a S aktivnim, jer se pojavljuje u ograničenjima zajedno sa O, a nije mu utvrđena konačna vrednost.

Aktivne: S C100 N C10 R C1 D Y	Neaktivne: E M O
$M \in \{1\}$	
$S \in \{8, 9\}$	Ograničenja:
$O \in \{0\}$	
$N \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$	O1: D + 5 = Y + 10C1
$E \in \{5\}$	O2: N + R + C1 = 5 + 10C10
$R \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$	O3: 5 + C10 = N + 10C100
$D \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$	O4: S + 1 + C100 = 10
$Y \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$	
$C100 \in \{0, 1\}$	O5: 1, S, 0, N, 5, R, D, Y su svi
$C10 \in \{0, 1\}$	međusobno različiti
$C1 \in \{0, 1\}$	

- Biramo S za razmatranje; ne može se eliminisati nijedna od vrednosti za S pa ga označavamo neaktivnim.
- Biramo C100 za razmatranje. Prema O3 mora biti C100 = 0, u suprotnom bi desna strana O3 bila veća od 10. Obeležavamo C100 neaktivnim, a S aktivnim (jer se pominje u O4 zajedno sa C100).

Aktivne: S N C10 R C1 D Y	Neaktivne: E M O C100
$M \in \{1\}$	
$S \in \{8, 9\}$	Ograničenja:
$O \in \{0\}$	
$N \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$	O1: D + 5 = Y + 10C1

E ∈ {5}	O2: N + R + C1 = 5 + 10C10
R ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O3: 5 + C10 = N
D ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O4: S + 1 = 10
Y ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	
C100 ∈ {0}	O5: 1, S, 0, N, 5, R, D, Y su svi
C10 ∈ {0, 1}	međusobno različiti
C1 ∈ {0, 1}	

- Biramo S za razmatranje; iz O4 sledi da je S jednako 9. Obeležavamo S neaktivnim.

Aktivne: N C10 R C1 D Y	Neaktivne: E M O C100 S
M ∈ {1}	
S ∈ {9}	Ograničenja:
O ∈ {0}	
N ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O1: D + 5 = Y + 10C1
E ∈ {5}	O2: N + R + C1 = 5 + 10C10
R ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O3: 5 + C10 = N
D ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O4: 10 = 10
Y ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	
C100 ∈ {0}	O5: 1, 9, 0, N, 5, R, D, Y su svi
C10 ∈ {0, 1}	međusobno različiti
C1 ∈ {0, 1}	

- Biramo N za razmatranje; N se pominje u O2, O3 i O5. O2 nije od koristi. Prema O3, sledi da je N jednako 5 ili 6. Prema O5 otpada vrednost N = 5 pa je N =6. N označavamo neaktivnim.

Aktivne: C10 R C1 D Y	Neaktivne: E M O C100 S N
M ∈ {1}	
S ∈ {9}	Ograničenja:
O ∈ {0}	
N ∈ {6}	O1: D + 5 = Y + 10C1
E ∈ {5}	O2: 6 + R + C1 = 5 + 10C10
R ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O3: 5 + C10 = 6
D ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O4: 10 = 10
Y ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	
C100 ∈ {0}	O5: 1, 9, 0, 6, 5, R, D, Y su svi

C10 ∈ {0, 1}	međusobno različiti
C1 ∈ {0, 1}	

- Biramo C10 za razmatranje; iz O3 sledi da mora biti C10 jednako 1. Obeležavamo C10 neaktivnim.

Aktivne: R C1 D Y	Neaktivne: E M O C100 S N C10
M ∈ {1}	
S ∈ {9}	Ograničenja:
O ∈ {0}	
N ∈ {6}	O1: D + 5 = Y + 10C1
E ∈ {5}	O2: 6 + R + C1 = 15
R ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O3: 6 = 6
D ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O4: 10 = 10
Y ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	
C100 ∈ {0}	O5: 1, 9, 0, 6, 5, R, D, Y su svi
C10 ∈ {1}	međusobno različiti
C1 ∈ {0, 1}	

- Biramo R za razmatranje. Prema O3 važi da je R + C1 = 9, pa zaključujemo da je R jednako 8 ili 9. Prema O5 R ne može biti jednako 9, pa je R = 8. Obeležavamo R neaktivnim.

Aktivne: C1 D Y	Neaktivne: E M O C100 S N C10 R
M ∈ {1}	
S ∈ {9}	Ograničenja:
O ∈ {0}	
N ∈ {6}	O1: D + 5 = Y + 10C1
E ∈ {5}	O2: 14 + C1 = 15
R ∈ {8}	O3: 6 = 6
D ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O4: 10 = 10
Y ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	
C100 ∈ {0}	O5: 1, 9, 0, 6, 5, 8, D, Y su svi
C10 ∈ {1}	međusobno različiti
C1 ∈ {0, 1}	

- Biramo C1 za razmatranje. Prema O2, imamo da je C1 jednako 1. Obeležavamo C1 neaktivnim.

Aktivne: D Y	Neaktivne: E M O C100 S N C10 R C1
M ∈ {1}	
S ∈ {9}	Ograničenja:
O ∈ {0}	
N ∈ {6}	O1: D + 5 = Y + 10
E ∈ {5}	O2: 15 = 15
R ∈ {8}	O3: 6 = 6
D ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	O4: 10 = 10
Y ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	
C100 ∈ {0}	O5: 1, 9, 0, 6, 5, 8, D, Y su svi
C10 ∈ {1}	međusobno različiti
C1 ∈ {1}	

- Biramo D za razmatranje. Prema O1, važi da je D = Y + 5 pa sledi da je D veće od 4. Od svih vrednosti za D ograničenje O5 dozvoljava samo D = 7. Obeležavamo D neaktivnim.

Aktivne: Y	Neaktivne: E M O C100 S N C10 R C1 D
M ∈ {1}	
S ∈ {9}	Ograničenja:
O ∈ {0}	
N ∈ {6}	O1: 7 + 5 = Y + 10
E ∈ {5}	O2: 15 = 15
R ∈ {8}	O3: 6 = 6
D ∈ {7}	O4: 10 = 10
Y ∈ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}	
C100 ∈ {0}	O5: 1, 9, 0, 6, 5, 8, 7, Y su svi
C10 ∈ {1}	međusobno različiti
C1 ∈ {1}	

- Biramo Y za razmatranje. Prema O1 sledi da je Y = 2, što ne protivreči ograničenju O5. Označavamo Y neaktivnim. Algoritam završava rad jer nema više aktivnih promenljivih. Pošto su sve cifre određene, a u skupu ograničenja nije se pojavila protivrečnost, konačno rešenje je pronađeno:

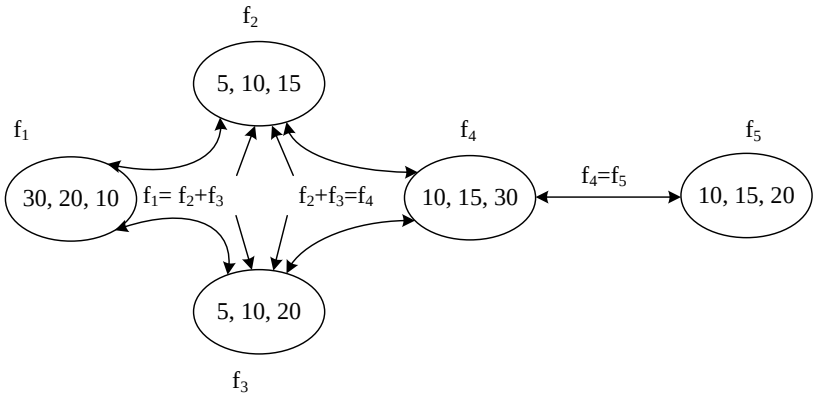
$$\begin{array}{r} 9567 \\ +\ 1085 \\ \hline 10652 \end{array}$$

Čitaocu se preporučuje da problem pokuša rešiti primenom algoritma proste relaksacije bez korišćenja uslova E = 5 iz postavke. Konačno rešenje neće moći biti nađeno, jer algoritam

proste relaksacije razmatra svako od ograničenja zasebno, zanemarujući pri tom vezivanja promenljivih nastala pri prethodnim ograničenjima.

Zadatak 94: Maksimiziranje protoka

Nina rešava problem maksimiziranja protoka kroz neku mrežu (slika 124). Sledeća mreža prikazuje ograničenja među čvorovima koja su uočena tom prilikom. Obratiti pažnju da postoje dva ograničenja koja povezuju po tri čvora.



Slika 124

U datoj mreži ograničenja, promenljive su celobrojne vrednosti koje predstavljaju protok kroz svaki od čvorova. Sva poznata ograničenja predstavljena su na dijagramu. Domen za svaku promenljivu naznačen je u odgovarajućem čvoru. Metodom proste relaksacije pronaći protok kroz svaki čvor. Da li je problem rešen upotrebom pomenutog metoda? Šta se primenjuje ukoliko problem nemože biti rešen metodom proste relaksacije.

Rešenje

Kod metoda proste relaksacije, posmatramo svaku promenljivu zasebno i pokušavamo da eliminišemo što više dozvoljenih vrednosti za tu promenljivu na osnovu ograničenja, uz eventualna vezivanja drugih promenljivih.

Na početku, svaka promenljiva je aktivna (znači da će biti obrađena u nekom od narednih koraka) i za svaku promenljivu su dozvoljene sve vrednosti iz njenog domena.

$f1 \in \{10, 20, 30\}$

$f2 \in \{5, 10, 15\}$

$f3 \in \{5, 10, 20\}$

$f4 \in \{10, 15, 30\}$

$f5 \in \{10, 15, 20\}$

Na osnovu slike 124, definišemo tri ograničenja:

1. f1 = f2 + f3

2. f4 = f2 + f3

3. f4 = f5

Biramo najpre promenljivu f5. Vrednosti 10 i 15 zadovoljavaju ograničenje 3, ali ne i vrednost 20 koju brišemo. Promenljivu proglašavamo neaktivnom. Na sličan način obrađujemo i promenljivu f4, brišući moguću vrednost 30.

Biramo promenljivu f3. Vrednost 20 ne može da zadovolji ograničenje 2 (to utvrđujemo uparujući ovu vrednost sa svim mogućih vrednostima promenljive f2 i posmatrajući da li se rezultat nalazi među mogućim vrednostima za f4), pa je brišemo. Promenljiva f4 je postala aktivna, jer su f3 i f4 pomenute u istom ograničenju. Na sličan način odbacujemo vrednost 15 za promenljivu f2, nakon što smo je uzeli u razmatranje. Promenljiva f3 je opet aktivna.

Biramo promenljivu f1. Vrednost 30 ne može da zadovolji ograničenje 1 (probamo sve moguće parove vrednosti za f2 i f3), pa je brišemo. Promenljiva f3 je postala aktivna.

Do kraja ne možemo da obrišemo nijednu drugu vrednost, pa se algoritam prekida. Rezultat je:

$f1 \in \{10, 20\}$

$f2 \in \{5, 10\}$

$f3 \in \{5, 10\}$

$f4 \in \{10, 15\}$

$f5 \in \{10, 15\}$

Problem nije rešen ovom metodom jer nismo odredili jednoznačnu vrednost za svaku promenljivu, ali smo suzili skup mogućih vrednosti. Dalje možemo da primenimo neki od algoritama pretrage da bismo našli rešenje.

Zadatak 95: Raspoređivanje vozova

Razmotrimo sledeći problem raspoređivanja vozova:

Date su 4 kompozicije T1, T2, T3 i T4 i 3 lokomotive: L1, L2 i L3. Tabela 8 prikazuje vremenski raspored upotrebe za svaku kompoziciju.

Kompozicija	Interval upotrebe
T1	8:00 – 10:00
T2	9:00 – 13:00
T3	10:00 – 11:00
T4	11:00 – 15:00

Tabela 8

Dodatno su poznata sledeća ograničenja:

1. Svaki voz mora vući neka lokomotiva
2. Svaka lokomotiva može vući samo jedan voz u jednom trenutku
3. Ako lokomotiva nije u upotrebi, može se trenutno upotrebiti za bilo koji voz

4. Lokomotiva L3 je suviše mala da vuče T3 i T1
5. Lokomotive L2 i L3 su suviše male da vuku T4
6. Lokomotiva L1 je suviše mala da vuče T1

Metodom zadovoljenja ograničenja (algoritmom proste relaksacije) naći koja lokomotiva treba da vuče koji voz.

Rešenje

Aktivne: T1, T2, T3, T4
Neaktivne: -
T1 ∈ {L1, L2, L3}
T2 ∈ {L1, L2, L3}
T3 ∈ {L1, L2, L3}
T4 ∈ {L1, L2, L3}

Prva u listi aktivnih je kompozicija T1. Na osnovu pravila 6 iz liste mogućih lokomotiva možemo izostaviti L1. Takodje na osnovu pravila 4 možemo izostaviti lokomotivu L3. Jedina lokomotiva koja može vući kompoziciju T1 je L2. T1 se pomera u listu neaktivnih promenljivih.

Aktivne: T2, T3, T4
Neaktivne: T1
T1 ∈ {L2}
T2 ∈ {L1, L2, L3}
T3 ∈ {L1, L2, L3}
T4 ∈ {L1, L2, L3}

Naredna za razmatranje je kompozicija T2. Jedino ograničenje koje možemo primeniti u ovom trenutku nad T2 dato je u tabeli 0. Naime, potrebno je primetiti da se vremena za transport kompozicije T1 i T2 preklapaju (period od 9:00 – 10:00), tako da se lokomotiva L2, jedina koja može vući T1 ne može upotrebiti za vuču T2 kompozicije.

Aktivne: T3, T4
Neaktivne: T1, T2
T1 ∈ {L2}
T2 ∈ {L1, L3}
T3 ∈ {L1, L2, L3}
T4 ∈ {L1, L2, L3}

Trenutno je aktivna kompozicija T3. Razmatranjem pravila 4 može se zaključiti da L3 nije u mogućnosti da vuče ovu kompoziciju. Kompozicija T3 prelazi u listu neaktivnih, dok kompozicija T2 koja je povezana sa kompozicijom preko intervala prelazi u listu aktivnih promenljivih.

Aktivne: T4, T2

Neaktivne: T1, T3

T1 ∈ {L2}

T2 ∈ {L1, L3}

T3 ∈ {L1, L2}

T4 ∈ {L1, L2, L3}

Na osnovu pravila 5, jednostavno je zaključiti da jedino L1 može vući kompoziciju T4. T4 postaje neaktivna dok T3 prelazi u listu aktivnih promenljivih.

Aktivne: T2, T3

Neaktivne: T1, T4

T1 ∈ {L2}

T2 ∈ {L1, L3}

T3 ∈ {L1, L2}

T4 ∈ {L1}

S obzirom da se vremena transporta kompozicija T2 i T4 preklapaju, a jedina lokomotiva koja može vući T4 je L1, iz liste mogućih vrednosti za T2 uklanjamo ovu vrednost.

Aktivne: T3

Neaktivne: T1, T4, T2

T1 ∈ {L2}

T2 ∈ {L3}

T3 ∈ {L1, L2}

T4 ∈ {L1}

Razmatranjem ograničenja nad trenutno aktivnom promenljivom, T3, može se zaključiti da njen skup vrednosti ostaje ne izmenjen.

Aktivne: -

Neaktivne: T1, T4, T2, T3

T1 ∈ {L2}

T2 ∈ {L3}

T3 ∈ {L1, L2}

T4 ∈ {L1}

Na osnovu sprovedenog postupka zaključujemo da će lokomotiva L1 vući T4 kompoziciju, lokomotiva L2 kompoziciju L2, lokomotiva L3 kompoziciju T2, dok će kompoziciju T3 vući ili lokomotiva L1 ili L2.

Zadatak 96: Slova kao sekvenca brojeva

Slova A, B, C, D, E, F i G, ne nužno u navedenom redosledu, zamenjuju sedam sukcesivnih celih brojeva u intervalu od 1 do 10. D je za 3 manje od A. B je u sredini. F je za toliko manje od B koliko je C veće od D. G je veće od F. Metodom zadovoljenja ograničenja odgovoriti na sledeća pitanja:

a) Koje slovo zamenjuje peti broj u sekvenci?
b) Koliko iznosi razlika A - F?
c) Broj T je veći od C za onoliko koliko je C veće od E. T se može napisati kao A+E. Koliko je D?
d) Za koliko je veća najveća moguća vrednost C od najmanje moguće vrednosti D?

Rešenje

Moguće vrednosti za svako od slova su sledeće (u pitanju je sedam sukcesivnih brojeva, ne vrednosti od 1 do 7):

$A \in \{1,2,3,4,5,6,7\}$

$B \in \{1,2,3,4,5,6,7\}$

$C \in \{1,2,3,4,5,6,7\}$

$D \in \{1,2,3,4,5,6,7\}$

$E \in \{1,2,3,4,5,6,7\}$

$F \in \{1,2,3,4,5,6,7\}$

$G \in \{1,2,3,4,5,6,7\}$

Definišemo pet pravila na osnovu postavke zadatka:

O1: $D = A - 3$

O2: B se nalazi u sredini $\Rightarrow B = 4$

O3: $B - F = C - D$

O4: $G > F$

O5: Svako slovo predstavlja jednu cifru

Aktivne: A,B,C,D,E,F,G

Neaktivne: -

1. Biramo A:	2. Biramo B	3. Biramo C
$A \in \{4,5,6,7\}$	$B = 4$	$C \neq 4, C \neq 1, C \neq 7$
Aktivne: B,C,D,E,F,G	Aktivne: C,D,E,F,G,A	Aktivne: D,E,F,G,A
Neaktivne: A	Neaktivne: B	Neaktivne: B,C
$A \in \{4,5,6,7\}$	$A \in \{4,5,6,7\}$	$A \in \{4,5,6,7\}$
$B \in \{1,2,3,4,5,6,7\}$	$B \in \{4\}$	$B \in \{4\}$

C ∈ {1,2,3,4,5,6,7}	C ∈ {1,2,3,4,5,6,7}	C ∈ {2,3,5,6}
D ∈ {1,2,3,4,5,6,7}	D ∈ {1,2,3,4,5,6,7}	D ∈ {1,2,3,4,5,6,7}
E ∈ {1,2,3,4,5,6,7}	E ∈ {1,2,3,4,5,6,7}	E ∈ {1,2,3,4,5,6,7}
F ∈ {1,2,3,4,5,6,7}	F ∈ {1,2,3,4,5,6,7}	F ∈ {1,2,3,4,5,6,7}
G ∈ {1,2,3,4,5,6,7}	G ∈ {1,2,3,4,5,6,7}	G ∈ {1,2,3,4,5,6,7}

4. Biramo D	5. Biramo E	6. Biramo F
D ≠ 4, D ∉ {5,6,7}	E ≠ 4	F ≠ 4, F ∉ {5,6,7}
Aktivne: E,F,G,A,C	Aktivne: F,G,A,C,D	Aktivne: G,A,C,D,E
Neaktivne: B	Neaktivne: B,E	Neaktivne: B,F
A ∈ {4,5,6,7}	A ∈ {4,5,6,7}	A ∈ {4, 5,6,7}
B ∈ {4}	B ∈ {4}	B ∈ {4}
C ∈ {2,3,5,6}	C ∈ {2,3,5,6}	C ∈ {2,3,5,6}
D ∈ {1,2,3}	D ∈ {1,2,3}	D ∈ {1,2,3}
E ∈ {1,2,3,4,5,6,7}	E ∈ {1,2,3,5,6,7}	E ∈ {1,2,3,5,6,7}
F ∈ {1,2,3,4,5,6,7}	F ∈ {1,2,3,4,5,6,7}	F ∈ {1,2,3}
G ∈ {1,2,3,4,5,6,7}	G ∈ {1,2,3,4,5,6,7}	G ∈ {1,2,3,5,6,7}

7. Biramo G	8. Biramo A	9. Biramo C
G ≠ 4, G ≠ 1	A ≠ 4, A ≠ 7	Aktivne: D,E,F,G,A
Aktivne: A,C,D,E,F	Aktivne: C,D,E,F,G	Neaktivne: B,C
Neaktivne: B,G	Neaktivne: B,A	A ∈ {5,6}
A ∈ {4, 5,6,7}	A ∈ {5,6}	B ∈ {4}
B ∈ {4}	B ∈ {4}	C ∈ {2,3,5,6}
C ∈ {2,3,5,6}	C ∈ {2,3,5,6}	D∈ {1,2,3}
D ∈ {1,2,3}	D∈ {1,2,3}	E ∈ {1,2,3,5,6,7}
E ∈ {1,2,3,5,6,7}	E ∈ {1,2,3,5,6,7}	F ∈ {1,2,3}
F ∈ {1,2,3}	F ∈ {1,2,3}	G ∈ {2,3,5,6,7}
G ∈ {2,3,5,6,7}	G ∈ {2,3,5,6,7}	

10. Biramo D	11. Biramo E	12. Biramo F
D ≠ 1, D ≠ 2	E ≠ 3	F ≠ 3
Aktivne: E,F,G,A,C	Aktivne: F,G,A,C	Aktivne: G,A,C,E

Neaktivne: B,D	Neaktivne: B,D,E	Neaktivne: B,D,F
A ∈ {5,6}	A ∈ {5,6}	A ∈ {5,6}
B ∈ {4}	B ∈ {4}	B ∈ {4}
C ∈ {2,3,5,6}	C ∈ {2,3,5,6}	C ∈ {2,3,5,6}
D∈ {3}	D∈ {3}	D∈ {3}
E ∈ {1,2,3,5,6,7}	E ∈ {1,2,5,6,7}	E ∈ {1,2,5,6,7}
F ∈ {1,2,3}	F ∈ {1,2,3}	F ∈ {1,2}
G ∈ {2,3,5,6,7}	G ∈ {2,3,5,6,7}	G ∈ {2,3,5,6,7}

13. Biramo G	14. Biramo A	15. Biramo C
G ≠ 3	A ≠ 5	C ≠ 6, C ≠ 3, C ≠ 2
Aktivne: A,C,E,F	Aktivne: C,E,F,G	Aktivne: E,F,G
Neaktivne: B,D,G	Neaktivne: B,D,A	Neaktivne: B,D,A,C
A ∈ {5,6}	A ∈ {6}	A ∈ {6}
B ∈ {4}	B ∈ {4}	B ∈ {4}
C ∈ {2,3,5,6}	C ∈ {2,3,5,6}	C ∈ {5}
D∈ {3}	D∈ {3}	D∈ {3}
E ∈ {1,2,5,6,7}	E ∈ {1,2,5,6,7}	E ∈ {1,2,5,6,7}
F ∈ {1,2}	F ∈ {1,2}	F ∈ {1,2}
G ∈ {2,5,6,7}	G ∈ {2,5,6,7}	G ∈ {2,5,6,7}

16. Biramo E	17. Biramo F	18. Biramo G
E ≠ 5, E ≠ 6	F ≠ 1	G ≠ 2, G ≠ 5, G ≠ 6
Aktivne: F,G	Aktivne: G,E	Aktivne: E
Neaktivne: B,D,A,C,E	Neaktivne: B,D,A,C,E	Neaktivne: B,D,A,C,F,G
A ∈ {6}	A ∈ {6}	A ∈ {6}
B ∈ {4}	B ∈ {4}	B ∈ {4}
C ∈ {5}	C ∈ {5}	C ∈ {5}
D∈ {3}	D∈ {3}	D∈ {3}
E ∈ {1,2,7}	E ∈ {1,2,7}	E ∈ {1,2,7}
F ∈ {1,2}	F ∈ {2}	F ∈ {2}
G ∈ {2,5,6,7}	G ∈ {2,5,6,7}	G ∈ {7}

19. Biramo E $E \neq 2, E \neq 7$ Aktivne: - Neaktivne: B,D,A,C,F,G,E $A \in \{6\}$ $B \in \{4\}$ $C \in \{5\}$ $D \in \{3\}$ $E \in \{1\}$ $F \in \{2\}$ $G \in \{7\}$	A = 6 B = 4 C = 5 D = 3 E = 1 F = 2 G = 7 **EFDBCAG**	

a) Na osnovu određene sekvence uočava se da je na petoj poziciji simbol C

b) Na osnovu pronađenog redosleda vrednosti jednostavno je odrediti razliku: A-F = 4

c) Na osnovu postavke zadatka možemo zapisati sledeće formule:

$T = C + (C - E) = C + 4$

$T = A + E \Rightarrow C + 4 = A + E$

Zamenom poznatih vrednosti dobija se:

$A = 4 + C - E = 4 + 4 = 8$

$D = A - 3 \Rightarrow D = 5$

d) Sprovedenim postupkom za svako slovo utvrđen je redosled u sekvenci. Na osnovu toga jednostavno je odrediti maksimalnu i minimalnu vrednost svakog broja. Razmotrimo, na primer, slovo C. Kako je određeno slovo C je peto u sekvenci. Kako je minimalna vrednost prvog slova u sekvenci 1 to se može zaključiti da je minimalna vrednost za C upravo vrednost 5. Sa druge strane maksimalna vrednost poslednjeg slova u sekvenci (slova G), kako je definisano postavkom je 10. Kako je slovo G na sedmom mestu u sekvenci jednostavno je zaključiti da je maksimalna vrednost za peto slovo u sekvenci (slovo C) jednaka osam. Sličnim razmatranjem može se odrediti minimalna vrednost čvora D. Na osnovu prethodnog razmatranja nalazimo:

$Cmax = 8$

$Dmin = 3$

$Cmax – Dmin = 5$

3.3. Metod sukcesivnih aproksimacija

Zadatak 97: Putovanje u Tivat

Tetka Marija koja živi u Tivtu pozvala je sestrića Nenada, studenta računarske tehnike iz Beograda da provede nekoliko dana na moru. Postoji niz različitih načina za putovanje tako da je Nenad odlučio da uz pomoć GPS (*General problem solver* - opšti rešavač problema) algoritma izabere odgovarajući prevoz u svakoj tački svog putovanja prema sledećim pravilima:

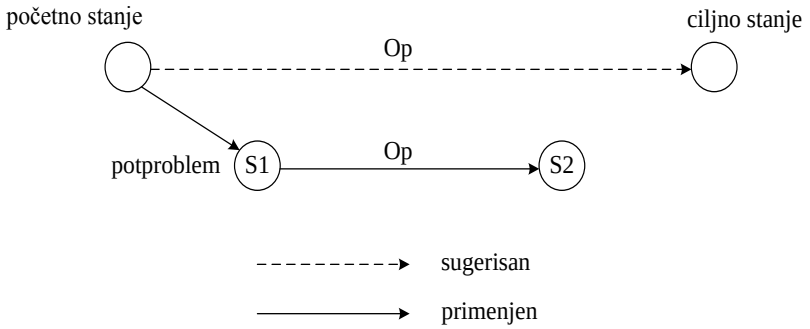
- ako je put duži od 250 km, putovati avionom ili vozom.
- ako je put duži od 50 km i kraći od 250 km, putovati vozom ili kolima.
- ako je put duži od 1 km, a kraći od 50 km, putovati kolima ili uzeti taksi.
- ako je put kraći od 1 km, ići pešice.

Preduslov za putovanje avionom je da se bude na aerodromu, za putovanje vozom da se bude na železničkoj stanici, a da se putuje kolima je da se poseduju kola.

Odrediti na koji će način Nenad, koji u Beogradu ima svoja kola, doputovati tetki Mariji.

Rešenje

Algoritam GPS naveden je u dodatku 1 (algoritam 9). GPS rešava probleme primenom strategije sukcesivnih aproksimacija (engl. *means-ends analysis*) koja se može opisati na sledeći način: na početku se identifikuju početno, ciljno stanje, i uvodi kriterijum za ocenu razlike između tekućeg i ciljnog stanja. Takođe se definišu operatori prevođenja u nova stanja i uslovi primene istih. U tekućem stanju S, na osnovu razlike tekućeg i ciljnog stanja za primenu se bira onaj operator koji najviše smanjuje tu razliku (koji prevodi u stanje najbliže ciljnom stanju). Ukoliko uslovi za primenu izabranog operatora Op u stanju S nisu zadovoljeni, na osnovu tih uslova definiše se neko stanje S_1 kao novi parcijalni cilj i rekurzivno primenjuje GPS algoritam na novi problem prelaska iz stanja S u stanje S_1. Rešavanjem ovog podproblema tekuće stanje postaje S_1 i tada je moguće primeniti operator Op koji prevodi iz stanja S_1 u neko novo stanje S_2 (slika 125). U novoj iteraciji traži se operator koji će, primenjen na stanje S_2, smanjiti razliku tekućeg i ciljnog stanja, itd. sve dok se ne postigne prvobitni cilj.



Slika 125

U našem problemu, stanje je opisano lokacijom na kojoj se Nenad u toku putovanja nalazi, početno stanje je Nenadov stan u Beogradu, ciljno stanje je tetkina kuća u Tivtu, operatori promene stanja su zadata prevozna sredstva (uključujući i hodanje) sa svojim ograničenjima, a razlika stanja u ovom slučaju predstavljena je geografskim rastojanjem tekuće lokacije na kojoj se Nenad nalazi od ciljne lokacije.

Tabela 9 definiše operatore promene stanja u skladu sa uslovima zadatka. Pretpostavljeno je da za operator *voziti se taksijem* nema posebnih preduslova jer Nenad može sa svakog mesta pozvati taksi telefonom. Tabela 10 predstavlja takozvanu tabelu razlika. Svaka vrsta tabele označena je određenom razlikom tekućeg i ciljnog stanja, dok su kolone označene operatorima promene stanja. Ulaz u vrsti koja odgovara razlici R i koloni koja odgovara operatoru Op popunjen je sa DA ako je moguće upotrebiti operator Op za smanjenje razlike R. Ukoliko je za smanjenje određene razlike moguće upotrebiti više od jednog operatora, prioritet operatora opada s leva na desno u odgovarajućoj vrsti tabele.

operator	preduslov	akcija
leteti avionom u mesto *x*	lokacija = aerodrom Surčin	lokacija = aerodrom u mestu *x*
putovati vozom u mesto *x*	lokacija = beogradska železnička stanica	lokacija = železnička stanica u mestu *x*
voziti se kolima u mesto *x*	lokacija = Nenadov parking	lokacija = *x*
voziti se taksijem u mesto *x*	-	lokacija = *x*
hodati do mesta *x*	-	lokacija = *x*

Tabela 9

razlika r	putovati avionom	putovati vozom	voziti se kolima	voziti se taksijem	hodati
r > 250 Km	DA	DA			
50 Km< r < 250 Km		DA	DA		
1 Km < r < 50 Km			DA	DA	
r < 1 Km					DA

Tabela 10

Pretraga započinje pozivom GPS procedure sa početnom lokacijom kao tekućim stanjem:

Nivo rekurzije: 1

Tekuće stanje:

lokacija = Nenadova kuća

Ciljno stanje:

lokacija = tetkina kuća u Tivtu

Razlika:

r = 300 Km

S obzirom da je rastojanje od Beograda do Tivta veće od 250 km, tabela razlika sugeriše putovanje avionom do Tivta kao prvi izbor. Preduslov za to je da se bude u avionu, što u početnom stanju nije zadovoljeno, pa ovo postaje cilj drugog poziva GPS procedure.

Nivo rekurzije: 2

Tekuće stanje:

lokacija = Nenadova kuća

Ciljno stanje:

lokacija = aerodrom Surčin

Razlika:

r = 10 Km

Rastojanje od Nenadovog stana do aerodroma je veće od 1 km a manje od 50 km, pa je vožnja kolima do aerodroma prvi izbor u tabeli razlika. Preduslov za to je da se bude u kolima, pa se ovaj problem rešava novim rekurzivnim pozivom GPS procedure.

Nivo rekurzije: 3

Tekuće stanje:

lokacija = Nenadova kuća

Ciljno stanje:

lokacija = Nenadov parking

Razlika:

r = 100 m

Razlika od 100m nalaže hodanje do kola što je neposredno primenljivo, pa tekuća lokacija za Nenada postaje parking.

Nivo rekurzije: 3

Tekuće stanje:

lokacija = Nenadov parking

Ciljno stanje:

lokacija = Nenadov parking

Razlika:

r = 0

Ustanovljava se da nema razlike između tekućeg i ciljnog stanja, pa treći nivo GPS algoritma sa uspehom završava rad i vraća kontrolu drugom nivou.

Nivo rekurzije: 2

Tekuće stanje:

lokacija = Nenadov parking

Ciljno stanje:

lokacija = aerodrom Surčin

Razlika:

r = 10 Km

Sada se može primeniti i ranije izabrani operator vožnje kolima do aerodroma da bi se eliminisala razlika između tekućeg stanja i ciljnog stanja za drugi nivo GPS-a.

Nivo rekurzije: 2

Tekuće stanje:

lokacija = aerodrom Surčin

Ciljno stanje:

lokacija = aerodrom Surčin

Razlika:

r = 0

Time je zadatak dolaska na aerodrom rešen, čime se okončava GPS procedura drugog nivoa i kontrola predaju prvom nivou.

Nivo rekurzije: 1

Tekuće stanje:

lokacija = aerodrom Surčin

Ciljno stanje:

lokacija = tetkina kuća u Tivtu

Razlika:

r = 300 km

Sada se u okviru GPS procedure prvog nivoa primenjuje operator vožnje avionom, tako da Nenad dospeva na tivatski aerodrom.

Nivo rekurzije: 1

Tekuće stanje:

lokacija = tivatski aerodrom

Ciljno stanje:

lokacija = tetkina kuća u Tivtu

Razlika:

r = 7 km

Rastojanje do tetkine kuće je veće od 1 km, a manje od 50 km. Na osnovu zadatih pravila, vožnja kolima je prvi izbor. Međutim, preduslov *lokacija = Nenadov parking* ne može se zadovoljiti jer ne postoji operator koji bi Nenada vratio u Beograd. Na taj način ostaje vožnja taksijem do tetkine kuće kao alternativa koja se odmah može primeniti.

Nivo rekurzije: 1

Tekuće stanje:

lokacija = tetkina kuća u Tivtu

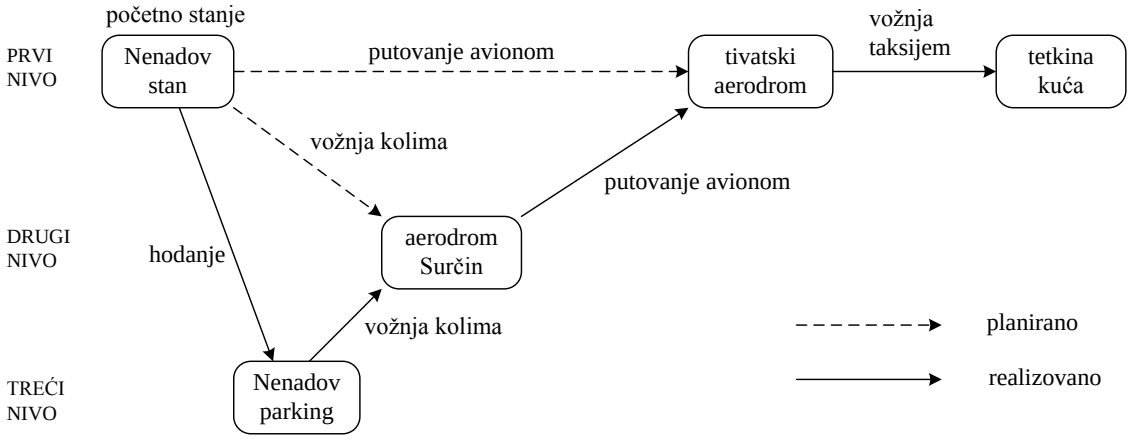
Ciljno stanje:

lokacija = tetkina kuća u Tivtu

Razlika:

r = 0

GPS procedura prvog nivoa ustanovljava da nema razlike između tekućeg i ciljnog stanja, čime je zadatak rešen. Slika 126 grafički prikazuje postupak rešavanja ovog problema.



Slika 126

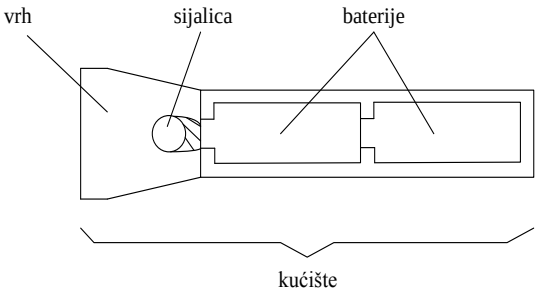
U ovom problemu rešenje je nađeno strategijom ulančavanja unapred (engl. *forward chaining*) to jest,. pri rešavanju se polazilo od startnog stanja da bi se dospelo u ciljno. GPS algoritam može se preformulisati i za slučaj povratnog ulančavanja (engl. *backward chaining*) kada se pri rešavanju kreće od ciljnog stanja, da bi se pronašao put do startnog rešavanjem niza podciljeva. Za neke probleme povratno ulančavanje predstavlja pogodniji način rešavanja.

Zadatak 98: Popravak baterijske lampe

Razmotrimo popravku baterijske lampe prikazane na slici 127.

Lampa ima dve baterije u kućištu koje dodiruju sijalicu. Da bi se došlo do baterija, mora se rastaviti kućište, a da bi se došlo do sijalice mora se rastaviti vrh baterijske lampe.

Primenom sukcesivnih aproksimacija rešiti problem otklanjanja kvara zbog pregorele sijalice.



Slika 127

Rešenje

Pojedina stanja problema opisivaćemo sledećim predikatima (značenja pojedinih predikata odgovaraju njihovim nazivima):

istrošene(baterije), ok(baterije), u_kućištu(baterije), van_kućišta(baterije), pregorela(sijalica), ok(sijalica), rastavljeno(kućište), sastavljeno(kućište), ispravno(kućište), polomljeno(kućište), rastavljen(vrh) i sastavljen(vrh).

Predikat polomljeno(kućište) predstavlja negaciju predikata ispravno(kućište) i predviđen je za situaciju kada popravljač, u nedostatku strpljenja, primeni grublji metod za rastavljanje kućišta.

Početno stanje u slučaju pod a) opisano je sa:

sastavljeno(kućište), sastavljen(vrh), u_kućištu(baterije), pregorela(sijalica), ok(baterije), ispravno(kućište)

Ciljno stanje se može parcijalno opisati navodeći samo predikate koji moraju da budu zadovoljeni na kraju pretrage:

ok(sijalica), sastavljeno(kućište), sastavljen(vrh)

Predikati sastavljeno(kućište) i sastavljen(vrh) su dodati u ciljno stanje da lampa ne bi ostala rastavljena. U opštem slučaju parcijalan opis ciljnog stanja omogućava da se pretraga završi u jednom od većeg broja različitih ciljnih stanja (kompletan opis bi eliminisao sva ciljna stanja osim jednog određenog).

Za promenu stanja koristićemo operatore definisane u tabeli 11. U preduslovima za primenu operatora navedeni su stavovi koji se moraju nalaziti u tekućem stanju da bi operator mogao da bude primenjen na to stanje. Primenom operatora neki stavovi se uklanjaju iz tekućeg stanja, a neki drugi se dodaju i time se dobija novo stanje koje postaje tekuće.

Razliku između tekućeg i ciljnog stanja definišemo kao listu svih predikata koji su prisutni u ciljnom stanju, a nisu prisutni u tekućem, na primer, za napred definisano početno i ciljno stanje razlika je samo u predikatu ok(sijalica).

Operator	Preduslov	Akcija	
		Ukloniti predikate	Dodati predikate
zameniti baterije	rastavljeno(kućište), van_kućišta(baterije), ispravno(kućište)	van_kućišta(baterije), istrošene(baterije)	u_kućištu(baterije), ok(baterije)
zameniti sijalicu	rastavljen(vrh)	pregorela(sijalica)	ok(sijalica)
rastaviti kućište	sastavljeno(kućište)	sastavljeno(kućište)	rastavljeno(kućište)
rastaviti vrh	rastavljeno(kućište), sastavljen(vrh)	sastavljen(vrh)	rastavljen(vrh)
sastaviti kućište	rastavljeno(kućište), sastavljen(vrh), ispravno(kućište)	rastavljeno(kućište)	sastavljeno(kućište)
sastaviti vrh	rastavljen(vrh)	rastavljen(vrh)	sastavljen(vrh)
istresti baterije	rastavljeno(kućište)	u_kućištu(baterije)	van_kućišta(baterije)
razbiti kućište	-	ispravno(kućište), sastavljeno(kućište), sastavljen(vrh), u_kućištu(baterije)	polomljeno(kućište), rastavljeno(kućište), rastavljen(vrh), van_kućišta(baterije)

Tabela 11

	zameniti baterije	zameniti sijalicu	rastaviti kućište	rastaviti vrh	sastaviti kućište	sastaviti vrh	istresti baterije	razbiti kućište
ok(baterije)	DA							
ok(sijalica)		DA						
rastavljeno(kućište)			DA					DA
rastavljen(vrh)				DA				DA
sastavljeno(kućište)					DA			
sastavljen(vrh)						DA		
van_kućišta(baterije)							DA	DA

Tabela 12

Za rad algoritma potrebna je tabela razlika koja za određenu razliku između tekućeg i ciljnog stanja sugeriše koji operator treba primeniti da bi se razlika smanjila. U ovom slučaju koristićemo tabelu 12 kod koje su vrste obeležene razlikama koje želimo da eliminišemo, a kolone operatorima. Tabela se čita odgore nadole i sleva udesno. Drugim rečima, ako se stanja razlikuju po većem broju predikata, prvo se zadovoljava onaj koji je naveden u prvoj vrsti tabele, pa zatim onaj iz druge vrste i tako dalje. Ako se za određeni predikat sugeriše veći broj operatora, najveći prioritet ima onaj iz prve kolone, pa zatim onaj iz druge i tako dalje.

Za rešavanje problema koristićemo GPS algoritam 9 iz dodatka 1.

1. Inicijalno je tekuće stanje algoritma jednako početnom stanju S_0, a ciljno stanje predstavlja parcijalno opisan krajnji cilj pretrage C_0.

Nivo rekurzije: 1
Tekuće stanje S_0:
sastavljeno(kućište), sastavljen(vrh), u_kućištu(baterije), pregorela(sijalica), ok(baterije), ispravno(kućište)
Ciljno stanje C_0:
ok(sijalica), sastavljeno(kućište), sastavljen(vrh)
Razlika:
ok(sijalica).

Tabela razlika diktira upotrebu operatora zameniti_sijalicu. Preduslov za ovaj operator, predikat rastavljen(vrh), nije ispunjen u tekućem stanju. Zato rekurzivno primenjujemo GPS algoritam na stanje S_0 kao tekuće i novo ciljno stanje C_1 opisano samo operatorom rastavljen(vrh).

Nivo rekurzije 2:
Tekuće stanje S_0:
sastavljeno(kućište), sastavljen(vrh), u_kućištu(baterije), pregorela(sijalica), ok(baterije), ispravno(kućište)
Ciljno stanje C_1:
rastavljen(vrh)
Razlika:
rastavljen(vrh).

2. Drugi rekurzivni nivo GPS algoritma ustanovljava predikat rastavljen(vrh) kao razliku tekućeg i ciljnog stanja. Konsultacijom tabele razlika ustanovljava se da u obzir dolaze dva operatora: rastaviti_vrh i razbiti_kućište, od kojih prema usvojenim pravilima biramo operator rastaviti_vrh. Preduslov ovog operatora, rastavljeno(kućište) i sastavljen(vrh) nije potpuno zadovoljen u tekućem stanju S_0 pa se od preduslova formira novo ciljno stanje C_2 i poziva treći rekurzivni nivo GPS algoritma.

Nivo rekurzije: 3

Tekuće stanje S_0:

sastavljeno(kućište), sastavljen(vrh), u_kućištu(baterije), pregorela(sijalica), ok(baterije), ispravno(kućište)

Ciljno stanje C_2:

rastavljeno(kućište), sastavljen(vrh)

Razlika:

rastavljeno(kućište).

3. Treći nivo ustanovljava razliku stanja S_0 i novog ciljnog stanja C_2; radi se o predikatu rastavljeno(kućište). Prema tabeli razlika bira se operator rastaviti_kućište. Preduslov ovog operatora, predikat sastavljeno(kućište) zadovoljen je u tekućem stanju S_0 pa se primenjuje na ovo stanje. Prema tabeli operatora, iz liste predikata stanja S_0 briše se sastavljeno(kućište) a dodaje se rastavljeno(kućište) čime se dobija novo tekuće stanje S_1.

Nivo rekurzije: 3

Tekuće stanje S_1:

rastavljeno(kućište), sastavljen(vrh), u_kućištu(baterije), pregorela(sijalica), ok(baterije), ispravno(kućište)

Ciljno stanje C_2:

rastavljeno(kućište), sastavljen(vrh)

Razlika:

-

Treći nivo GPS algoritma ustanovljava da nema razlike između stanja S_1 i njegovog ciljnog stanja C_2 pa sa uspehom završava rad čime se kontrola vraća drugom nivou GPSa.

Nivo rekurzije: 2

Tekuće stanje S_1:

rastavljeno(kućište), sastavljen(vrh), u_kućištu(baterije), pregorela(sijalica), ok(baterije), ispravno(kućište)

Ciljno stanje C_1:

rastavljen(vrh)

Razlika:

rastavljen(vrh).

4. Drugi nivo GPS algoritma primenjuje ranije izabrani operator rastaviti_vrh jer su sada njegovi preduslovi zadovoljeni. Dobija se novo tekuće stanje S_2.

Nivo rekurzije: 2

Tekuće stanje S_2:

rastavljeno(kućište), rastavljen(vrh), u_kućištu(baterije), pregorela(sijalica), ok(baterije), ispravno(kućište)

Ciljno stanje C_1:

rastavljen(vrh)

Razlika:

-

Pošto razlike nema, kontrola se vraća prvom nivou GPS algoritma.

Nivo rekurzije: 1

Tekuće stanje S_2:

rastavljeno(kućište), rastavljen(vrh), u_kućištu(baterije), pregorela(sijalica), ok(baterije), ispravno(kućište)

Ciljno stanje C_0:

ok(sijalica), sastavljeno(kućište), sastavljen(vrh)

Razlika:

ok(sijalica), sastavljeno(kućište), sastavljen(vrh)

5. Prvi nivo GPS algoritma primenjuje izabrani operator zameniti_sijalicu (čiji su preduslovi sada ispunjeni) te se dobija novo tekuće stanje S_3.

Nivo rekurzije: 1

Tekuće stanje S_3:

rastavljeno(kućište), rastavljen(vrh), u_kućištu(baterije), ok(sijalica), ok(baterije), ispravno(kućište)

Ciljno stanje C_0:

ok(sijalica), sastavljeno(kućište), sastavljen(vrh)

Razlika:

sastavljeno(kućište), sastavljen(vrh)

Krajnje ciljno stanje razlikuje se od ovog stanja po predikatima sastavljeno(kućište) i sastavljen(vrh). Konsultuje se tabela razlika za predikat sastavljeno(kućište) pošto je ovaj predikat naveden u tabeli pre predikata sastavljen(vrh). Izbor pada na operator sastaviti_kućište. Preduslov: rastavljeno(kućište), sastavljen(vrh), ispravno(kućište) nije u potpunosti zadovoljen, pa se od ovog preduslova formira novo ciljno stanje C_3 i kontrola rekurzivnim pozivom prenosi na drugi nivo GPS algoritma.

Nivo rekurzije: 2
Tekuće stanje S_3:
rastavljeno(kućište), rastavljen(vrh), u_kućištu(baterije), ok(sijalica), ok(baterije), ispravno(kućište)
Ciljno stanje C_3:
rastavljeno(kućište), sastavljen(vrh), ispravno(kućište)
Razlika:
sastavljen(vrh)

6. Drugi nivo ustanovljava razliku: sastavljen(vrh). Bira se operator sastaviti_vrh, i s obzirom da je preduslov rastavljen(vrh) zadovoljen u tekućem stanju S_3, primenom ovog operatora dobija se novo tekuće stanje S_4.

Nivo rekurzije: 2
Tekuće stanje S_4:
rastavljeno(kućište), sastavljen(vrh), u_kućištu(baterije), ok(sijalica), ok(baterije), ispravno(kućište)
Ciljno stanje C_3:
rastavljeno(kućište), sastavljen(vrh), ispravno(kućište)
Razlika:
-

Drugi nivo završava rad s obzirom da se tekuće stanje ne razlikuje od njegovog ciljnog stanja, pa se kontrola vraća na prvi nivo.

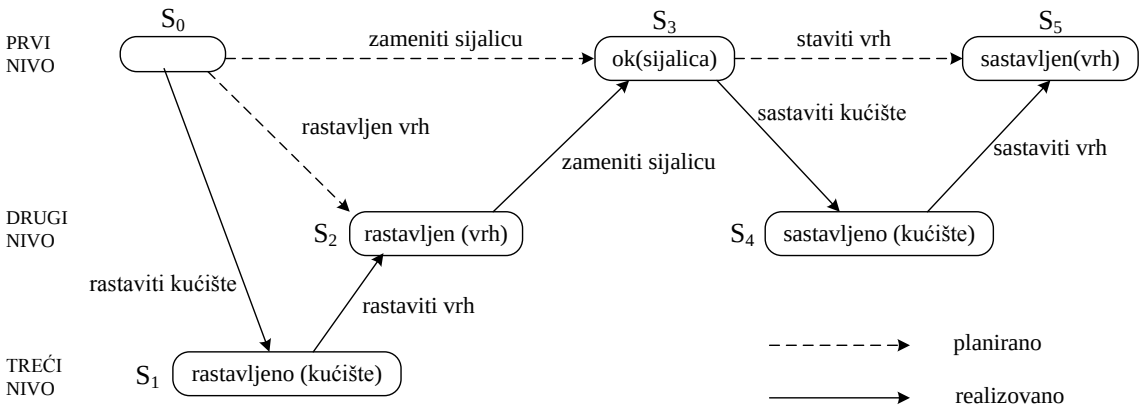
Nivo rekurzije: 1
Tekuće stanje S_4:
rastavljeno(kućište), sastavljen(vrh), u_kućištu(baterije), ok(sijalica), ok(baterije), ispravno(kućište)
Ciljno stanje C_0:
ok(sijalica), sastavljeno(kućište), sastavljen(vrh)
Razlika:
sastavljeno(kućište)

7. Prvi nivo algoritma primenjuje izabrani operator sastaviti_kućište čime se dobija novo tekuće stanje S_5.

Nivo rekurzije: 1
Tekuće stanje S_5:
sastavljeno(kućište), sastavljen(vrh), u_kućištu(baterije), ok(sijalica), ok(baterije), ispravno(kućište)

Ciljno stanje C_0:
ok(sijalica), sastavljeno(kućište), sastavljen(vrh)
Razlika:
-

Sada se ustanovljava da nema razlike između tekućeg stanja i krajnjeg ciljnog stanja i prvi nivo algoritma GPS okončava rad. Slika 128 rezimira rad GPS algoritma za zadati problem. Za pojedina stanja S_i, osim stanja S_0 navedeni su samo predikati koji ih razlikuju od stanja-prethodnika tokom pretrage, a koji ujedno predstavljaju ciljeve.



Slika 128

Zadatak 99: Problem majmuna i banane

Majmun stoji kod vrata sobe, na čijoj sredini visi banana okačena o plafon. Uz prozor stoji sanduk. Majmun ne može sam da dohvati bananu, ali može ako stoji na sanduku. Takođe, majmun može da se šeta po sobi, da odgura sanduk ili da se popne na njega. Metodom sukcesivnih aproksimacija odrediti na koji će način majmun doći do banana.

Rešenje

Stanja problema opisana su vrednostima sledeće četiri promenljive:

- *sanduk* daje informaciju o poziciji sanduka, i može da ima jednu od sledećih vrednosti: kod_prozora, kod_vrata, na_sredini_sobe
- *majmun_H* daje informaciju o poziciji majmuna u horizontalnoj ravni i može da ima isti skup vrednosti kao i promenljiva *sanduk*
- *majmun_V* daje informaciju o poziciji majmuna po vertikali i može da ima sledeće vrednosti: na_podu, na_sanduku
- *ima_bananu* je ispunjeno (TRUE) ako majmun poseduje bananu, inače je netačno (FALSE).

Operatori promene stanja definisani su tabelom 13. Operatori ŠETATI i POMERITI SANDUK poseduju kao argument promenljivu *nova_poz* koja označava ciljnu poziciju kretanja majmuna.

Razlike između stanja u ovom problemu definišemo kao razlike u vrednostima promenljivih sa istim imenom u tekućem i ciljnom stanju. U tabeli 14 svaka vrsta odgovara određenoj razlici. Ukoliko u vrsti koja odgovara razlici R i koloni obeleženoj operatorom Op stoji DA, znači da se operator Op može koristiti za eliminaciju razlike R. U toku rada algoritma prilikom konsultovanja tabele razlika koristi se konvencija da prioritet razlika opada odozgo nadole a prioritet operatora sleva-nadesno. U trećoj i četvrtoj vrsti tabele, *pozicija* označava vrednost promenljive majmun_H odnosno sanduk u ciljnom stanju; prilikom izbora nekog od operatora ŠETATI i POMERITI SANDUK, promenljivoj *nova_poz* dodeljuje se *pozicija*.

Operator	Preduslov	Akcija
POPETI SE NA SANDUK	majmun_H = sanduk majmun_V = na_podu	majmun_V := na_sanduku
ŠETATI(*nova_poz*)	-	majmun_H := *nova_poz*
POMERITI SANDUK(*nova_poz*)	majmun_H = sanduk majmun_V = na_podu	majmun_H:= *nova_poz* sanduk := *nova_poz*
UZETI_BANANU	ima_bananu = FALSE majmun_V = na_sanduku sanduk = na_sredini_sobe	ima_bananu = TRUE

Tabela 13

	UZETI BANANU	POPETI SE NA SANDUK	ŠETATI	POMERITI SANDUK
ima_bananu=TRUE	DA			
majmun_V=na_sanduku		DA		
majmun_H = *pozicija*			DA	DA
sanduk = *pozicija*				DA

Tabela 14

Pretraga započinje pozivom GPS procedure sa tekućim stanjem inicijalno jednakim početnom stanju S0, a ciljno stanje jednako je parcijalnom opisu C0 konačnog cilja.

Nivo rekurzije: 1

Tekuće stanje S_0:

sanduk = kod_prozora, majmun_H = kod_vrata,

majmun_V = na_podu, ima_bananu = FALSE

Ciljno stanje C_0:
ima_bananu = TRUE
Razlika:
ima_bananu = TRUE

Redosled događaja pri pretrazi je sledeći:

1. GPS algoritam na prvom nivou ustanovljava razliku početnog i ciljnog stanja u vrednosti promenljive ima_bananu i na osnovu tabele razlika bira operator uzeti_bananu. Od preduslova se formira novo ciljno stanje C_1:

ima_bananu = FALSE, majmun_V = na_sanduku, sanduk = na_sredini_sobe

i aktivira drugi rekurzivni nivo GPS algoritma.

Nivo rekurzije: 2
Tekuće stanje S_0:
sanduk = kod_prozora, majmun_H = kod_vrata,
majmun_V = na_podu, ima_bananu = FALSE
Ciljno stanje C_1:
ima_bananu = FALSE, majmun_V = na_sanduku, sanduk = na_sredini_sobe
Razlika:
majmun_V = na_sanduku, sanduk = na_sredini_sobe

2. Utvrđuje se nova razlika: majmun_V = na_sanduku, sanduk = na_sredini_sobe.

Prema usvojenoj konvenciji o prioritetima iz tabele razlika se bira operator koji zadovoljava cilj majmun_V = na_sanduku, a to je POPETI SE NA SANDUK. Preduslovi:

majmun_H = kod_prozora (vrednost promenljive sanduk iz tekućeg stanja),

majmun_V = na_podu

nisu potpuno zadovoljeni pa se za novo ciljno stanje C_2 opisano navedenim preduslovima aktivira treći rekurzivni nivo GPS algoritma.

Nivo rekurzije: 3
Tekuće stanje S_0:
sanduk = kod_prozora, majmun_H = kod_vrata,
majmun_V = na_podu, ima_bananu = FALSE
Ciljno stanje C_2:
majmun_H = kod_prozora, majmun_V = na_podu
Razlika:
majmun_H = kod_prozora

3. Nova razlika je: majmun_H = kod_prozora. Prema tabeli razlika bira se operator ŠETATI(kod_prozora). Pošto ovaj operator nema preduslova, moguće ga je odmah primeniti čime se dobija stanje S_1.

Nivo rekurzije: 3
Tekuće stanje S_1:
sanduk = kod_prozora, majmun_H = kod_prozora,
majmun_V = na_podu, ima_bananu = FALSE
Ciljno stanje C_2:
majmun_H = kod_prozora, majmun_V = na_podu
Razlika:
-

Treći nivo GPSa ustanovljava da više nema razlike između tekućeg stanja i njemu zadatog cilja pa kontrolu vraća drugom nivou.

Nivo rekurzije: 2
Tekuće stanje S_1:
sanduk = kod_prozora, majmun_H = kod_prozora,
majmun_V = na_podu, ima_bananu = FALSE
Ciljno stanje C_1:
ima_bananu = FALSE, majmun_V = na_sanduku, sanduk = na_sredini_sobe
Razlika:
majmun_V = na_sanduku, sanduk = na_sredini_sobe

4.Drugi nivo GPSa primenjuje ranije izabrani operator POPETI SE NA SANDUK čime se dobija novo tekuće stanje S_2.

Nivo rekurzije: 2
Tekuće stanje S_2:
sanduk = kod_prozora, majmun_H = kod_prozora,
majmun_V = na_sanduku, ima_bananu = FALSE
Ciljno stanje C_1:
ima_bananu = FALSE, majmun_V = na_sanduku, sanduk = na_sredini_sobe
Razlika:
sanduk = na_sredini_sobe

Ovo stanje ide na stek tekućih stanja lokalan za drugi nivo GPSa na kome se inicijalno nalazilo početno stanje. Zatim se računa razlika između cilja C_1 i tekućeg stanja S_2 a to je:

sanduk = na_sredini_sobe

Bira se operator POMERITI SANDUK(na_sredini_sobe). Preduslovi:

majmun_H = kod_prozora, majmun_V = na_podu

nisu u potpunosti zadovoljeni pa se na osnovu njih definiše novo ciljno stanje C_2 i rekurzivno poziva treći nivo GPSa.

Nivo rekurzije: 3
Tekuće stanje S_2:
sanduk = kod_prozora, majmun_H = kod_prozora,
majmun_V = na_sanduku, ima_bananu = FALSE
Ciljno stanje C_2:
majmun_H = kod_prozora, majmun_V = na_podu
Razlika:
majmun_V = na_podu

5. Treći nivo ustanovljava razliku stanja C_2 i S_2: majmun_V = na_podu.

Konsultacijom tabele razlika utvrđuje se da se ova razlika ne može zadovoljiti. Pošto nema načina da iz početnog stanja S_2 dođe do ciljnog stanja C_2, izvršavanje trećeg nivoa GPS algoritma se neuspešno završava, i kontrola vraća drugom nivou.

Nivo rekurzije: 2
Tekuće stanje S_2:
sanduk = kod_prozora, majmun_H = kod_prozora,
majmun_V = na_sanduku, ima_bananu = FALSE
Ciljno stanje C_1:
ima_bananu = FALSE, majmun_V = na_sanduku, sanduk = na_sredini_sobe
Razlika:
sanduk = na_sredini_sobe

6. Kao posledica neuspeha, na drugom nivou GPSa ponovo se razmatra razlika tekućeg stanja S_2 i njegovog cilja C_1 a to je sanduk = na_sredini_sobe da bi se pronašao alternativni operator koji će zadovoljiti cilj. Pošto, pored već probanog operatora POMERITI SANDUK drugi operator ne postoji, ustanovljava se da se dati cilj ne može zadovoljiti i preduzima vraćanje unatrag (engl. *backtracking*): sa lokalnog steka se skida vršni element (stanje S_2) čime stanje S_1 postaje ponovo tekuće stanje.

Nivo rekurzije: 2
Tekuće stanje S_1:
sanduk = kod_prozora, majmun_H = kod_prozora,
majmun_V = na_podu, ima_bananu = FALSE

Ciljno stanje C_1:

ima_bananu = FALSE, majmun_V = na_sanduku, sanduk = na_sredini_sobe

Razlika:

majmun_V = na_sanduku, sanduk = na_sredini_sobe

Ponovo se razmatra razlika između cilja C_1 i stanja S_1:

majmun_V = na_sanduku, sanduk = na_sredini_sobe

Prioritetnija razlika je majmun_V = na_sanduku. Prethodno izabrani operator (u koraku 2) POPETI SE NA SANDUK nije dao dobar rezultat, pa se razmatra alternativa. Pošto ne postoji alternativa koja bi smanjila ovu razliku, prelazi se na razmatranje manje prioritetne razlike: sanduk = na_sredini_sobe. Ova razlika može biti eliminisana operatorom POMERITI SANDUK(na_sredinu_sobe). Treba primetiti da je ovaj operator već bio razmatran, ali ne ovom nivou GPS algoritma i ne za trenutno stanje, tako da se u ovom trenutku on može odabrati. Preduslovi:

majmun_H = kod_prozora (vrednost promenljive sanduk iz tekućeg stanja),

majmun_V = na_podu

su zadovoljeni u tekućem stanju S_1, pa se primenjuje operator POMERITI SANDUK(na_sredini_sobe) čime se dobija novo tekuće stanje S_2'.

Nivo rekurzije: 2

Tekuće stanje S_2':

sanduk = na_sredini_sobe, majmun_H = na_sredini_sobe,

majmun_V = na_podu, ima_bananu = FALSE

Ciljno stanje C_1:

ima_bananu = FALSE, majmun_V = na_sanduku, sanduk = na_sredini_sobe

Razlika:

majmun_V = na_sanduku

Utvrđuje se razlika između stanja S_2' i cilja C_1 a to je:

majmun_V = na_sanduku

Radi eliminacije razlike bira se operator POPETI SE NA SANDUK. Preduslovi ovog operatora zadovoljeni su u tekućem stanju S_2', pa se njegovom primenom dobija novo tekuće stanje S_3.

Nivo rekurzije: 2

Tekuće stanje S_3:

sanduk = na_sredini_sobe, majmun_H = na_sredini_sobe,

majmun_V = na_sanduku, ima_bananu = FALSE

Ciljno stanje C_1:

ima_bananu = FALSE, majmun_V = na_sanduku, sanduk = na_sredini_sobe

Razlika:

-

Na drugom nivou utvrđuje se da je zadovoljen cilj C_1 pa se kontrola vraća prvom nivou GPSa.

Nivo rekurzije: 1

Tekuće stanje S_3:

sanduk = na_sredini_sobe, majmun_H = na_sredini_sobe,

majmun_V = na_sanduku, ima_bananu = FALSE

Ciljno stanje C_0:

ima_bananu = TRUE

Razlika:

ima_bananu = TRUE

7. Prvi nivo GPSa primenjuje izabrani (u koraku 1) operator UZETI_BANANU čime se dobija stanje S_4.

sanduk = na_sredini_sobe, majmun_H = na_sredini_sobe,

majmun_V = na_podu, ima_bananu = TRUE

Nivo rekurzije: 1

Tekuće stanje S_4:

sanduk = na_sredini_sobe, majmun_H = na_sredini_sobe,

majmun_V = na_podu, ima_bananu = TRUE

Ciljno stanje C_0:

ima_bananu = TRUE

Razlika:

-

Utvrđuje se da je zadovoljen cilj C_0 čime algoritam uspešno završava rad. Rešenje je, prema tome, opisano sekvencom operatora:

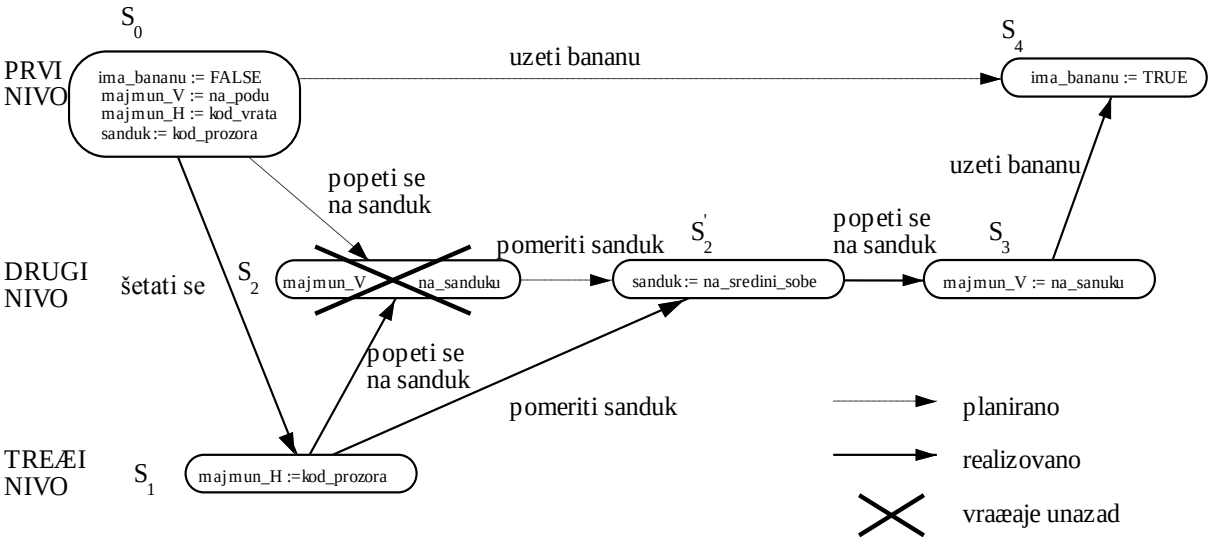
ŠETATI(kod_prozora), POMERITI_SANDUK(na_sredini_sobe), UZETI_BANANU.

Slika 129 rezimira prethodno razmatranje.

Usvojeni prioritet razmatranja razlika i prioritet primene operatora imaju znatan uticaj na broj vraćanja unazad pri radu GPS algoritma. Čitaocu se preporučuje da sam ponovi prethodni zadatak u slučaju da se druga vrsta tabele razlika premesti na kraj tabele, tako da majmun_V = na_sanduku ima najniži prioritet; u ovom slučaju rešenje se pronalazi bez vraćanja unatrag.

Kod definisanja prioriteta razlika radi se zapravo o apstrakciji problema - prioritet treba da dobiju glavni koraci u rešavanju, a manje važni da se rešavaju na višim rekurzivnim nivoima

algoritma. Pažljivo odabrani prioriteti znatno smanjuju količinu vraćanja unazad u odnosu na druge algoritme pretrage.



Slika 129

4. Rad u neizvesnom okruženju

4.1. Rezonovanje na osnovu faktora izvesnosti

Zadatak 100: Popravak automobila

a) Pregledom automobila ustanovljeno je sledeće:

1. problem zahteva hitnu popravku (izvesnost 0.8)
2. kvar je na električnoj instalaciji (0.6)
3. postoji kratak spoj na instalaciji (0.4)
4. kvar je u računaru za kontrolu ubrizgavanja (0.2)

Odrediti faktor izvesnosti zaključka: kvar je u električnoj instalaciji i potrebno ga je hitno popraviti i problem je kratak spoj ili kvar računara.

b) Poznate su vrednosti stepena izvesnosti

$CF(c,e_1) = cf_1$

$CF(c,e_2) = cf_2$

$CF(c,e_3) = cf_3$

Izračunati vrednost CF(*c*,*e*) gde *e* predstavlja sve činjenice (dokaze, događaje) povezane sa zaključkom *c*.

Rešenje

U situacijama kada ekspertski sistem radi u *neizvesnom okruženju*, ne zadovoljava karakterisanje činjenica i zaključaka samo kao tačnih ili netačnih. Jedno od rešenja ovoga problema je karakterisanje svake činjenice i zaključka *faktorom izvesnosti* (engl. *certainity factor, CF*) koji predstavlja racionalan broj u intervalu

$-1 \le CF \le 1.$

Vrednost -1 za neku činjenicu predstavlja potpuno odsustvo poverenja u datu činjenicu, dok vrednost 1 predstavlja potpuno poverenje u nju. Ostale vrednosti predstavljaju veći ili manji stepen (ne)poverenja u datu činjenicu, npr. CF = 0 označava da se data činjenica ne može ni potvrditi niti opovrgnuti. Faktor izvesnosti se izražava kao razlika *mere poverenja* (engl. *measure of belief, MB*) i *mere nepoverenja* (engl. *measure of disbelief, MD*):

$CF(c) = MB(c) - MD(c)$

gde su MB i MD brojevi u intervalu od 0 do 1. Na meru poverenja utiču sve činjenice koje mogu potvrditi dati stav *z*, a na meru nepoverenja sve činjenice koje opovrgavaju dati stav *z*.

a) Obeležimo pretpostavke sa e_1 do e_4 onim redosledom kojim su zadate u postavci zadatka. Zaključak *z* možemo predstaviti izrazom:

$c = e_2 \wedge e_1 \wedge (e_3 \vee e_4)$

Da bismo našli faktor izvesnosti CF zaključka moramo odrediti meru poverenja MB i meru nepoverenja MD u zaključak:

$CF(c) = MB(c) - MD(c)$

Pri tome koristimo sledeće formule za konjukciju pretpostavki:

$MB(e_1 \wedge e_2) = \min (MB(e_1), MB(e_2))$

$MD(e_1 \wedge e_2) = \max (MD(e_1), MD(e_2))$

Odgovarajuće formule za disjunkciju pretpostavki su:

$MB(e_1 \vee e_2) = \max (MB(e_1), MB(e_2))$

$MD(e_1 \vee e_2) = \min (MD(e_1), MD(e_2))$

Na osnovu ovih formula računamo mere poverenja MB i nepoverenja MD u zaključak:

$MB(e_2 \wedge e_1 \wedge (e_3 \vee e_4)) = \min(MB(e_2), MB(e_1), \max[MB(e_3), MB(e_4)]) =$

$= \min(0.6, 0.8, \max[0.4, 0.2])$

$= 0.4$

$MD(e_2 \wedge e_1 \wedge (e_3 \vee e_4)) = 0$ jer su mere nepoverenja u pretpostavke jednake 0.

Faktor izvesnosti zaključka je:

$$CF(c) = MB(c) - MD(c) = MB(c) = 0.4.$$

b) U datom slučaju svaka od činjenica e_1, e_2, e_3 vodi do istog zaključka z sa određenom izvesnošću. Ove faktore izvesnosti razložićemo na mere poverenja $MD(c, e_i)$ i mere nepoverenja $MB(c, e_i)$ na sledeći način:

Ako je $cf_i > 0$ onda je $MB(c, e_i) = cf_i$, $MD(z, e_i) = 0$

inače je $MD(c, e_i) = | cf_i |$, $MB(c, e_i) = 0$.

Zbirni faktor izvesnosti zaključka na osnovu svih pretpostavki možemo izračunati pod uslovom *nezavisnosti pretpostavki* e_1, e_2, e_3, kao razliku zbirnih mera poverenja i nepoverenja:

$CF(c, e) = MB_{cum}(c, e) - MD_{cum}(c, e)$

Generalno, zbirne mere poverenja i nepoverenja zaključka c na osnovu dve nezavisne pretpostavke e_1 i e_2 računaju se po sledećim formulama:

$MB_{cum}(c, e_{1,2}) = 0$ ako je $MD_{cum}(c, e_{1,2}) = 1$

$MB_{cum}(c, e_{1,2}) = MB(c, e_1) + MB(c, e_2) - MB(z, e_1) * MB(c, e_2)$

$MD_{cum}(c, e_{1,2}) = 0$ ako je $MB_{cum}(c, e_{1,2}) = 1$

$MD_{cum}(c, e_{1,2}) = MD(c, e_1) + MD(c, e_2) - MD(c, e_1) * MD(c, e_2)$

U našem slučaju, na osnovu gornjih formula dobijamo zbirnu meru poverenja $MB_{cum}(c, e_{1,2})$ i nepoverenja $MB_{cum}(c, e_{1,2})$ u prve dve pretpostavke. Da bi se uračunala i treća pretpostavka, potrebno je još jedanput primeniti gornje formule da bi se dobila konačna zbirna mera poverenja $MB_{cum}(c, e_{1,2,3})$ i nepoverenja $MD_{cum}(c, e_{1,2 ,3})$. U formulama kombinujemo zbirnu meru poverenja, odnosno nepoverenja, u prve dve pretpostavke sa merom poverenja, odnosno nepoverenja u treću pretpostavku.

$MB_{cum}(c, e_{1,2,3}) = 0$ ako je $MD_{cum}(c, e_{1,2,3}) = 1$

$MB_{cum}(c, e_{1,2,3}) = MB_{cum}(c, e_{1,2}) + MB(c, e_3) - MB_{cum}(c, e_{1,2}) * MB(c, e_3)$

$MD_{cum}(c, e_{1,2,3}) = 0$ ako je $MB_{cum}(c, e_{1,2,3}) = 1$

$$MD_{cum}(c, e_{1,2,3}) = MD_{cum}(c, e_{1,2}) + MD(c, e_3) - MD_{cum}(c, e_{1,2}) * MD(c, e_3)$$

Faktor izvesnosti zaključka se dobija kao razlika zbirnih mera poverenja i nepoverenja u sve tri pretpostavke:

$$CF(c, e_{1,2,3}) = MB_{cum}(c, e_{1,2,3}) - MD_{cum}(c, e_{1,2,3})$$

Napomena: konačni rezultat ne zavisi od redosleda kojim kombinujemo pretpostavke pri računanju kumulativnih mera poverenja, odnosno nepoverenja.

Pri računanju izvesnosti koristimo sledeće konvencije u obeležavanju: Izvesnost nekog zaključka *c* obeležavamo sa CF(*c*) pri čemu ne navodimo eksplicitno pretpostavke koje su dovele do tog zaključka. Ako želimo da naglasimo da pretpostavka *e* dovodi do zaključka *c*, tada izvesnost zaključka *c* obeležavamo sa CF(*c*, *e*). Isto važi i za meru poverenja, odnosno nepoverenja u taj zaključak.

Zadatak 101: Dijagnostika računarskog sistema

U dijagnostici računarskog sistema, na osnovu kvara izvučeni su sledeći zaključci sa odgovarajućim faktorima izvesnosti:

1. postoji problem u operativnoj memoriji (0.6)
2. problem je u centralnom procesoru (0.2)
3. problem je u memorijskom elementu (0.1)
4. problem je u kontroleru memorije (0.4)
5. problem je u opštim registrima (0.5)
6. problem je u registru naredbi (0.3)

Naći meru poverenja da postoji problem u operativnoj memoriji i centralnom procesoru i da se radi o kontroleru memorije i opštim registrima ili registru naredbi.

Rešenje

Obeležimo pretpostavke sa e_1 do e_6 onim redosledom kojim su zadate u postavci zadatka. Zaključak z možemo predstaviti izrazom:

$$z = e_1 \wedge e_2 \wedge e_4 \wedge (e_5 \vee e_6)$$

Mere poverenja MB, odnosno nepoverenja MD u zaključak su:

$$MB(e_1 \wedge e_2 \wedge e_4 \wedge (e_5 \vee e_6)) =$$

$$= \min(MB(e_1), MB(e_2), MB(e_4), \max[MB(e_5), MB(e_3)]) =$$

$$= \min(0.6, 0.2, 0.4, \max[0.5, 0.3])$$

$$= 0.2$$

$MD(e_1 \wedge e_2 \wedge e_4 \wedge (e_5 \vee e_6)) = 0$ jer su mere nepoverenja u pretpostavke jednake 0.

Faktor izvesnosti zaključka je:

$$CF(z) = MB(z) - MD(z) = MB(z) = 0.2.$$

Zadatak 102: Računanje izvesnosti zaključka

Poznate su vrednosti faktora izvesnosti sledećih pravila:

- pravilo p_1 koje vodi do zaključka z_1: 0.8
- pravilo p_2 koje vodi do zaključka z_2: 0.8
- pravilo p_3 koje vodi do zaključka z_1: 0.9
- pravilo p_4 koje vodi do zaključka z_2: 0.7
- pravilo p_5 koje, na osnovu z_1 i z_2, vodi zaključku z: 0.6

Ako su pretpostavke pravila p_1, p_2 i p_3 potpuno izvesne, a faktor izvesnosti pretpostavki pravila p_4 iznosi 0.8, izračunati faktor izvesnosti zaključka z.

Rešenje

Izvesnost zaključka z_1 na osnovu pravila p_1 i p_3 je:

$$MB(z_1) = MB'(z_1,e_{p1}) + MB'(z_1,e_{p3}) - MB'(z_1,e_{p1}) * MB'(z_1,e_{p3})$$

$$= 0.8 + 0.9 - 0.8 * 0.9 = 0.98$$

Izvesnost zaključka z_2 na osnovu pravila p_4 je:

$$MB(z_2,e_{p4}) = MB'(z_2,e_{p4}) * MB'(e_{p4}) =$$

$$= 0.7 * 0.8 = 0.56$$

Zbirna izvesnost zaključka z_2 na osnovu pravila p_2 i p_4 je:

$$MB(z_2) = MB'(z_2,e_{p2}) + MB(z_2,e_{p4}) - MB'(z_2,e_{p2}) * MB(z_2,e_{p4})$$

$$= 0.8 + 0.56 - 0.8 * 0.56 = 0.912$$

Izvesnost pretpostavke pravila p_5 je:

$$MB(e_{p5}) = MB(z_1 \wedge z_2) = \min(MB(z_1),MB(z_2)) =$$

$$= \min(0.98, 0.912) = 0.912$$

Tražena izvesnost zaključka z je:

$$CF(z) = MB(z, e_{p5}) = MB'(z, e_{p5}) * MB(e_{p5}) =$$

$$= 0.6 * 0.912 = 0.5472$$

Zadatak 103: Medicinska dijagnostika

Data su neka od pravila ekspertskog sistema za medicinsku dijagnostiku:

P1: AKO pacijent ima manje od 8 ili više od 60 godina
ONDA (1.0) pacijent je u kritičnim godinama

P2: AKO pacijent ima visoku temperaturu I
(pacijent oseća malaksalost ILI

pacijent oseća bolove u mišićima)

ONDA (0.7) pacijent ima grip

P3: AKO pacijent ima natečeno grlo I

pacijent ima kijavicu

ONDA (0.6) pacijent ima grip

P4: AKO pacijent ima grip I

pacijent je u kritičnim godinama

ONDA (0.9) pacijent treba hitno da se obrati lekaru

Pacijent ima 65 godina, visoku temperaturu, natečeno grlo i kijavicu, oseća malaksalost sa izvesnošću 0.8 i bolove u mišićima sa izvesnošću 0.9. Odrediti faktor izvesnosti zaključaka:

a) pacijent ima grip

b) pacijent treba hitno da se obrati lekaru.

Rešenje

a) Zadate su nam mere poverenja u pretpostavke.

e_1: pacijent ima 65 godina MB(e_1) = 1.0

e_2: pacijent ima visoku temperaturu MB(e_2) = 1.0

e_3: pacijent ima natečeno grlo MB(e_3) = 1.0

e_4: pacijent ima kijavicu MB(e_4) = 1.0

e_5: pacijent oseća malaksalost MB(e_5) = 0.8

e_6: pacijent oseća bolove u mišićima MB(e_6) = 0.9.

Zadate su takođe mere poverenja u zaključke iz pojedinih pravila u situaciji potpune izvesnosti pretpostavki pravila (pretpostavka pravila *i* obeležena je sa e_{Pi}):

z_1: pacijent je u kritičnim godinama MB'(z_1, e_{P1}) = 1.0

z_2: pacijent ima grip MB'(z_2, e_{P2}) = 0.7, MB'(z_2, e_{P3}) = 0.6

z_3: pacijent treba hitno da se obrati lekaru MB'(z_4, e_{P4}) = 0.9

Potrebno je izračunati faktore izvesnosti CF pretpostavki pojedinih pravila i na osnovu njih revidirati mere poverenja u zaključke. S obzirom da su mere nepoverenja MD svih pretpostavki jednake nuli, ovo važi i za zaključke, pa su faktori izvesnosti CF zaključaka jednaki njihovim merama poverenja MB. Za odgovor pod a) potrebno je razmotriti pravila P2 i P3.

Mera poverenja u pretpostavku pravila P2 je:

$$MB(e_{P2}) = MB(e_2 \wedge (e_5 \vee e_6)) = \min(MB(e_2) , MB(e_5 \vee e_6)) =$$

$$= \min(MB(e_2), \max[MB(e_5), MB(e_6)])=$$

$$= \min(1.0, \max(0.8,0.9))=$$

$$= 0.9$$

Mera poverenja u zaključak pravila P2 je:

$$MB(z_2 , e_{P2}) = MB'(z_2 , e_{P2}) * MB(e_{P2}) = 0.7 * 0.9 = 0.63$$

Mera poverenja u pretpostavku pravila P3 je:

$$MB(e_{P3}) = MB(e_3 \wedge e_4) = \min(MB(e_3) , MB(e_4)) = \min(1.0, 1.0) = 1.0$$

Mera poverenja u zaključak pravila P3 je:

$$MB(z_2 , e_{P3}) = MB'(z_2 , e_{P3}) * MB(e_{P3}) = 0.6 * 1.0 = 0.6$$

S obzirom da pravila P2 i P3 nezavisno dolaze do istog zaključka z_2 potrebno je izračunati zbirnu meru poverenja u zaključak z_2:

$$MB_{cum}(z_2 , e_{P2, P3}) = MB(z_2 , e_{P2}) + MB(z_2 , e_{P3}) - MB(z_2 , e_{P2}) * MB(z_2 , e_{P3}) =$$

$$= 0.63 + 0.6 - 0.63 * 0.6 =$$

$$= 0.852$$

Prema tome odgovor na pitanje pod a) je da je faktor izvesnosti zaključka da pacijent ima grip jednak $CF(z_2) = MB_{cum}(z_2 , e_{P2, P3}) = 0.852$.

b) Traženi zaključak sledi na osnovu pravila P1 i P4. Pretpostavke pravila P1 su potpuno izvesne; zaključak ovog pravila je takođe potpuno izvestan:

$$MB(z_1 , e_{P1}) = 1.0$$

Mera poverenja u pretpostavku pravila P4 je:

$$MB(e_{P4}) = MB(z_2 \wedge z_1) = \min(MB_{cum}(z_2 , e_{P2, P3}), MB(z_1 , e_{P1})) =$$

$$= \min (0.852, 1.0) = 0.852$$

Mera poverenja u zaključak pravila P4 je:

$$MB(z_4 , e_{P4}) = MB'(z_4 , e_{P4}) * MB(e_{P4}) = 0.9 * 0.852 = 0.7668$$

Prema tome, s obzirom da je mera nepoverenja MD u pravila P1 i P2 jednaka nuli, faktor izvesnosti zaključka da pacijent treba hitno da se obrati lekaru je

$$CF(z_4) = MB(z_4 , e_{P4}) = 0.7668.$$

U rešenju zadatka korišćena je formula da je mera poverenja $MB(z, e_P)$ u zaključak z pravila P jednaka proizvodu mere poverenja MB'() u zaključak z pod potpuno izvesnim pretpostavkama e_P (ova veličina je zadata u pravilu) i mere poverenja u pretpostavku $MB(e_P)$.

$$MB(z, e_P) = MB'(z, e_P) * MB(e_P)$$

Generalan oblik formula za računanje mera poverenja i nepoverenja u zaključak je:

$$MD(z, e_P) = MB'(z, e_P) * \max(0,CF(e_P))$$

$$MD(z, e_P) = MD'(z, e_P) * \max(0,CF(e_P))$$

U datom problemu uvek je važilo da je

$$CF(e_P) = MB(e_P) > 0$$

pa su korišćene uprošćene verzije navedenih formula.

Zadatak 104: Popravak računarskog monitora

U nekom sistemu za popravku računara važe sledeća pravila:

P1: AKO osvetljaj ekrana monitora je stalno na maksimumu
I osvetljaj ekrana monitora ne može da se podesi
ONDA (0.6) ekran monitora treba zameniti.

P2: AKO osvetljaj ekrana monitora ne može da se podesi
I kontrast ne može da se podesi
ONDA (0.7) ekran monitora treba zameniti.

P3: AKO nešto je puklo u monitoru
I osvetljaj ekrana monitora je stalno na maksimumu
ONDA (0.9) kvar je u visokonaponskom kolu monitora.

P4: AKO osvetljaj ekrana monitora stalno je na maksimumu
I kontrast ekrana može da se podesi
ONDA (0.75) kvar je u visokonaponskom kolu monitora.

Poznate su sledeće činjenice: nešto je puklo u monitoru (0.6), osvetljaj ekrana stalno je na maksimumu (1.0), osvetljaj ekrana ne može se podesiti (1.0), a kontrast ekrana ne može da se podesi (0.5). Odrediti koji je kvar izvesniji: u ekranu ili u visokonaponskom kolu.

Rešenje

Zadate su mere poverenja u pretpostavke:

e_1 : nešto je puklo u monitoru, MB(e_1) = 0.6

e_2 : osvetljaj ekrana stalno je na maksimumu MB(e_2) = 1.0

e_3 : osvetljaj ekrana ne može da se podesi MB(e_3) = 1.0

e_4 : kontrast ne može da se podesi MB(e_4) = 0.5

U svakom pravilu P_i data je mera poverenja u zaključak pod uslovom potpune izvesnosti pretpostavki e_{Pi}:

z_1 : ekran monitora treba zameniti. MB'(z_1 , e_{P1}) = 0.6 , MB'(z_1 , e_{P2}) = 0.7

z_2 : kvar je u visokonaponskom kolu monitora. MB'(z_2 , e_{P3}) = 0.9 , MB'(z_2 , e_{P4}) = 0.75

Pravila P1 i P2 vode istom zaključku z_1 , pa je potrebno izračunati zbirnu meru poverenja u ovaj zaključak. Pretpostavke pravila P1 i P2 nisu potpuno nezavisni, ali pošto je zajednička pretpostavka e_3 potpuno izvesna, može se usvojiti pretpostavka o nezavisnosti ova dva pravila jer izvesnost svakog od pravila zavisi od izvesnosti ostalih, nezavisnih pretpostavki.

Mera poverenja u pretpostavku pravila P1 je:

$$MB(e_{P1}) = MB(e_2 \wedge e_3) = \min(MB(e_2), MB(e_3)) =$$

$$= \min(1.0, 1.0) = 1.0$$

Mera poverenja u zaključak pravila P1 je:

$MB(z_1, e_{P1}) = MB'(z_1, e_{P1}) * MB(e_{P1}) = 0.6 * 1.0 = 0.6$

Mera poverenja u pretpostavku pravila P2 je:

$MB(e_{P2}) = MB(e_3 \wedge e_4) = \min(MB(e_3), MB(e_4)) =$

$= \min(1.0, 0.5) = 0.5$

Mera poverenja u zaključak pravila P2 je:

$MB(z_1, e_{P2}) = MB'(z_1, e_{P2}) * MB(e_{P2}) = 0.7 * 0.5 = 0.35$

Zbirna mera poverenja u zaključak z_1 na osnovu ovih pravila je:

$MB_{cum}(z_1, e_{P1, P2}) = MB(z_1, e_{P1}) + MB(z_1, e_{P2}) - MB(z_1, e_{P1}) * MB(z_1, e_{P2}) =$

$= 0.6 + 0.35 - 0.6 * 0.35 = 0.74$

Prema tome, faktor izvesnosti zaključka z_1 je $CF(z_1) = MB_{cum}(z_1, e_{P1, P2}) = 0.74$

Pravila P3 i P4 vode ka zaključku z2. U slučaju ova dva pravila, slično kao i u slučaju pravila P1 i P2 može se usvojiti pretpostavka o međusobnoj nezavisnosti.

Mera poverenja u pretpostavku pravila P3 je:

$MB(e_{P3}) = MB(e_1 \wedge e_2) = \min(MB(e_1), MB(e_2)) =$

$= \min(0.6, 1.0) = 0.6$

Mera poverenja u zaključak pravila P3 je:

$MB(z_2, e_{P3}) = MB'(z_2, e_{P3}) * MB(e_{P3}) = 0.9 * 0.6 = 0.54$

U pretpostavkama pravila P4 sadržana je negacija pretpostavke e_4. Mera nepoverenja MD u negaciju pretpostavke e_4 jednaka je meri poverenja MB u pretpostavku e_4 i obrnuto, pa je

$MB(\neg e_4) = 0$ i $MD(\neg e_4) = 0.5$

Mera poverenja u pretpostavku pravila P4 je:

$MB(e_{P4}) = MB(e_2 \wedge \neg e_4) = \min(MB(e_2), MB(\neg e_4)) =$

$= \min(1.0, 0) = 0$

Mera nepoverenja u pretpostavku pravila P4 je:

$MD(e_{P4}) = MD(e_2 \wedge \neg e_4) = \max(MD(e_2), MD(\neg e_4)) =$

$= \max(0, 0.5) = 0.5$

Faktor izvesnosti pretpostavke pravila P4 je, prema tome

$CF(e_{P4}) = MB(e_{P4}) - MD(e_{P4}) = -0.5$

Mera poverenja u zaključak pravila P4 je:

$MB(z_2, e_{P4}) = MB'(z_2, e_{P4}) * \max(0, CF(e_{P4})) =$

$= 0.75 * \max(0, -0.5) = 0.75 * 0 =$

$= 0$

Mera nepoverenja u zaključak z_2 prema pravilima P3, odnosno P4 pod potpuno izvesnim pretpostavkama je:

$MD'(z_2, e_{P3}) = 0$ i $MD'(z_2, e_{P4}) = 0$

(jer su zadate mere poverenja u zaključak različite od nule) pa su odgovarajuće mere nepoverenja kada pretpostavke nisu potpuno izvesne:

$$MD(z_2, e_{P3}) = MD'(z_2, e_{P3}) * \max(0, CF(e_{P3})) = 0$$

$$MD(z_2, e_{P4}) = MD'(z_2, e_{P4}) * \max(0, CF(e_{P4})) = 0$$

Zbirna mera nepoverenja u zaključak z_2 na osnovu pravila P3 i P4 je takođe jednak nuli:

$$MD_{cum}(z_2, e_{P3,P4}) = 0$$

Zbirni faktor izvesnosti zaključka z_2 određen je, prema tome, zbirnom merom poverenja na osnovu u zaključak z_2 pravila P3 i P4:

$$CF(z_2) = MB_{cum}(z_2, e_{P3,P4}) = MB(z_2, e_{P3}) + MB(z_2, e_{P4}) - MB(z_2, e_{P3}) * MB(z_2, e_{P4}) =$$

$$= 0.54 + 0 - 0.54 * 0$$

$$= 0.54$$

Pošto je $CF(z_1) = 0.74$ veće od $CF(z_2) = 0.54$ to je kvar u ekranu izvesniji od kvara u visokonaponskom kolu monitora.

Pri rešavanju zadatka dobijen je faktor izvesnosti pretpostavke pravila P4:

$$CF(e_{P4}) = -0.5$$

što znači da nemamo poverenja u ovu pretpostavku. Faktor izvesnosti zaključka z_2 na osnovu pravila P4 jednak:

$$CF(z_2, e_{P4}) = MB(z_2, e_{P4}) - MD(z_2, e_{P4}) = 0 - 0 = 0$$

Drugim rečima, pod datim pretpostavkama o zaključku z_2 ništa se ne može zaključiti na osnovu pravila P4. Ovo je potpuno logičan rezultat kada se zna da se iz netačnih pretpostavki može izvući proizvoljan zaključak.

Zadatak 105: Reakcija vlade na terorizam

Data su neka od pravila ekspertskog sistema za zaključivanje o reakciji vlade na terorističke napade:

P1. AKO javno mnenje osuđuje terorističke napade
ONDA (0.7) vlada preduzima oštre mere

P2. AKO broj žrtava u napadima je veliki
I očekuje se intenziviranje terorističke kampanje
ONDA (0.9) javno mnenje osuđuje terorističke napade

P3. AKO postoje informacije o listi mogućih ciljeva
I vreme između napada je kraće od mesec dana
ONDA (0.8) očekuje se intenziviranje terorističke kampanje

P4. AKO ciljevi napada su javne ustanove
ONDA (0.9) broj žrtava je veliki

Ako su ciljevi napada javne ustanove (1.0), postoje informacije o listi mogućih ciljeva (0.75), a vreme između napada kraće je od mesec dana (0.85), odrediti izvesnost preduzimanja oštrih mera od strane vlade.

Rešenje

Pretpostavke su okarakterisane sledećim merama poverenja:

e_1 : ciljevi napada su javne ustanove; MB(e_1) = 1.0

e_2 : postoje informacije o listi mogućih ciljeva; MB(e_2) = 0.75

e_3 : vreme između napada je kraće od mesec dana; MB(e_3) = 0.85

Mere poverenja u zaključke pod izvesnim pretpostavkama pojedinih pravila su:

z_1 : vlada preduzima oštre mere; MB'(z_1 , e_{P1}) = 0.7

z_2 : javno mnjenje osuđuje terorističke napade; MB'(z_2 , e_{P2}) = 0.9

z_3 : očekuje se intenziviranje terorističke kampanje; MB'(z_3 , e_{P3}) = 0.8

z_4 : broj žrtava je veliki; MB'(z_4 , e_{P4}) = 0.9

Pravila razmatramo redosledom P4, P3, P2, P1 jer zaključak svakog od prethodnih pravila u ovom redosledu predstavlja pretpostavku sledećeg pravila.

Mera poverenja u pretpostavku pravila P4 je:

$$MB(e_{P4}) = MB(e_1) = 1.0$$

Mera poverenja u zaključak pravila P4 je:

$$MB(z_4, e_{P4}) = MB'(z_4, e_{P4}) * MB(e_{P4}) = 0.9 * 1.0 = 0.9$$

Mera poverenja u pretpostavku pravila P3 je:

$$MB(e_{P3}) = MB(e_2 \wedge e_3) = \min(MB(e_2), MB(e_3)) =$$

$$= \min(0.75, 0.85) = 0.75$$

Mera poverenja u zaključak pravila P3 je:

$$MB(z_3, e_{P3}) = MB'(z_3, e_{P3}) * MB(e_{P3}) = 0.8 * 0.75 = 0.6$$

Mera poverenja u pretpostavku pravila P2 je:

$$MB(e_{P2}) = MB(z_4 \wedge z_3) = \min(MB(z_4, e_{P4}), MB(z_3, e_{P3})) =$$

$$= \min(0.9, 0.6) = 0.6$$

Mera poverenja u zaključak pravila P2 je:

$$MB(z_2, e_{P2}) = MB'(z_2, e_{P2}) * MB(e_{P2}) = 0.9 * 0.6 = 0.54$$

Mera poverenja u pretpostavku pravila P1 je:

$$MB(e_{P1}) = MB(z_2, e_{P2}) = 0.54$$

Mera poverenja u zaključak pravila P1 je:

$$MB(z_1, e_{P1}) = MB'(z_1, e_{P1}) * MB(e_{P1}) = 0.7 * 0.54 = 0.378$$

Poslednji podatak predstavlja faktor izvesnosti CF da će vlada preuzeti oštre mere.

Treba zapaziti da je, kao posledica indirektnog zaključivanja uz korišćenje niza pravila, faktor izvesnosti zaključka relativno mali i pored toga što pretpostavke i pravila poseduju velike faktore izvesnosti.

Zadatak 106: Klasifikacija zaposlenih

Data su neka od pravila ekspertskog sistema za klasifikaciju zaposlenih.

P1. AKO osoba ima borben duh
I osoba je poverljive prirode
ONDA (0.8) osoba sposobna da preživi

P2. AKO osoba sposobna za putovanja
I osoba sposobna da ubedi druge
I osoba smirena i staložena
ONDA (0.9) osoba sposobna da upravlja

P3. AKO osoba ima govorničku veštinu
I osoba sposobna da preživi
ONDA (0.7) osoba sposobna da ubedi druge

P4. AKO (osoba ume da vozi I starost između 21 i 65)
ILI osoba vozi bicikl
ONDA (0.6) osoba sposobna za putovanja

Poznata su, takođe, zapažanja (i odgovarajući faktori izvesnosti) o osobama Mišku i Blašku navedeni u tabeli 15.

ime	starost	vozi auto	vozi bicikl	govornička veština	poverljiva priroda	borben duh	smirenost i staloženost
Miško	45	1.0	1.0	0.5	1.0	0.4	0.5
Blaško	33	1.0	0.6	0.8	0.5	0.9	0.9

Tabela 15

Odrediti faktor izvesnosti zaključaka da su Miško i Blaško sposobni da upravljaju.

Rešenje

Za Miška važi:

Izvesnosti sposobnosti preživljavanja računamo koristeći pravilo P1. Potrebne mere poverenja o duhu i prirodi osobe čitamo iz tablice. Reč je o konjukciji pretpostavki, pa se uzima minimum mera poverenja.

MB(osoba sposobna da preživi) =

= 0.8 * min(MB(osoba ima borben duh), MB(osoba je poverljive prirode)) =

= 0.8 * min(0.4,1.0) = 0.8 * 0.4 = 0.32

Izvesnost sposobnosti da se ubede drugi računa se iz pravila P3. Izvesnost govorničke veštine je tablična vrednost, a izvesnost sposobnosti za preživljavanje je rezultat pravila P1.

MB(osoba sposobna da ubedi druge) =

= 0.7 * min(MB(osoba ima govorničku veštinu), MB(osoba sposobna da preživi)) =

= 0.7 * min(0.5, 0.32) = 0.7 * 0.32 = 0.224

Sposobnost za putovanja, to jest odgovarajući faktor izvesnosti, dobija se na osnovu pravila P4 i tabličnih podataka za starost i sposobnost vožnje bicikla i automobila. Miškova starost je u intervalu zadatom u okviru pravila, pa je izvesnost te pretpostavke potpuna (1.0).

MB(osoba sposobna za putovanja) =

= 0.6 * max(min[MB(osoba ume da vozi), MB(starost između 21 i 65)],

MB(osoba vozi bicikl)) =

= 0.6 * max(min[1.0, 1.0], 1.0) = 0.6

Tražena veličina, izvesnost sposobnosti za upravljanje, računa se na osnovu pravila P2 koristeći rezultate dobijene primenom prethodnih pravila.

MB(osoba sposobna da upravlja) =

= 0.9* min(MB(osoba sposobna za putovanja), MB(osoba sposobna da ubedi druge),

MB(osoba smirena i staložena)) =

= 0.9 * min(0.6, 0.224, 0.5)

= 0.2016

Za Blaška važi:

Prema pravilu P1:

MB(osoba sposobna da preživi) =

= 0.8 * min(MB(osoba ima borben duh), MB(osoba je poverljive prirode)) =

= 0.8 * min(0.9,0.5) = 0.8 * 0.5 = 0.40

Prema pravilu P3:

MB(osoba sposobna da ubedi druge) =

= 0.7 * min(MB(osoba ima govorničku veštinu), MB(osoba sposobna da preživi)) =

= 0.7 * min(0.8, 0.40) = 0.7 * 0.4 = 0.28

Prema pravilu P4:

MB(osoba sposobna za putovanja) =

= 0.6 * max(min[MB(osoba ume da vozi), MB(starost između 21 i 65)],

MB(osoba vozi bicikl)) =

= 0.6 * max(min[1.0, 1.0], 0.6) = 0.6

Prema pravilu P2:

MB(osoba sposobna da upravlja) =

= 0.9* min(MB(osoba sposobna za putovanja), MB(osoba sposobna da ubedi druge),

MB(osoba smirena i staložena)) =

= 0.9 * min(0.6, 0.28, 0.9)

= 0.252

Faktor izvesnosti da je Miško sposoban da upravlja iznosi 0.2016, dok za Blaška iznosi 0.252.

Zadatak 107: Utvrđivanje gradiva

Poznati su faktori izvesnosti sledećih pravila:

- Pravilo P1 koje vodi ka zaključku z1: 0,3
- Pravilo P2 koje vodi ka zaključku z2: 0,6
- Pravilo P3 koje vodi ka zaključku z1: 0,7
- Pravilo P4 koje vodi ka zaključku z1: 0,5
- Pravilo P5 koje na osnovu z1 or not z2 vodi ka zaključku z: 0,6

Ako su faktori izvesnosti pretpostavki pravila: $CF(e_{P1}) = 0,5$, $CF(e_{P2}) = -1$, $CF(e_{P3}) = 0$, $CF(e_{P4}) = 0,8$, odrediti faktor izvesnosti zaključka z.

Rešenje

Najpre se vrši izračunavanje mere poverenja i nepoverenja u zaključke z1 i z2 na osnovu pravila P1 do P4.

Na osnovu pravila P1 dobijamo

$MB(z1) = MB'(z1) * MB(e_{P1}) = 0,3 * 0,5 = 0,15$

$MD(z1) = MD'(z1) * MD(e_{P1}) = 0 * 0 = 0$

Na osnovu pravila P2 dobijamo

$MB(z2) = MB'(z2) * MB(e_{P2}) = 0,6 * 0 = 0$

$MD(z2) = MD'(z2) * MD(e_{P2}) = 0 * 1 = 0$

Na osnovu pravila P3 dobijamo

$MB(z1) = MB'(z1) * MB(e_{P3}) = 0,7 * 0 = 0$

$MD(z1) = MD'(z1) * MD(e_{P3}) = 0 * 0 = 0$

Na osnovu pravila P4 dobijamo

$MB(z1) = MB'(z1) * MB(e_{P4}) = 0,5 * 0,8 = 0,4$

$MD(z1) = MD(z1) * MD(e_{P4}) = 0 * 0 = 0$

Sada je moguće za zaključak z1 izračunati kumulativnu meru poverenja odnosno nepoverenja

$MBcum(z1, e_{P1,P3}) = MB(z1, e_{P1}) + MB(z1, e_{P3}) – MB(z1, e_{P1}) * MB(z1, e_{P3}) = 0,15$

$MBcum(z1, e_{P1,P2,P3}) = MBcum(z1, e_{P1,P3}) + MB(z1, e_{P4}) – MBcum(z1, e_{P1,P3}) * MB(z1, e_{P4}) = 0,55 – 0,06 = 0,49$

$MDcum(z1, e_{P1,P2,P3}) = 0$

Zaključak z je istinit kada važi $z1 \vee \neg z2$.

Sada se mogu izračunati mera poverenja i nepoverenja u pretpostavku pravila z.

$MB(z1 \vee \neg z2) = MAX(MB(z1), MD(z2)) = MAX(0,49, 0) = 0,49$

$MD(z1 \vee \neg z2) = MIN(MD(z1), MB(z2)) = MIN(0, 0) = 0$

Verovatnoće u zaključak s toga su

$MB(z, e_{P5}) = MB'(z, e_{P5}) * MB(e_{P5}) = 0,6 * 0,49 = 0,294$

$MD(z, e_{P5}) = MD'(z, e_{P5}) * MD(e_{P5}) = 0 * 0 = 0$

Sada je jednostavno odrediti faktor izvesnosti zaključka z

$CF(z, e_{P5}) = MB(z, e_{P5}) – MD(z, e_{P5}) = 0,294$

4.2. Fuzzy logika

Zadatak 108: Rasplinuto zaključivanje

Dati su fazi skupovi A, A' i B. Odrediti zaključak B' koji sledi iz A' na osnovu pravila $A \Rightarrow B$

a) zaključivanjem max-min

b) zaključivanjem max-proizvod.

Pri čemu je poznato:

A = (0.6/a, 1/b, 0.3/c)

B = (0.3/x, 0.8/y, 1/z)

A' = (0.7/a, 0.0/b, 0.0/c)

Rešenje

Zavisnost zaključka B pravila od njegovog preduslova A izražava se fazi relacijom M:

A M = B

U fazi logici, pretpostavka A' može se u opštem slučaju razlikovati od preduslova A pravila. Tada se korišćenjem pravila dobija zaključak B' koji se razlikuje od zaključka B pravila:

B' = A' M

odnosno

$$b_j = \max(\min(a_i, m_{ij})) \; 1 \le i \le n.$$

a) U ovom slučaju, fazi relacija M se definiše sa: $m_{ij} = \min(a_i, b_j)$:

$$M = \begin{bmatrix} 0.3 & 0.6 & 0.6 \\ 0.3 & 0.8 & 1 \\ 0.3 & 0.3 & 0.3 \end{bmatrix}$$

na osnovu čega sledi:

$$B` = A` \cdot M = \begin{bmatrix} 0.7 & 0.0 & 0.0 \end{bmatrix} \cdot \begin{bmatrix} 0.3 & 0.6 & 0.6 \\ 0.3 & 0.8 & 1 \\ 0.3 & 0.3 & 0.3 \end{bmatrix} = \begin{bmatrix} 0.3 & 0.6 & 0.6 \end{bmatrix}$$

b) U ovom slučaju, fazi relacija M se definiše sa: $m_{ij} = a_i \, b_j$

$$M = \begin{bmatrix} 0.18 & 0.48 & 0.6 \\ 0.3 & 0.8 & 1 \\ 0.09 & 0.24 & 0.3 \end{bmatrix}$$

iz čega sledi

$$B` = A` \cdot M = \begin{bmatrix} 0.7 & 0.0 & 0.0 \end{bmatrix} \cdot \begin{bmatrix} 0.18 & 0.48 & 0.6 \\ 0.3 & 0.8 & 1 \\ 0.09 & 0.24 & 0.3 \end{bmatrix} = \begin{bmatrix} 0.18 & 0.48 & 0.6 \end{bmatrix}$$

Zadatak 109: Defazifikacija

Izvršiti defazifikaciju skupa A = (0.36/1, 0.4/2, 0.64/3, 0.8/4, 1/5) metodom centra mase.

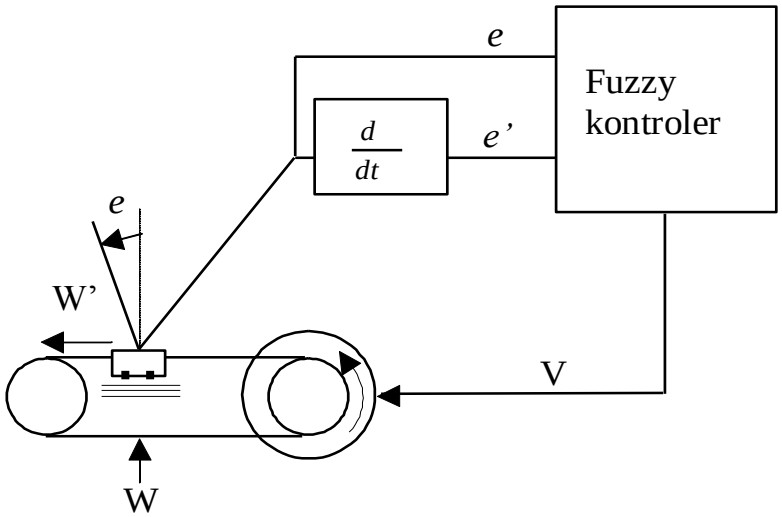
Rešenje

Način dobijanja singularne vrednosti koja reprezentuje skup A je sledeći:

$$a = \frac{\sum \mu_i x_i}{\sum \mu_i} = \frac{0.36 \cdot 1 + 0.4 \cdot 2 + 0.64 \cdot 3 + 0.8 \cdot 4 + 1 \cdot 5}{0.36 + 0.4 + 0.64 + 0.8 + 1} = 3.525$$

Zadatak 110: Projektovanje fuzzy kontrolera

Potrebno je projektovati fuzzy kontroler za postavljanje obrnutog klatna u uspravni položaj (slika 130). Štap koji može da rotira oko jednog svog kraja pričvršćen je na vozilo pomoću nosača. Upravljački zadatak je da se ovaj štap ("obrnuto klatno") postavi u uspravan položaj, tako što će se na pogodan način pokretati vozilo na kome je štap pričvršćen. Tri promen1jive značajne za rešenje ovog zadatka (date na slici) su: *e* - ugao između trenutne pozicije štapa i vertikalne pozicije, *e'* - stepen promene promen1jive *e* (ugaona brzina štapa) i *v* je proporcionalno brzini vozila *w'*. Promenljive *e* i *e'* su upravljačke, a *v* upravljana promenljiva. To znači da na osnovu greške na ulazu upravljamo brzinom vozila, tako da ovu grešku svedemo na što manju vrednost.



Slika 130

Rešenje

Usvajamo da svaka od promenljivih (*e*, *e'*, *v*) ima sedam lingvističkih vrednosti, sa pridruženim trougaonim funkcijama pripadnosti. Podela na sedam lingvističih vrednosti je iskustvenog karaktera i česta je pri izradi fazi kontrolera.

NL: veliki negativan,

NM: srednji negativan,

NS: mali negativan,

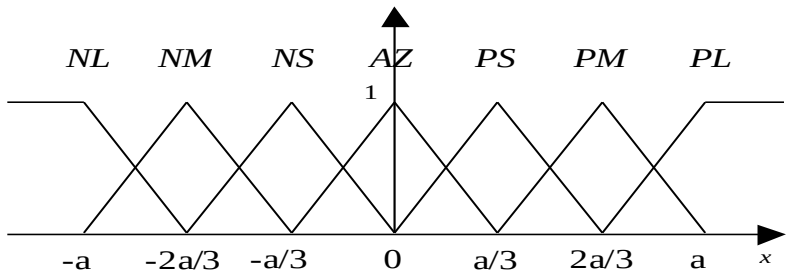
AZ: nula,

PS: mali pozitivan,

PM: srednji pozitivan,

PL: veliki pozitivan.

Lingvističke promenljive prikazane su na slici 131.



Slika 131

Opseg [-*a*,*a*] za svaku promenljivu zavisi od fizičkih ograničenja konkretnog sistema. Na primer, u našem slučaju možemo za ugao otklona da usvojimo da se, zbog pretpostavke manjih poremećaja, menja u opsegu [-60^o,$+60^o$] , umesto [-90^o,$+90^o$].

Skup pravila za upravljanje kontrolerom (primenjuje se zaključivanje max-min, kao defazifikacija metodom centra mase)

R1: Ako je $e = NM$ i $e' = AZ$, onda je $v = NM$,

R3: Ako je $e = NS$ i $e' = NS$, onda je $v = NS$,

R2: Ako je $e = NS$ i $e' = PS$, onda je $v = AZ$,

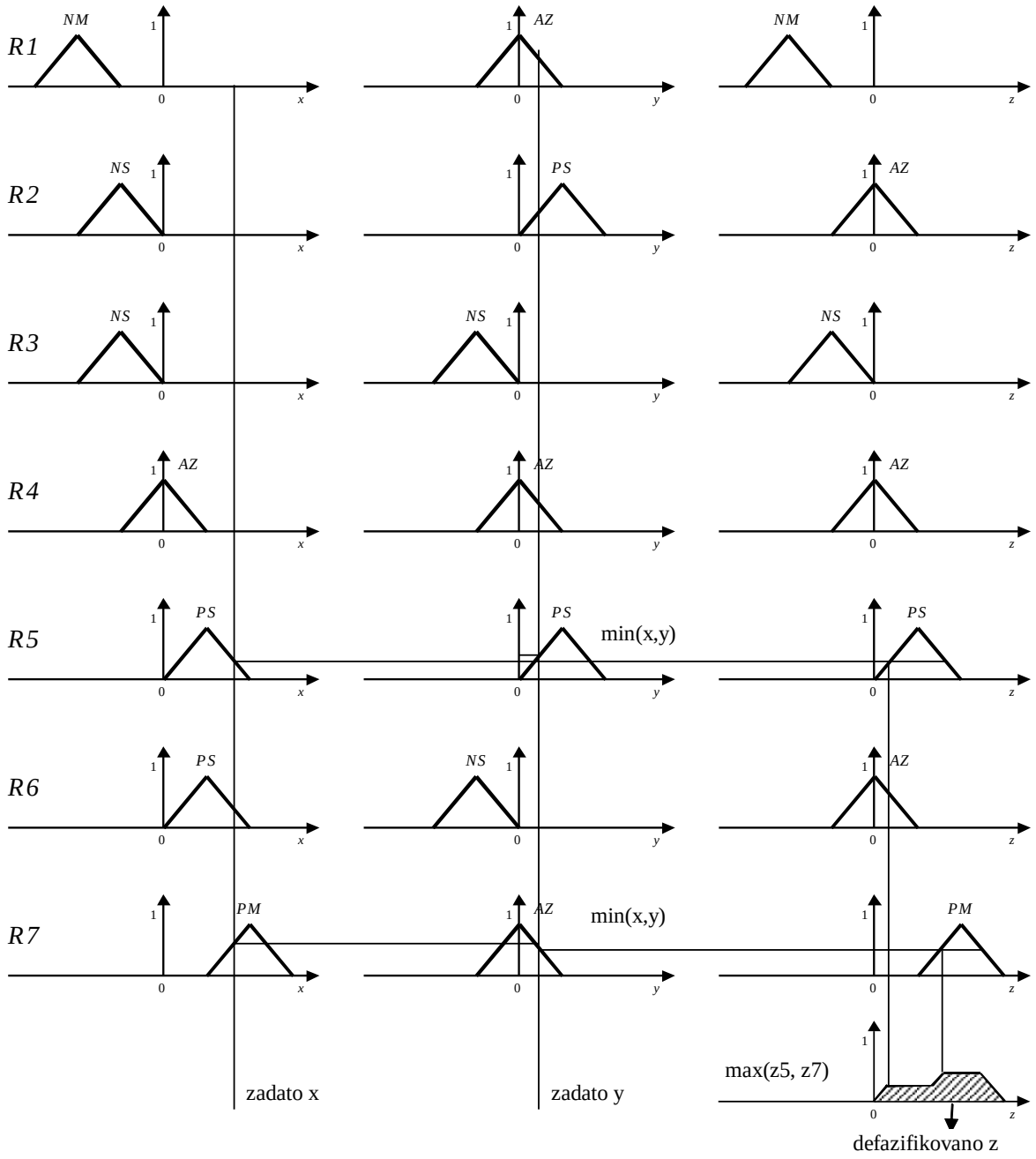
R4: Ako je $e = AZ$ i $e' = AZ$, onda je $v = AZ$,

R6: Ako je $e = PS$ i $e' = NS$, onda je $v = AZ$,

R5: Ako je $e = PS$ i $e' = PS$, onda je $v = PS$,

R7: Ako je $e = PM$ i $e' = AZ$, onda je $v = PM$.

Prethodna pravila prikazana su na slici 132.


R1
NM
AZ
NM
R2
NS
PS
AZ
R3
NS
NS
NS
R4
AZ
AZ
AZ
R5
PS
PS
PS
min(x,y)
R6
PS
NS
AZ
R7
PM
AZ
PM
min(x,y)
max(z5, z7)
zadato x
zadato y
defazifikovano z

Slika 132

Zadatak 111: Donošenje odluka

Osoba želi da izabere jedan od četiri ponuđena posla iz skupa $A = \{ a_1, a_2, a_3, a_4 \}$ sa ciljem da to bude posao koji nudi veliku zaradu (G_1), i sa ograničenjima da je posao zanimljiv (C_1), i da je blizu mesta stanovanja (C_2). Karakteristike poslova su navedene u tabeli 16.

Posao	Godišnja primanja u hiljadama dolara	Rastojanje od mesta Stanovanja u km	Zanimljivost posla
a_1	40000	27	0.4
a_2	45000	7.5	0.6
a_3	50000	12	0.2
a_4	60000	2.5	0.2

Tabela 16

Rešenje

Odluka se donosi u situaciji koja se opisuje sledećim elementima:

A : skup *mogućih akcija*,

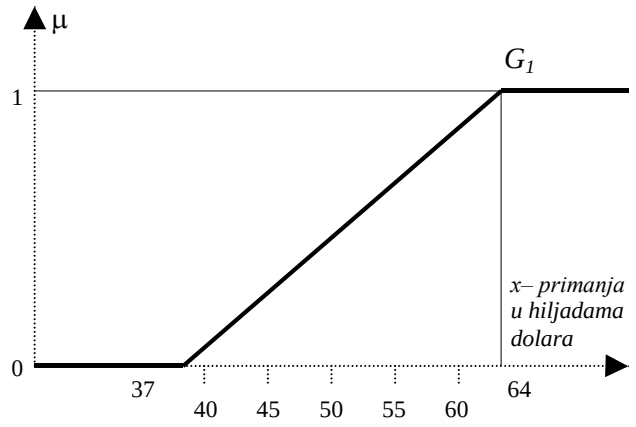
G_i , $i \in N_m$: skup *ciljeva*, od kojih je svaki predstavljen pomoću fazi skupa definisanog nad A

C_j , $j \in N_m$: skup *ograničenja* od kojih je svaki predstavljen pomoću fazi skupa definisanog nad A

Fazi odluka D je fazi set definisan nad A koji istovremeno zadovoljava (u izvesnom stepenu) zadate ciljeve G_i i C_j.

$$D(a) = \left[\inf_{i \in N_n} G_i(a), \inf_{j \in N_m} C_j(a) \right] \qquad \text{za svako } a \in A .$$

Cilj $G_1 = G$ se predstavlja pomoću fazi skupa *velika zarada* prikazanim na slici 133

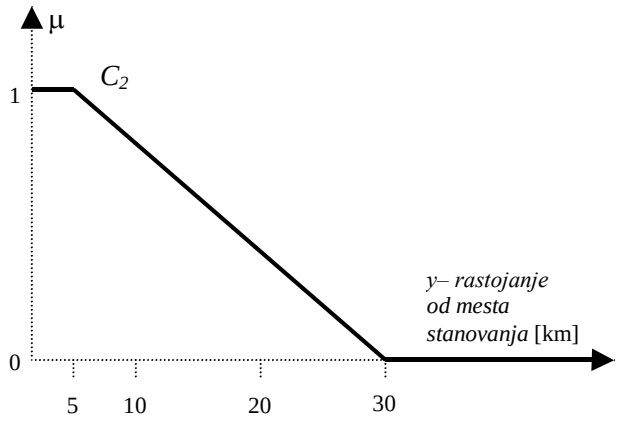


Slika 133

Ograničenje C_1 (interesantan posao) je dato neposredno u postavci:

$$C_1 = \frac{0.4}{a_1} + \frac{0.6}{a_2} + \frac{0.2}{a_3} + \frac{0.2}{a_4}$$

Ograničenje C_2 se predstavlja pomoću fazi skupa *malo rastojanje*, prikazano na slici 134.



Slika 134

Sada ćemo primeniti formulu za fazi odluku, za svaki element skupa *A*:

$$D(a_1) = \min\left[\inf_{i\in\{0\}} G_i(a_1), \inf_{j\in\{1,2\}} C_j(a_1)\right] =$$
$$= \min[0.11, \inf\{0.4, 0.1\}] = \min[0.11, 0.1] = 0.1$$

$$D(a_2) = \min\left[\inf_{i\in\{0\}} G_i(a_2), \inf_{j\in\{1,2\}} C_j(a_2)\right] =$$
$$= \min[0.3, \inf\{0.6, 0.9\}] = \min[0.3, 0.6] = 0.3$$

$$D(a_3) = \min\left[\inf_{i\in\{0\}} G_i(a_3), \inf_{j\in\{1,2\}} C_j(a_3)\right] =$$
$$= \min[0.48, \inf\{0.2, 0.7\}] = \min[0.48, 0.2] = 0.2$$

$$D(a_4) = \min\left[\inf_{i\in\{0\}} G_i(a_4), \inf_{j\in\{1,2\}} C_j(a_4)\right] =$$
$$= \min[0.8, \inf\{0.2, 1\}] = \min[0.8, 0.2] = 0.2$$

što znači da se izraz *poželjan posao* može predstaviti fazi skupom:

$$D = \frac{D(a_1)}{a_1} + \frac{D(a_2)}{a_2} + \frac{D(a_3)}{a_3} + \frac{D(a_4)}{a_4} =$$
$$= \frac{0.1}{a_1} + \frac{0.3}{a_2} + \frac{0.2}{a_3} + \frac{0.4}{a_4}$$

Na osnovu ovog rezultata, zaključuje se da valja izabrati posao a_2, pošto je stepen pripadnosti ovog posla u fazi skupu *poželjan posao* najveći.

4.3. Drugi načini izražavanja neizvesnosti

Zadatak 112: Verovatnoće i nenumeričko izražavanje neizvesnosti

Pretpostavimo da želimo da obrađujemo ILI kombinacije neizvesnosti nenumerički. Pretpostavimo sledeće moguće stepene neizvesnosti: *definitivno*, *verovatno*, *verovatno ne* i *definitivno ne*.

a) Neka je ILI kombinacija bilo koja dva stepena neizvesnosti data tabelom 17:

	definitivno	*verovatno*	*verovatno ne*	*definitivno ne*
definitivno	*definitivno*	*definitivno*	*definitivno*	*definitivno*
verovatno	*definitivno*	*verovatno*	*verovatno*	*verovatno*
verovatno ne	*definitivno*	*verovatno*	*verovatno ne*	*verovatno ne*
definitivno ne	*definitivno*	*verovatno*	*verovatno ne*	*definitivno ne*

Tabela 17

Kojem od numeričkih metoda ILI kombinacija (konzervativnom, nezavisnom ili liberalnom) je ekvivalentna ova tabela?

b) Neka je ILI kombinacija data tabelom 18:

	definitivno	*verovatno*	*verovatno ne*	*definitivno ne*
definitivno	*definitivno*	*definitivno*	*definitivno*	*definitivno*
verovatno	*definitivno*	*definitivno*	*definitivno*	*verovatno*
verovatno ne	*definitivno*	*definitivno*	*verovatno*	*verovatno ne*
definitivno ne	*definitivno*	*verovatno*	*verovatno ne*	*definitivno ne*

Tabela 18

Kojem od numeričkih metoda ILI kombinacija (konzervativnom, nezavisnom ili liberalnom) je ekvivalentna ova tabela?

c) Za slučaj b), pretpostavimo da želimo da preslikamo *definitivno*, *verovatno*, *verovatno ne* i *definitivno ne* u verovatnoće. Logično je da je *definitivno* = 1, a *definitivno ne* 0. Dati intervale mogućih vrednosti verovatnoća za *verovatno* i *verovatno ne* koje bi bile saglasne sa tabelom 18.

Rešenje

U nekim ekspertskim sistemima neizvesnost se izražava korišćenjem verovatnoća: Svakoj činjenici se dodeljuje racionalan broj P ($0 \leq P \leq 1$) koji predstavlja verovatnoću da data činjenica važi. Zaključcima pravila se pridružuju *uslovne verovatnoće* koje izražavaju verovatnoće zaključaka pri ispunjenim preduslovima pravila. Verovatnoće polaznih pretpostavki pri zaključivanju utvrđuju se statističkim metodama (relativnom frekvencijom pojavljivanja događaja opisanog posmatranom činjenicom). Potrebno je definisati način izračunavanja verovatnoće zaključaka na osnovu datih pretpostavki korišćenjem pravila iz baze znanja pri čemu su od interesa su sledeće kombinacije verovatnoća:

- *I kombinacijom verovatnoća* definiše se način računanja verovatnoće zaključka koji predstavlja konjunkciju dveju pretpostavki uz poznate verovatnoće pretpostavki. Na ovaj način se može izračunati verovatnoća za preduslov pravila (pošto je preduslov pravila konjunkcija više pretpostavki). Takođe se na ovaj način može kombinovati verovatnoća preduslova pravila sa uslovnom verovatnoćom zaključka pravila da bi se dobila konačna verovatnoća zaključka.
- *ILI kombinacijom verovatnoća* definiše se način računanja verovatnoće zaključka koji predstavlja disjunkciju dveju pretpostavki uz poznate verovatnoće pretpostavki. Na ovaj način može se izračunati verovatnoća zaključka koji proizilazi iz više različitih pravila.

Način na koji se definišu I i ILI kombinacije verovatnoća pretpostavki zavisi od međusobne statističke zavisnosti ili nezavisnosti pretpostavki. Razlikujemo tri slučaja:

- *Statistički nezavisne* I i ILI kombinacije verovatnoća primenjuju se u slučaju da važi statistička nezavisnost pretpostavki, što znači da ispunjenost jedne od pretpostavki ni na koji način ne utiče na verovatnoću zadovoljenosti drugih pretpostavki.
- *Konzervativne* I i ILI kombinacije verovatnoća daju najnižu procenu verovatnoće zaključka. Konzervativna I kombinacija koristi se u slučaju kada pretpostavke nisu statistički nezavisne, već između njih postoji znatna negativna korelacija, odnosno ispunjenost jedne od pretpostavki povlači u najvećem broju slučajeva neispunjenost ostalih. Konzervativna ILI kombinacija koristi se kada između pretpostavki postoji znatna pozitivna korelacija.
- *Liberalne* I i ILI kombinacije verovatnoća daju najvišu moguću procenu verovatnoće zaključka. Liberalna I kombinacija koristi se u slučaju kada između pretpostavki postoji znatna pozitivna korelacija. Liberalna ILI kombinacija koristi se kada između pretpostavki postoji znatna negativna korelacija.

U tabeli 19 date su formule za statistički nezavisne, konzervativne i liberalne I i ILI kombinacije dve pretpostavke čije su verovatnoće p_1 i p_2. U slučaju kombinovanja većeg broja pretpostavki, formula se prvo primeni na prve dve pretpostavke, zatim se dobijeni rezultat kombinuje sa trećom pretpostavkom i tako dalje. Konačan rezultat ne zavisi od redosleda kojim se razmatraju pretpostavke.

	I kombinacija	ILI kombinacija
nezavisan metod	$p_1 * p_2$	$p_1 + p_2 - p_1 * p_2$
konzervativan metod	$\max(0, p_1 + p_2 - 1)$	$\max(p_1, p_2)$
liberalan metod	$\min(p_1, p_2)$	$\min(1, p_1 + p_2)$

Tabela 19

a) Da bismo mogli primeniti formule iz tabele 19, preslikaćemo redom iskaze *definitivno, verovatno, verovatno ne* i *definitivno ne* u verovatnoće 1, 2/3, 1/3 i 0 čime smo ravnomerno pokrili interval od 0 do 1. Prema tabeli 17, ILI kombinacija *verovatno* i *verovatno* daje rezultat *verovatno*. Razmotrimo redom nezavisan, liberalan i konzervativan metod za ILI kombinaciju odgovarajućih verovatnoća 2/3 i 2/3.

- prema nezavisnom metodu: 2/3 + 2/3 - 2/3 * 2/3 = 8/9 što je bliže broju 1 nego broju 2/3 (granica između brojeva 1 i 2/3 je broj 5/6 koji je manji od 8/9). Prema tome, rezultat bi prema nezavisnom metodu bio opisan sa *definitivno*, što nije odgovarajuće.
- prema liberalnom metodu: min(1, 2/3 + 2/3) = min(1, 4/3) =1 što odgovara iskazu *definitivno*. Za odbacivanje liberalnog metoda računica u ovom slučaju nije ni bila neophodna, s obzirom da liberalni metod daje uvek veću verovatnoću nego nezavisni metod.
- prema konzervativnom metodu: max(2/3, 2/3) = 2/3 što odgovara iskazu *verovatno*. Proverom ostalih ulaza tabele 17, može se utvrditi da se kao rezultat uvek dobija veća od dve verovatnoće što je u skladu upravo sa konzervativnim metodom.

b) Prema tabeli 18, ILI kombinacija *verovatno* i *verovatno ne* daje rezultat *definitivno*. Razmotrimo redom konzervativni, nezavisni i liberalni metod:

- prema konzervativnom metodu: max(2/3, 1/3) = 2/3 što odgovara iskazu *verovatno*, a ne *definitivno*.
- prema nezavisnom metodu: 2/3 + 1/3 - 2/3 * 1/3 = 7/9. Broj 7/9 je bliži broju 2/3 nego broju 1, što odgovara iskazu *verovatno*. Prema tome, ni ovaj metod ne daje zadovoljavajući rezultat.
- prema liberalnom metodu: min (1, 2/3 + 1/3) = 1 što odgovara iskazu *definitivno*. Proverom ostalih ulaza tabele 18 može se utvrditi da su u skladu sa liberalnim metodom.

c) Neka x predstavlja numerički izraženu verovatnoću koja odgovara iskazu *verovatno ne*, a y verovatnoću koja odgovara iskazu *verovatno*. Prema tabeli 18, kombinacija *verovatno ne* i *verovatno ne* daje rezultat *verovatno*. Uvrstimo ove vrednosti u formulu za liberalnu ILI kombinaciju verovatnoća:

$$\min(1, x + x) = y,$$

odnosno $2x = y$ jer mora biti $y < 1$ (*verovatno* mora imati manju verovatnoću nego *definitivno*). Iz poslednjeg uslova sledi da je $2x < 1$, odnosno $x < 1/2$.

Kombinacija *verovatno* i *verovatno ne* daje rezultat *definitivno*, pa imamo da je

$$\min(1, x + y) = 1$$

odnosno $x + y \geq 1$. Zamenom $y = 2x$ dobijamo da je $3x \geq 1$, odnosno $x \geq 1/3$.

Prema tome, da bi tabela 18 bila zadovoljena u slučaju liberalnog metoda, za iskaze *verovatno ne* i *verovatno* mogu se izabrati bilo koje vrednosti verovatnoća x i y, koje ispunjavaju uslove:

$1/3 \leq x < 1/2$

$2/3 \leq y < 1$.

Zadatak 113: Sistem za održavanje istinitosti TMS

Jutro je, i treba se odlučiti za odeću. Pravila kojih se pridržavamo glase:

1. obući farmerke osim ako su prljave ili idemo na razgovor radi zaposlenja
2. ako ne nosimo farmerke, obući odelo
3. ako je hladno, obući džemper
4. ako je zima, hladno je

a) Predstaviti ova pravila listom čvorova slično TMS sistemu. Pretpostaviti da je zima, farmerke nisu prljave i ne idemo na razgovor radi zaposlenja.

b) Pokazati šta se dešava ako u sistem unesemo podatak da idemo na razgovor radi zaposlenja.

Rešenje

Sistem za održavanje istinitosti (engl. *Truth Maintenance System - TMS*) predstavlja pomoćni sistem čiji je cilj održavanje konzistentnosti baze znanja za sisteme koji rade u neizvesnom okruženju.

U TMS-u svaki stav ili pravilo naziva se *čvor*. U bilo kom trenutku čvor može biti bilo u stanju IN, kada verujemo u njegovu istinitost, ili OUT u suprotnom slučaju. Istinitost nekog čvora zavisi, u opštem slučaju, od istinitosti drugih čvorova. Zavisnost se izražava *listom podrške* (engl. *support list - SL*) oblika:

(SL (IN-lista) (OUT-lista)).

Da bi čvor kome je pridruženo "opravdanje" tipa SL bio u stanju IN potrebno je da svi čvorovi navedeni u IN-listi budu u stanju IN, a svi čvorovi navedeni u OUT-listi u stanju OUT.

a) Zadatom skupu pravila odgovaraju sledeći čvorovi u TMS sistemu:

1. Obući farmerke	(SL () (2,3))
2. Farmerke su prljave	(SL () ())
3. Idemo na razgovor	(SL () ())
4. Obući odelo	(SL () (1))
5. Obući džemper	(SL (6) ())
6. Napolju je hladno	(SL (7) ())
7. Godišnje doba je zima	(SL () ())

Čvorovi 2., 3. i 7. imaju prazne IN i OUT liste i predstavljaju *premise*. Njihova istinitost ne zavisi od drugih čvorova u bazi već od opažanja vezanih za okruženje sistema. Čvorove 5. i 6. sa nepraznom IN listom i praznom OUT listom smatramo za tvrđenja dobijena primenom normalnog monotonog zaključivanja. Čvorove sa nepraznom OUT listom (1., 4.) smatramo verovanjima koja usvajamo u odsustvu poznavanja neke činjenice. Kada se činjenično stanje promeni, potrebno je revidirati ova verovanja i zaključke proizašle iz njih.

Prema uslovu zadatka, pretpostavićemo da su inicijalno:

- čvorovi 2. i 3. u stanju OUT, a
- čvor 7 u stanju IN.

Na osnovu toga računamo stanja ostalih čvorova:

- čvor 1 je u stanju IN jer su oba čvora iz njegove OUT liste u stanju OUT
- čvor 4 je u stanju OUT jer je čvor 1 koji se nalazi u njegovoj OUT listi, u stanju IN
- čvor 6 je u stanju IN jer je čvor 7 u stanju IN
- čvor 5 je u stanju IN jer je čvor 6 u stanju IN

b) Činjenica da idemo na razgovor radi zaposlenja odgovara prevođenju čvora 3 u stanje IN. Sada je potrebno ažurirati stanja ostalih čvorova koji direktno ili indirektno zavise od čvora 7.

- čvor 1 ide u stanje OUT i kao posledica toga
- čvor 4 ide u stanje IN.

Ovime je postupak ažuriranja znanja završen.

Zadatak 114: Letovanje

Dejan se sprema za letovanje i treba da donese niz odluka na osnovu sledećih pravila

1. Ako ide na more ne voditi devojku a povesti drugove
2. Ako ide u banju povesti devojku
3. Ako ima dosta para za letovanje ići na more
4. Ako vodi devojku iznajmiti sobu za dvoje
5. Ako ide na more kupiti kremu za sunčanje
6. Ako ide sa roditeljima povesti devojku
7. Ako je nestašica goriva ne ići autom
8. Ako vodi devojku ići autom
9. Ako ne ide sa roditeljima ići autom
10. Ako ide autom ne voditi drugove
11. Ako nema dosta para za letovanje ići u banju

Pretpostavka je da Dejan nema dosta para za letovanje, da ide sa roditeljima i da trenutno ne vlada nestašica goriva.

Predstaviti navedena pravila nizom čvorova kao u TMS sistemu i odrediti stanje svakog čvora.

Rešenje

Identifikovano je 10 čvorova na osnovu pravila datih u postavci zadatka. Svakom čvoru pridružena je lista podrške i trenutno stanje čvora.

Naziv čvora	lista podrške	stanje čvora
1. Ima dosta para za odmor	(SL () ())	OUT
2. Ide na more sa roditeljima	(SL () ())	IN
3. Nestašica je goriva	(SL () ())	OUT
4. Ide na more	(SL (1) ())	OUT
5. Ide u banju	(SL () (1))	IN
6. Voditi devojku	(SL (2,5) (4))	IN
7. Voditi drugove	(SL (4) (10))	OUT
8. Iznajmiti sobu za dvoje	(SL (6) ())	IN
9. Kupiti kremu za sunčanje	(SL (4) ())	OUT
10. Ići autom	(SL (6) (2,3))	OUT

Dodatak: Izabrani algoritmi

Algoritam 1. Pretraživanje po širini (engl. *breadth-first*)

1. Formirati listu čvorova koja inicijalno sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste ciljni čvor
   2.1. Ako je prvi element ciljni čvor, ne raditi ništa.
   2.2. Ako prvi element nije ciljni čvor, ukloniti prvi element iz liste i dodati njegove sledbenike iz stabla pretrage (ako ih ima) *na kraj* liste.
3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretraga je neuspešna.

Algoritam 2. Pretraživanje po dubini (engl. *depth-first*)

1. Formirati listu čvorova koja inicijalno sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste ciljni čvor
   2.3. Ako je prvi element ciljni čvor, ne raditi ništa.
   2.2. Ako prvi element nije ciljni čvor, ukloniti prvi element iz liste i dodati njegove sledbenike iz stabla pretrage (ako ih ima) *na početak* liste.
3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretraga je neuspešna.

Algoritam 3. Pretraživanje metodom planinarenja (engl. *hill-climbing*)

1. Formirati listu čvorova koja inicijalno sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste ciljni čvor
   2.1. Ako je prvi element ciljni čvor, ne raditi ništa.

2.2. Ako prvi element nije ciljni čvor, ukloniti prvi element iz liste, sortirati njegove sledbenike iz stabla pretrage (ako ih ima) po rastućim vrednostima heurističke funkcije. Zatim te sledbenike dodati *na početak* liste tako da prvi element liste bude sledbenik sa najmanjom vrednošću heurističke funkcije.

3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretraga je neuspešna.

Algoritam 4. Pretraživanje metodom prvo najbolji (engl. *best-first*)

1. Formirati listu čvorova koja inicijalno sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste ciljni čvor

   2.1. Ako je prvi element ciljni čvor, ne raditi ništa.

   2.2. Ako prvi element nije ciljni čvor, ukloniti prvi element iz liste i dodati njegove sledbenike iz stabla pretrage (ako ih ima) u listu. Celokupnu listu sortirati po rastućim vrednostima heurističkih funkcija čvorova.

3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretraga je neuspešna.

Algoritam 5. Pretraživanje metodom grananja i ograničavanja (engl. *branch and bound*)

1. Formirati listu parcijalnih putanja. Inicijalno lista sadrži samo jednu putanju nulte dužine koja sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste putanja koja dostiže ciljni čvor.

   2.1. Ako je prva putanja dostigla ciljni čvor, ne raditi ništa.

   2.2. Ako prva putanja nije dostigla ciljni čvor, uraditi sledeće:

      2.2.1. Ukloniti prvu putanju iz liste.

      2.2.2. Za svaki sledbenik poslednjeg čvora na uklonjenoj putanji formirati po jednu novu putanju produžujući sledbenikom uklonjenu putanju.

      2.2.3. Za svaku od novodobijenih putanja izračunati ukupnu (kumulativnu) cenu koštanja c kao zbir cena koštanja operatora na toj putanji.

      2.2.4. Dodati nove putanje u listu parcijalnih putanja.

      2.2.5. Sortirati listu putanja po rastućim vrednostima cena koštanja putanja.

3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretraga je neuspešna.

Algoritam 6. Pretraživanje metodom A*

1. Formirati listu parcijalnih putanja. Inicijalno lista sadrži samo jednu putanju nulte dužine koja sadrži samo startni čvor.
2. Dok se lista čvorova ne isprazni ili se ne dođe do ciljnog čvora, proveriti da li je prvi element liste putanja koja dostiže ciljni čvor.
   2.1. Ako je prva putanja dostigla ciljni čvor, ne raditi ništa.
   2.2. Ako prva putanja nije dostigla ciljni čvor, uraditi sledeće:
      2.2.1. Ukloniti prvu putanju iz liste.
      2.2.2. Za svaki sledbenik poslednjeg čvora na uklonjenoj putanji formirati po jednu novu putanju produžujući sledbenikom uklonjenu putanju.
      2.2.3. Za svaku od novodobijenih putanja izračunati ukupnu (kumulativnu) cenu koštanja c kao zbir cena koštanja operatora na toj putanji; za poslednji čvor na putanji izračunati heurističku funkciju h. Funkciju procene f za svaku od novih putanja izračunati kao zbir heurističke funkcije h i cene koštanja putanje c (f = h + c).
      2.2.4. Dodati nove putanje u listu parcijalnih putanja.
      2.2.5. Sortirati listu putanja po rastućim vrednostima funkcije procene f.
      2.2.6. Ako dve ili više putanja iz liste imaju isti poslednji čvor, ukloniti iz liste sve takve putanje osim jedne koja ima najmanju cenu koštanja (princip dinamičkog programiranja).
3. Ako je pronađen ciljni čvor, pretraga je uspešno završena; u suprotnom pretraga je neuspešna.

Algoritam 7. Minimax algoritam

Dati algoritam je rekurzivan. Svakim korakom rekurzije vrši se izračunavanje najbolje vrednosti u trenutnom koraku. Ovo se postiže razmatranjem svih mogućih poteza iz tekuće pozicije. Za svaki potez rekurzivno se izračunava najbolja vrednost svih rezultujućih stanja. U cilju ograničavanja dubine stanja pretrage u algoritmu postoji i maksimalna dubina do koje će se vršiti pretraživanje. Ako je trenutna dubina jednaka maksimalnoj dozvoljenoj dubini ili smo dospeli u terminalno stanje rekurzija se završava a povratna vrednost jednaka je vrednosti statičke funkcije procene, pozvanoj nad trenutnim stanjem.

Ukoliko je MAX igrač na potezu vraća se maksimalna vrednost, inače minimalna. Ovo je upravo promena poteza igrača, odnosno zamena maksimizujuće i minimizujuće faze. Kada se rekurzivni pozivi vrate do dubine 0, određena je maksimalna vrednost korenog čvora stabla igre, čime je ujedno određen i naredni potez MAX igrača.

U nastavku je dat pseudokod algoritma:

minimax(trenutnoStanje, maxDubina, trenutnaDubina):

ukoliko je terminalno stanje ili je trenutna dubina jednaka maksimalnoj

```
        return vrednost statičke funkcije procene za trenutno stanje
    ukoliko je trenutni igrač MAX
        najboljaVrednost = - BESKONAČNO
    inče ukoliko je trenutni igrač MIN
        najboljaVrednost = BESKONAČNO

    za svaki mogući potez trenutnog igrača određujemo novo stanje i njegovu vrednost
        novoStanje = Kreiraj izgled novog stanja u koje bi se prešlo
        trenutnaVrednost = minimax(novoStanje,maxDubina,trenutnaDubina+1)
        ukoliko je trenutni igrač MAX i trenutnaVrednost je veća od najboljeVrednosti
            najboljaVrednost = trenutnaVrednost
        ukoliko je trenutni igrač MIN i trenutnaVrednost je manja od najboljeVrednosti
            najboljaVrednost = trenutnaVrednost

    return najboljaVrednost
# kraj minimax algoritma
```

Algoritam 8. Minimax algoritam sa alfa-beta odsecanjem

Za razliku od originalne minimax procedure, dodata su dva parametra, alfa i beta. Prilikom pretraživanja proverava se da li tekući potez pripada intervalu koji je ograničen sa ove dve vrednosti. Ukoliko ne, vraća se najbolji potez određen do tog trenutka.

```
minimax(trenutnoStanje, maxDubina, trenutnaDubina, alfa, beta):
    ukoliko je terminalno stanje ili je trenutna dubina jednaka maksimalnoj
        return vrednost statičke funkcije procene za trenutno stanje
    ukoliko je trenutni igrač MAX
        najboljaVrednost = - BESKONAČNO
    inče ukoliko je trenutni igrač MIN
        najboljaVrednost = BESKONAČNO

    za svaki mogući potez trenutnog igrača određujemo novo stanje i njegovu vrednost
        novoStanje = Kreiraj izgled novog stanja u koje bi se prešlo
        trenutnaVrednost = minimax(novoStanje,maxDubina,trenutnaDubina+1,alfa,beta)
        ukoliko je trenutni igrač MAX i trenutnaVrednost je veća od najboljeVrednosti
```

najboljaVrednost = trenutnaVrednost

ukoliko je najboljaVrednost veća ili jednaka beta vrši se odsecanje

return najboljaVrednost

alfa = max (alfa, najboljaVrednost)

ukoliko je trenutni igrač MIN i trenutnaVrednost je manja od najbolјeVrednosti

najboljaVrednost = trenutnaVrednost

ukoliko je najboljaVrednost manja ili jednaka alfa vrši se odsecanje

return najboljaVrednost

beta = min (beta, najboljaVrednost)

return najboljaVrednost

kraj minimax algoritma sa alfa-beta odsecanjem

Algoritam 9. Opšti rešavač problema GPS (engl. *General Problem Solver*)

Da bi se iz tekućeg stanja došlo u ciljno stanje, treba uraditi sledeće:

1. Formirati listu koja će inicijalno sadržavati samo tekuće stanje.
2. Nazovimo stanje na čelu liste tekućim stanjem. Dok se lista ne isprazni ili dok se ne dostigne ciljno stanje, raditi sledeće:
   - 2.1. Ako se pregledom tabele razlika ustanovi da ne postoji neupotrebljen operator za smanjenje razlike između tekućeg i ciljnog stanja, ukloniti tekuće stanje iz liste.
   - 2.2. Inače, izabrati operator za smanjivanje razlike iz tabele razlika.
     - 2.2.1. Ako preduslovi za primenu operatora nisu zadovoljeni, pokušati njihovo zadovoljavanje formiranjem novog ciljnog stanja od tih preduslova i rekurzivnim pozivom GPS algoritma radi dostizanja novog ciljnog stanja iz tekućeg.
     - 2.2.2. Ako su preduslovi zadovoljeni primeniti operator i novodobijeno stanje staviti na početak liste stanja.
3. Ako se dostigne ciljno stanje, obavestiti o uspehu; u suprotnom, obavestiti o neuspehu.

Algoritam 10. AO*

1. Graf pretrage G sastoji se inicijalno samo od startnog čvora n_0. Funkcija procene f za startni čvor jednaka je njegovoj heurističkoj funkciji $f(n_0) = h(n_0)$.
2. Sve dok startni čvor ne bude obeležen kao REŠEN, raditi sledeće:

2.1. Odrediti graf parcijalnog rešenja G' sledeći markirane konektore u grafu pretrage G od startnog čvora. Inicijalno G' se sastoji samo od startnog čvora. Izabrati (na proizvoljan način) neki od nerazvijenih čvorova iz G'.

2.2. Razviti izabrani čvor n. Ako se neki od čvorova naslednika čvora n ne nalazi u grafu, uneti ga u graf i vrednost njegove funkcije procene f postaviti na vrednost heurističke funkcije toga čvora. Ako uneti čvor predstavlja jednog od ciljnih čvorova, obeležiti ga kao REŠENOG.

2.3. Ažurirati funkciju procene izabranog čvora n na sledeći način:

2.3.1. Funkcija procene f(n) čvora n jednaka je minimumu svih funkcija procene f_{Ki} po svakom od izlaznih konektora K_1, K_2 do K_p:

$$f(n) = \min(f_{K1},f_{K2},...,f_{Kp})$$

pri čemu se funkcija procene f_{Ki} izlaznog konektora K_i računa kao zbir cene c_{Ki} konektora K_i i (ranije izračunatih) vrednosti funkcija procene f_{nj} svih čvorova n_1, n_2,...,n_m koji su konektorom K_i spojeni sa izabranim čvorom n:

$$f_{Ki} = c_{Ki} + f(n_1) + f(n_2) + ... + f(n_m).$$

2.3.2. Obeležiti onaj konektor K_i među izlaznim konektorima izabranog čvora n koji ima minimalnu vrednost funkcije procene (prethodno ukloniti oznaku sa ranije obeleženog konektora, ako takav postoji).

2.3.3. Ako su svi čvorovi naslednici čvora n po izabranom konektoru K_i obeleženi kao REŠENI, obeležiti i čvor n kao REŠEN.

2.3.4. Ako je funkcija procene čvora n izračunata u koraku 2.3.1 različita (veća od) stare ili je čvor n u koraku 2.3.3. obeležen kao rešen, ponoviti korak 2.3. (ažurirati njihove funkcije procene i status rešenosti) za svaki od prethodnika čvora n po obeleženom konektoru u grafu G'.

Algoritam 11. Zaključivanje povratnim ulančavanjem

1. Tokom zaključivanja razmatraju se ciljevi koji su predstavljeni konjunkcijom predikata. Za svaki od ciljeva pamti se tekući predikat (predikat do koga se stiglo u razmatranju cilja), i za svaki od zadovoljenih predikata redni broj činjenice ili pravila u bazi znanja koje zadovoljava taj predikat i vezivanja promenljivih nastala tom prilikom.

2. Pozvati proceduru TEST za početni upit. Rezultat koji vrati procedura predstavlja zaključak o zadovoljenosti upita.

Procedura TEST za ispitivanje zadovoljenosti cilja C:

1. Neka je P tekući predikat (ako cilj C nije tokom zaključivanja bio razmatran, P je krajnje levi predikat). Neka je N redni broj u bazi znanja činjenice ili pravila koje je poslednje korišćeno za zadovoljavanje predikata P cilja C (inicijalno nula ako se predikat P razmatra prvi put). Poništiti sva vezivanja promenljivih argumenata predikata P nastala ranijim razmatranjem ovog predikata (ako takva postoje; ovo ne uključuje vezivanja promenljivih nastala razmatranjem predikata levo od P u cilju C).

2. Ustanoviti zadovoljenost predikata P:

   2.1. Ako postoje činjenice koje uparuju predikat P, izabrati prvu od njih sa rednim brojem većim od N; neka je to činjenica F. Za predikat P zapamtiti redni broj činjenice F i odgovarajuća vezivanja promenljivih. Predikat P je zadovoljen.

   2.2. Ako postoje pravila čiji se zaključak može upariti sa predikatom P, izabrati prvo od njih sa rednim brojem jednakim ili većim od N; neka je to pravilo R. Zapamtiti redni broj pravila R uz predikat P (nova vrednost za N). Promenljive predikata P koje nisu ranije vezane, vezuju se za odgovarajuće promenljive pravila R. Vezane promenljive predikata P smenjuju odgovarajuće promenljive desne strane pravila R. Ovako transformisana desna strana pravila R postaje tekući cilj u novom pozivu procedure TEST. Ako se cilj ne zadovolji, uvećati N u tekućem predikatu za jedan i ponoviti korak 2.2.

   2.3. U suprotnom, predikat P ne može se zadovoljiti.

3. Ako P nije uspeo, a nije krajnje levi, predikat P1 levo od P postaje tekući. Preći na korak 1. Napomena: ovaj korak predstavlja vraćanje u zaključivanju (engl. *backtracking*).

4. Ako je P uspeo, a nije krajnje desni, predikat desno od P postaje tekući. Preći na korak 1.

5. Ako P nije uspeo, a krajnje je levi, cilj C nije zadovoljen. Sledi povratak iz procedure TEST.

6. Ako P jeste uspeo, a krajnje je desni, cilj C jeste zadovoljen. Sledi povratak iz procedure TEST. Napomena: informacije pridružene predikatima cilja C moraju se sačuvati zbog mogućeg naknadnog vraćanja u zaključivanju.

Kraj procedure TEST

Algoritam 12. Zaključivanje direktnim ulančavanjem

1. Formirati listu L neupotrebljenih činjenica od zadatih činjenica pri čenu treba očuvati redosled zadatih činjenica.

2. Izabrati činjenicu F sa početka liste L i ukloniti je sa liste. Činjenica F se *sledi* na sledeći način:

   2.1. Za svako pravilo R koje u preduslovu sadrži predikat P koji može (vezivanjem promenljivih) upariti činjenicu F preduzeti akcije 2.1.1. do 2.1.4. **Pravila koja sadrže not(P) se ignorišu u ovom koraku**.

      2.1.1. Kreirati novo pravilo R' koje je istog oblika kao pravilo R s tom razlikom što je predikat P uklonjen iz preduslova pravila R. Ako je pri uparivanju činjenice F predikatom P došlo do vezivanja promenljivih, u pravilu R' se pojavljuju ove zamenjene vrednosti promenljivih.

      2.1.2. Ako se novo pravilo R' sastoji samo od zaključka (preduslov ne postoji) radi se o novodobijenoj činjenici koja se stavlja na čelo liste L. Iz baze znanja ukloniti sva pravila čija je desna strana (zaključak) istovetna novoj činjenici. Ne uklanjati pravila kod kojih zaključci mogu upariti

novu činjenicu, ali su opštiji od nove činjenice. U ovom slučaju preskaču se koraci 2.1.3. i 2.1.4.

2.1.3. Ako novo pravilo R' pored zaključka sadrži i preduslov, potrebno je novo pravilo staviti u bazu znanja. Novo pravilo se stavlja neposredno ispred pravila R od koga je nastalo, osim u slučaju kada postoji još neki predikat u preduslovu pravila R koji može upariti činjenicu F. Tada se novo pravilo R' stavlja neposredno iza pravila R.

2.1.4. Ukoliko je pravilo R suvišno posle dodavanja pravila R', ukloniti pravilo R iz baze znanja. Pravilo R je suvišno ako njegov zaključak nije opštiji od zaključka pravila R'.

3. Ponavljati korak 2 sve dok se lista L ne isprazni.

4. Razmatrati redom pravila: za svaki stav oblika not(P) koji se pojavljuje u pravilima, ispitati da li predikat P uparuje neku od činjenica. Ako je odgovor negativan, dodati not(P) listi L neupotrebljenih činjenica i ponoviti korak 2.

Algoritam 13. Zaključivanje cikličkim hibridnim ulančavanjem

1. Sve dok se dobijaju nove činjenice ponavljati korak 2, u suprotnom preći na korak 3.
2. Razmatrati redom sva pravila u bazi znanja, **ignorišući pravila sa not**.

2.1. Za svako pravilo R, tretirati njegov preduslov kao upit o činjenicama (bez upotrebe drugih pravila, na primer, povratnim ulančavanjem). Ako je preduslov ispunjen, dodati zaključak pravila R sa eventualnim smenama promenljivih na čelo liste činjenica.

2.2. Po dodavanju nove činjenice, eliminisati iz daljeg razmatranja sva pravila čiji se zaključci sadrže u novoj činjenici (dakle ili su identični, ili se zamenom promenljivih u novoj činjenici može dobiti zaključak). Ponoviti korake 2.1. i 2.2. za sve moguće smene promenljivih u pravilu.

3. Ponavljati korake 2.1. i 2.2. (sve dok se dobijaju nove činjenice), sa svim originalnim pravilima (uključujući i ona sa not), pri čemu se stavovi oblika not(p) smatraju tačnim ako p nije među zadatim i dobijenim činjenicama.

Algoritam 14. Unifikacija predikatskih stavova

1. Predstaviti oba literala koja se unificiraju listama u kojoj je predikatni simbol prvi element iza kojeg slede argumenti tačno po redosledu.
2. Napustiti proceduru ako dve liste nisu iste dužine (unifikacija se ne može sprovesti).
3. Porediti elemente koji se nalaze na istim pozicijama u obe liste:

3.1. Predikatski i funkcijski simboli, kao i konstante moraju biti identični.

3.2. Za promenljive izvesti uparivanje putem zamene; kada se naiđe na promenljivu, zameniti je, kao i sva njena dešavanja u listi odgovarajućim elementima druge liste. Jedino ograničenje u pogledu uparivanja je da promenljiva ne može biti

zamenjena izrazom koji sadrži istu promenljivu (da bi se sprečila beskonačna zamena).

4. Dva predikata su unificirana ako se svi elementi poklapaju posle zamene promenljivih.

Algoritam 15. Nasleđivanje uz prisustvo *default* vrednosti i *if-needed* procedura (Z- nasleđivanje):

Neka je F zadati čvor, a S zadati pregradak.

1. Formirati red koji sadrži čvor F i sve čvorove povezane IS_A vezom sa čvorom F (čvor F je 'rep' strelice). Čvor F treba da se nalazi na početku reda.
2. Dok se red ne isprazni, ili se nađe vrednost, utvrditi da li prvi element reda poseduje traženi pregradak S.
   - 2.1. Ako čvor poseduje pregradak S i u njemu se nalazi vrednost skok na korak 3.
   - 2.2. Ako čvor poseduje pregradak S sa if-needed procedurom P: ako se mogu odrediti ulazni parametri procedure P (ovo može uključiti rekurzivni poziv procedure nasleđivanja počev od čvora F) izvršiti proceduru P čiji rezultat predstavlja traženu vrednost; skok na korak 3.
   - 2.3. Ako čvor poseduje pregradak S sa podrazumevanom vrednošću skok na korak 3.
   - 2.4. U suprotnom, ukloniti prvi element iz reda i na kraj reda dodati njegove sledbenike u mreži po AKO vezama.
3. Procedura završava rad i vraća ili nađenu vrednost ili indikator da vrednost nije nađena.

Algoritam 16. Zadovoljavanje ograničenja metodom relaksacije

1. Napraviti liste mogućih vrednosti za sve slobodne promenljive (one čije su nam vrednosti potrebne) u upitu.
2. Obeležiti svaku od promenljivih kao 'aktivnu'.
3. Ponavljati korake 3.1. do 3.4. sve dok postoji bar jedna aktivna promenljiva.
   - 3.1. Izabrati neku od aktivnih promenljvih A. (Pri izboru se može primeniti neka heuristika).
   - 3.2. Za svaku od mogućih vrednosti V za A razmatrati svako od ograničenja C u upitu u kome figuriše promenljiva A
     - 3.2.1. Proveriti da li je ograničenje C zadovoljeno ako A ima vrednost V (pri čemu se mogu vezati i druge promenljive ako je potrebno).
     - 3.2.2. Ako se C ne može zadovoljiti, izbaciti V iz liste mogućih vrednosti za A i ne razmatrati ostala ograničenja. U suprotnom, ne raditi ništa.
     - 3.2.3. Pri razmatranju sledećeg ograničenja zanemariti vezane vrednosti promenljivih dobijene pri razmatranju prethodnog ograničenja.

3.3. Obeležiti promenljivu A kao 'neaktivnu'. Ako je neka od mogućih vrednosti za A eliminisana u prethodnim koracima, obeležiti kao 'aktivne' sve promenljive koje ispunjavaju sledeća tri uslova:

- pomenute su u ograničenjima sa A
- prethodno su bile neaktivne i
- u skupu mogućih vrednosti imaju više od jedne vrednosti.

3.4. Ako je skup vrednosti za A sveden na jednu vrednost, zameniti tu vrednost u svim ograničenjima upita koja pominju A i eliminisati iz daljeg razmatranja sva ograničenja u kojima se ne pojavljuje nijedna promenljiva posle smene A.

4. Ukoliko u ovom koraku svaka promenljiva ima jednoznačno određenu vrednost, problem je rešen. U suprotnom, primeniti neku od tehnika pretraživanja pri čemu promenljive uzimaju samo one vrednosti koje su ostale u listi mogućih vrednosti.

Literatura

1. Russell, Norvig, *Artificial Intelligence A Modern Approach, Second Edition*, Prentice Hall, 2003.
2. Jackson, *Introduction to Expert Systems, Third Edition*, Addison-Wesley, 1999.
3. Devedžić, *Ekspertni sistemi za rad u realnom vremenu*, Institut Mihajlo Pupin, 1994.
4. Durkin, *Expert Systems/Design and Development*, Macmillian Publishing Company, 1994.
5. Frost, *Introduction to Knowledge Based Systems*, Collins Professional and Technical Books, 1986.
6. Harmon, King, *Expert Systems/Artificial Intelligence in Business*, John Willey & Sons, 1985.
7. Harmon, Maus, Morrissey, *Expert Systems/Tools & Applications*, John-Willey & Sons, 1988.
8. Hayes Roth, Jacobstein, *The State of Knowledge-Based Systems*, Communications of the ACM, March 1994, pp. 27-40.
9. Keller, *Expert System Technology/Development & Application*, Prentice-Hall, 1987.
10. Laurere, *Problem solving and Artificial Intelligence*, Prentice-Hall 1990.
11. Nilsson, *Principles of Artificial Intelligence*, Springer-Verlag, 1982.
12. Rolston, *Principles of Artificial Intelligence and Expert Systems Development*, McGraw-Hill, 1988.
13. Rowe, *Artificial Intelligence Through Prolog*, Prentice-Hall, 1988.
14. Winston, *Artificial Intelligence*, Adisson-Wesley 1984.
15. Zukowski, *Mastering Java 2 J2SE 1.4*, Kompjuterska biblioteka, 2002.
16. Subašić, *Fazi logika i neuronske mreže*, Tehnička knjiga, 1997.