1. zad. - rješenje

Usporedba algoritama pretraživanja u dubinu (DFS), pretraživanja u širinu (BFS) i pretraživanja jednolike cijene (UCS = Uniform Cost Search), uz standardne oznake za veličine koje opisuju složenost problema:

- b = faktor grananja
- d = dubina najplićeg rješenja (najmanje dubine)
- m = maksimalna dubina stabla pretraživanja (za DFS, m može biti ∞)
- C^* = optimalna cijena do cilja (za UCS)
- ε = minimalna cijena prijelaza u susj. stanje (UCS)

$\Rightarrow$ dubina d za UCS je $d \leq \lfloor C^*/\varepsilon \rfloor$

Svojstvo \ Alg =	DFS	BFS	UCS
Potpunost = nalazi rješenje ako postoji	NE	DA	DA
Optimalnost = nalazi optimalno rješenje	NE	DA	DA
Vremenska složenost	$O(b^m)$	$O(b^{d+1})$	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$
Prostorna složenost	$O(b \cdot m)$	$O(b^{d+1})$	$\underbrace{O(b^{1+\lfloor C^*/\varepsilon \rfloor})}_{O(b^{d+1})}$ s gornjom ogradom za $d \leq \lfloor C^*/\varepsilon \rfloor$
Struktura podataka za otvorene čvorove	stog	red	prioritetni red (sortiran ↑ po cijeni)

- Mane DFS – ne mora završiti ($m \to \infty$), bez kontrole i izbjegavanja već posjećenih stanja
- Prednost – prostorna složenost samo $O(b \cdot m)$ za pamćenje svih otvorenih čvorova. (Može i bolje – u backtracking varijanti treba samo $O(m)$).

Oznake čvorova za pamćenje putera - preko indeksa:

D_B = "roditelj od D je B"
= u stanje D stigli iz B

(a) DFS - vrh stoga je desno, cijene nemaju ulogu

trenutni čvor	STOG	lista posjećenih	roditelj (put) unatrag
početak –	A	–	

[susjedi od A su B, C, E]

A	$E_A\ C_A\ B_A$	A	$B_A \to A$

[susjedi od B su ~~A~~, D]

B_A	$E_A\ C_A\ D_B$	A B	$D_B \to B_A$

[susjedi od D su ~~B~~, E, F]

D_B	$E_A\ C_A\ F_D\ E_D$	A B D	$E_D \to D_B$

~~[susjedi od E su ~~A~~, C, ~~D~~, F]~~

~~E_D | $E_A\ C_A\ F_D\ F_E$ | ABDE | $F_E \to E$~~

~~[F_E je cilj ⇒ STOP]~~

~~F_E~~

~~Nađeni put (čitamo roditelje unatrag i onda pišemo unaprijed)~~

~~$A \to B \to D \to E \to F$~~

[susjedi od E su ~~A~~, C, ~~D~~, F]

E_D	$E_A\ C_A\ F_D\ F_E\ C_E$	A B D E	$C_E \to E_D$

[susjedi od C su ~~A~~, ~~E~~]

C_E	$E_A\ C_A\ F_D\ F_E$	A B D E C	$F_E \to E_D$

[F_E je cilj ⇒ STOP]

F_E

$\boxed{A \to B \to D \to E \to F}$

Nađeni put (čitamo roditelje unatrag i onda pišemo unaprijed)

$F_E \to E_D \to D_B \to B_A \to A$

(b) BFS - početak <u>reda</u> je lijevo, kraj je desno $\boxed{1-3}$

	trenutni čvor	RED	lista posj.	roditelj (put unatrag)
početak	—	A	—	
[susjedi od A su B, C, E]				
	A	$B_A \; C_A \; E_A$	A	$B_A \to A$
[susjedi od B su ~~A~~, D]				
	B_A	$C_A \; E_A \; D_B$	$A \; B$	$C_A \to A$
[susjedi od C su ~~A~~, E]				
	C_A	$E_A \; D_B \; E_C$	$A \; B \; C$	$E_A \to A$
[susjedi od E su ~~A~~ ~~C~~, D, F]				
	E_A	$D_B \; E_C \; D_E \; F_E$	$A \; B \; C \; E$	$D_B \to B_A$
[susjedi od D su ~~B~~, ~~E~~, F]				
	D_B	$\underbrace{\cancel{E_C} \; \cancel{D_E}}_{\text{već posj.}} \; F_E \; F_D$	$A \; B \; C \; E \; D$	$F_E \to E_A$
[F je cilj ⇒ STOP]				
	F_E			

Put unatrag: $F_E \to E_A \to A$ i napišemo unaprijed

$$\boxed{A \to E \to F}$$

(c) UCS – početak prioritetnog reda je lijevo, sortirano uzlazno po cijeni slijeva udesno

Uz čvor s indeksom roditelja pišemo i cijenu puta do njega (preko roditelja)

	trenutni čvor	PRIORITETNI RED	lista posjećenih	roditelj (put unatrag)
početak		$\boxed{(A,\emptyset)}$		

[susjedi od $(A,\emptyset)$ su $(B,2)$, $(C,3)$, $(E,5)$ – sortirano po zbroju s $\emptyset$ dodamo u RED]

$(A,\emptyset)$	$\boxed{(B_A,2)}$ $(C_A,3)$ $(E_A,5)$	A	$B_A \xrightarrow{2} A$

[susjedi od $(B,2)$ su ~~(A,2)~~ i $(D,4)$ – zbrojimo duljinu i $(D,6)$ dodamo – na kraj]

$(B_A,2)$	$\boxed{(C_A,3)}$ $(E_A,5)$ $(D_B,6)$	AB	$C_A \xrightarrow{3} A$

[susjedi od $(C,3)$ su ~~(A,3)~~, $(E,4)$ – zbrojimo i $(E_C,7)$ dodamo, opet ide na kraj]

$(C_A,3)$	$\boxed{(E_A,5)}$ $(D_B,6)$ $(E_C,7)$	ABC	$E_A \xrightarrow{5} A$

[susjedi od $(E_A,5)$ su ~~(A,5)~~, ~~(C,4)~~, $(D,1)$, $(F,5)$ – zbrojimo udalj. i dobivamo $(D_E,6)$, $(F_E,10)$ – dodamo sortirano u red]

$(E_A,5)$	$\boxed{(D_B,6)}$ $(D_E,6)$ $(E_C,7)$ $(F_E,10)$	ABCE	$D_B \xrightarrow{4} B_A$

→ (smijemo uzeti i $(D_E,6)$ – idem po manjem roditelju)

[susjedi od $(D_B,6)$ su ~~(B,4)~~, ~~(E,1)~~, $(F,2)$ – zbrojimo udaljenosti i dodamo $(F_D,8)$ u red]

$(D_B,6)$	$\underbrace{\cancel{(D_E,6)}\ \cancel{(E_C,7)}}_{D,E\text{ već posj.}}$ $\boxed{(F_D,8)}$ $(F_E,10)$	ABCED	$F_D \xrightarrow{2} D_B$

[F je cilj => STOP]

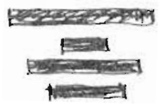
$(F_E, 8)$ => duljina najkraćeg puta od A do F je 8

Unatrag: $F_E \xrightarrow{2} D_B \xrightarrow{4} B_A \xrightarrow{2} A$ i put je $\boxed{A \to B \to D \to F}$

– Da smo uzeli $(D_E,6)$ dobili bismo isto $(F_E,8)$ i put $A \to E \to D \to F$

2. zad. - rješenje

Ilustracija stanja za $n=4$:



Ako palačinke numeriramo po veličini najmanja $=1$, ..., najveća $=n$ i složimo uređenu n-torku od vrha prema dnu – dobivamo permutaciju brojeva $1,\dots,n$.

Dakle, stanja promatramo kao $\boxed{\text{permutacije } p\in S_n}$

Na pr. gornje stanje odgovara permutaciji $(4,1,3,2)$

Ciljno stanje je permutacija $p = id = (1,2,\dots,n)$ (identiteta) – svi brojevi na svom mjestu, sortirano uzlazno

- Dozvoljeni potez = okrenuti gornjih k palačinki, $2\le k\le n$,
  = okrenuti naopako prvih k elemenata u permutaciji. Cijena tog poteza je k

$$(p_1,\dots,p_k,p_{k+1},\dots,p_n) \xrightarrow[k]{} (\underbrace{p_k,\dots,p_1}_{\text{naopako!}},p_{k+1},\dots,p_n)$$

- Dakle, za $k=2,\dots,n$, svaka permutacija p može imati $n-1$ "sljedbenika". Uočiti da su potezi reverzibilni (dva puta okrenemo blok iste veličine k i vratimo se u polazno stanje).
  Drugim riječima, graf stanja je neusmjeren (grane idu simetrično u oba smjera $\Longleftrightarrow$).

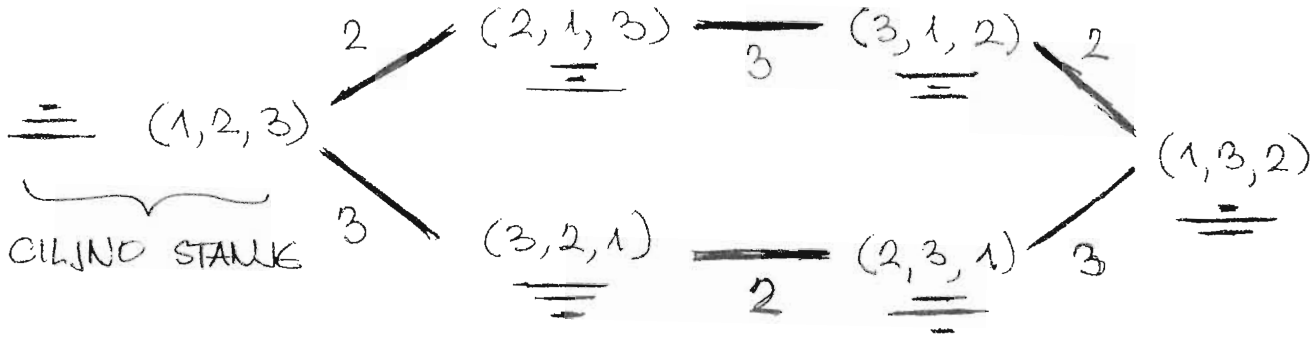
- Na primjer, sva moguća "sljedbenička" stanja za gornji primjer su

$$\text{sljedbenici}(p=(4,1,3,2)) = \left\{ \begin{array}{ccc} k & = \text{cijena} & \text{succ} \\ \hline 2 & 2 & (1,4,3,2) \\ 3 & 3 & (3,1,4,2) \\ 4 & 4 & (2,3,1,4) \end{array} \right.$$

- Iz svakog stanja p imamo $n-1$ grana, svaka ima različitu cijenu, od 2 do n.

2-2

(a) Za $n=3$ imamo $3! = 6$ različitih stanja, a iz svakog stanja idu dvije grane "okreni 2" i "okreni 3".
Slika = prstenasti šesterokut:



Usmjereno ili informirano pretraživanje strategijom "najbolji prvi" ("best-first"):
varijanta pretraživanja jednolike cijene (tj. po najboljem čvoru u prioritetnom redu – tj. po prvom), uz korištenje funkcije procjene f (evaluacijska funkcija), umjesto prave cijene.

— —

Odgovori na "teorijska" pitanja:

(b) $f(n)$ = procjena ~~puta~~ cijene puta od početnog stanja do cilja, kroz čvor n.

$g(n)$ = prava (optimalna) cijena puta do n

Standardni rastav funkcije f za "usmjeravanje" prema cilju

$$f(n) = g(n) + h(\text{stanje}(n))$$

gdje je $h(s)$ heuristička funkcija koja ovisi samo o stanjima (ne o čvorovima u pretrazi), sa značenjem

$h(s)$ = procjena cijene (najjeftinijeg) puta od s do cilja

U svim razumnim izborima je h nenegativna (tj. cijena je ≥ 0, inače je besmislena)

– Ako uzmemo $g = 0$, tj.

$$f(n) = h(\text{stanje}(n))$$

dobivamo "čisto heuristički" algoritam "<u>pohlepni najbolji prvi</u>" (greedy best-first), koji (naravno) NE MORA naći optimalno rješenje.

– Ako uzmemo $h = 0$, tj.

$$f(n) = g(n)$$

onda dobivamo obični algoritam <u>neusmjerenog pretraživanja jednolike cijene</u> (UCS).

(c) Heuristika h je <u>dopustiva</u> ili <u>optimistična</u>, ako nikad <u>ne precjenjuje</u> stvarni najbolji put, tj.

za <u>svako</u> stanje $s \in S$ vrijedi

$$\boxed{h(s) \leq h^*(s)}$$

gdje je $h^*(s)$ prava = <u>optimalna</u> cijena puta od s do cilja.

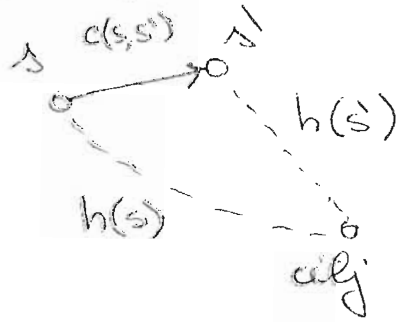
– Dodatno, uzima se da je $h(s) \geq 0$ (smisleno!), pa u završnom stanju = cilju <u>mora</u> vrijediti

$$\underline{h(\text{cilj}) = \emptyset.}$$

(d) Heuristika h je <u>konzistentna</u> ili <u>monotona</u> ako i samo ako u svakom stanju $s \in S$ zadovoljava sljedeću nejednakost trokuta:

Za <u>bilo kojeg</u> sljedbenika $s' \in \text{succ}(s)$ vrijedi

$$\boxed{h(s) \leq h(s') + c(s,s')}$$



gdje je $c(s,s')$ cijena (direktnog) puta od s do s'.

<u>Posljedice</u>: funkcija f je <u>padajuća</u> duž <u>svakog</u> puta

$\Rightarrow$ A^* algoritam je <u>optimalan</u>. (u trenutku kad A^* izabere čvor n za proširenje ima <u>optimalan put</u> do n.)

(Usput, $\Rightarrow$ h je dopustiva)

(e) Heuristika h_1 dominira nad heuristikom h_2 (ili h_1 je bolje informirana od h_2) ako je $h_1 \geq h_2$ u svim stanjima, tj.

$$\boxed{\forall s \in S \quad h_1(s) \geq h_2(s)}$$

Uz dodatak dopustivosti (ili konzistentnosti u A^*) <u>najbolje</u> informirana heuristika je h^* (v. dopustivost)

$h^*(s)$ = optimalna cijena od s do cilja.

(Više od toga je zabranjeno zbog dopustivosti i onda algoritam ne mora biti optimalan)

Zapišimo zadane heuristike u modelu označavanja stanja permutacijama. Neka je

$$p = (p_1, p_2, \dots, p_n)$$

permutacija koja opisuje stanje s (trebalo bi pisati $p(s) = (p_1(s), \dots, p_n(s))$ – ali ispuštamo "s", jasno je iz konteksta).

h_1 = broj palačinki koje nisu na svom mjestu

$$\boxed{h_1 = \text{card}\{\, i \mid p_i \neq i \,\}} \qquad (\text{naravno, } 1 \leq i \leq n)$$

h_2 = za jedan manje od veličine nagornje palačinke

$$h_2 = p_1 - 1$$

(Za jedan manje osigurava da je $h_2 = 0$ u ciljnom stanju $p = (1, 2, \dots, n)$, kako i treba).

– Za nastavak nam treba <u>cijena</u> poteza:

<u>okretanje (gornjih) k palačinki ima cijenu $= k$.</u>

Dopustivost: 2-5

(i) h_1 je dopustiva:

ako k palačinki nije na svom mjestu, onda ih barem k (najgorujih) treba okrenuti, tj. cijena je $\geq k$. (Sve na mjestu $\Rightarrow h_1 = 0$, ok)
(Naravno, tih k na krivim mjestima NE moraju biti na vrhu - onda ih još više treba okrenuti)

(ii) h_2 je dopustiva:

- ako je $h_2 = 0$, tj. $p_1 = 1$, onda je najmanja na svom mjestu, a ostale NE moraju biti (u ciljnom stanju jesu i tad je $h_1 = 0$)

(iii) $h_1 + h_2$ nije dopustiva - može precijeniti!

Jednostavni kontraprimjer = obratni poredak svih palačinki, tj.

$$(n, n-1, \ldots, 2, 1)$$

Tada je h_2 maksimalna moguća (najveća je na vrhu) tj

$$h_2 = n - 1$$

a i h_1 je velika (samo srednja palačinka je na svom mjestu, ako je n neparan

$$h_1 = \begin{cases} n & n \text{ paran} \\ n-1 & n \text{ neparan} \end{cases}$$

Dakle,
$$h_1 + h_2 \geq 2(n-1).$$

Međutim, jedno okretanje svih (cijena je n) dovodi obratni poredak u ciljno stanje.
No, očito je $h_1 + h_2 \geq 2n - 2 > n \iff n > 2$,
pa $h_1 + h_2$ precjenjuje. Isto vrijedi i za $n = 2$,
jer je n paran, pa je $h_1 + h_2 = 2n - 1 = 3 > 2 = n$.

(iv) Pokazali smo da su h_1, h_2 dopustive, tj.

$$h_1(s), h_2(s) \leq h^*(s) \quad \text{za } \forall s \in S$$

pa onda i $\max\{h_1, h_2\}$ <u>je dopustiva</u>.

<u>Konzistentnost</u> – treba gledati nejednakost trokuta za <u>susjedna</u> stanja s i s'

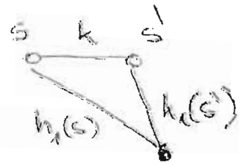
(i) h_1 <u>je konzistentna</u>

Ako iz s u s' stižemo tako da okrenemo točno k palačinki ($s \xrightarrow{k} s'$), cijena tog poteza je k.

No, broj palačinki koje tim potezom <u>mogu</u> doći na svoja <u>prava</u> mjesta je najviše k (tako da svih k na vrhu stignu u $(1, \ldots, k, —)$)

Dakle, $h_1(s') \geq h_1(s) - k$ ili

$$h_1(s) \leq h_1(s') + \underbrace{k}_{= c(s,s')} \quad \checkmark$$



(ii) h_2 <u>nije konzistentna</u>

– Međutim, kontraprimjer treba tražiti za malo veći n.

Naime, za $n=2$ je $h_2(s) \leq 1$, a cijena poteza $= 2$ pa relacija trokuta sigurno vrijedi.

Analogno, za $n=3$ je $h_2(s) \leq 2$ (za $\forall s$), a cijena poteza je 2 ili 3, pa opet vrijedi

– Ideja: uzeti najveću palačinku na vrhu u s, tako da $h_2(s)$ bude <u>velik</u>, napraviti okret s <u>najmanjim</u> k (idealno $k=2$) i u stanju s' dobiti <u>najmanju</u> palačinku na vrhu, tako da je $h_2(s') = 0$.

Dakle: $s = (n, 1, \ldots)$ $\quad h_2(s) = n-1 \quad s \to s'$ s $k=2$

$s' = (\underbrace{1, n}, -\text{ll}-)$ $\quad h_2(s') = 0$

pa je

$$\boxed{h_2(s) > h_2(s') + 2 \quad \text{za } n \geq 4}$$

Digresija: U općem slučaju, čim je $n \geq 4$, nijedna od heuristika h_1, h_2 NE dominira onu drugu.

Primjer za $h_1(s) > h_2(s)$ je $p = (p_1, \dots, p_n)$ s $p_1 \neq 1$ i $p_i \neq i$ za neki $i \geq 2$.

Onda je $h_2(p) = 0$, a $h_1(p) \geq 2$ – bar dvije palačinke nisu na svom mjestu

Primjer za $h_1(s) < h_2(s)$ je $p = (n, 2, 3, \dots, n-1, 1)$

pa je $h_2(p) = n-1$, $h_1(p) = 2$ (samo 1 i n nisu na svom mjestu)

s tim da je $n \geq 4$.

Za $n = 2, 3$ lako se vidi da je $h_1 \geq h_2$ (ako nismo u cilju, bar dvije palačinke nisu na svom mjestu, tj. $h_1(p) \geq 2$ za $\forall p \neq$ identiteta, a $h_2(p) \leq n-1 \leq 2$).

Bolja ~~[illegible]~~ heuristika h_3 – na bazi samo jedne palačinke

Ideja: gledati najveću palačinku koja nije na svom mjestu, jer ona mora otići najdublje ($\Rightarrow$ visoka cijena)

h_3 = veličina najveće palačinke koja nije na svom mjestu

$$\boxed{h_3 = \max \{ i \mid p_i \neq i \}} \qquad (\text{naravno, } 1 \leq i \leq n)$$

(razlika obzirom na h_1 je samo "card" → "max", tj. "broj" u "najveća").

Naravno, ako su sve na mjestu, onda je $h_3 = 0$ (max praznog skupa nije ~~[illegible]~~ definiran, pa ovo treba dodatno definirati).

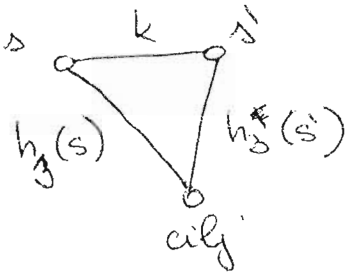
Dopustivost h_3

Ako je $h_3 = k > 0$, onda ta palačinka mora doći na svoje mjesto na "dubini" k, tj. barem jedno okretanje mora okrenuti <u>barem k</u> palačinki (inače se sva okretanja zbivaju iznad). Dakle, cijena je $\geq k$.

Naravno, ako se ta palačinka nalazi dublje od k-tog mjesta, onda cijena mora biti još veća.

Konzistentnost h_3

Uzmimo susjedna stanja s i s', s tim da je cijena prijelaza iz s u s' jednaka k (okret gornjih k palačinki.



Ako je $h_3(s') \geq h_3(s)$, zbog $k \geq 0$ vrijedi i relacija trokuta

$$h_3(s') + k \geq h_3(s).$$

Dakle, treba gledati samo slučaj

$$h_3(s') < h_3(s).$$

No to znači da je palačinka $h_3(s)$ (polazna najveća kriva) morala stići na svoje pravo mjesto $h_3(s)$. Obzirom na to da je promijenila položaj - morala je sudjelovati u okretu, tj. $k \geq h_3(s)$ = njezino NOVO (i pravo) mjesto.
Iz $h_3^*(s') \geq 0$ odmah slijedi

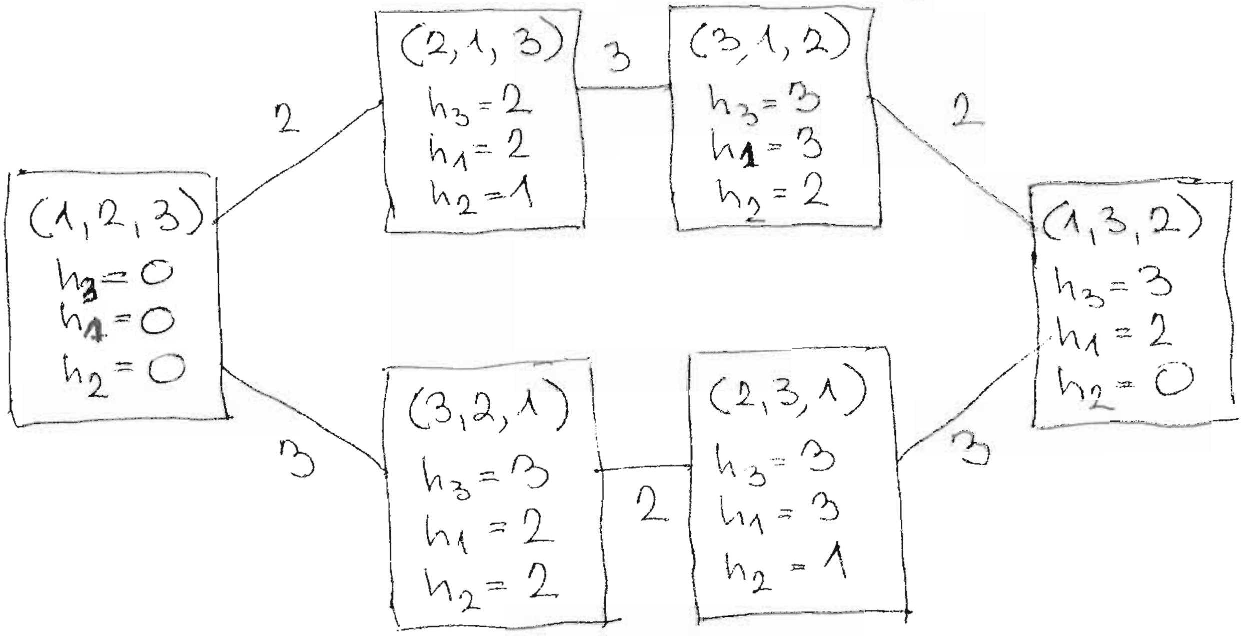
$$h_3(s') + k \geq h_3(s).$$

Dominantnost nad h_1, h_2

$h_3 \geq h_1$: ako je $h_3(s) = k$ <u>najveća</u> na krivom mjestu, onda ih <u>najviše</u> k može biti na krivom mjestu $\Rightarrow h_3(s) \geq h_1(s)$.

Ako je $s = p = (p_1, \ldots, p_n)$ i $h_2(s) = p_1 - 1 > 0$, tj. $p_1 > 1$, onda p_1 <u>nije</u> na svom mjestu, a možda ima još većih. Dakle, mora biti $h_3(s) \geq p_1 > h_2(s)$. Za $h_2(s) = 0$ sigurno je $h_3(s) \geq h_2(s)$.

Vrijednosti ovih heuristika za $n = 3$ (tu znamo da vrijedi dominantnost $h_3 \geq h_1 \geq h_2$)



3. zad - rješenje

~~Stanje~~ Vrijednost u čvoru = dobitak za MAX, odnosno, gubitak za MIN

Optimalna strategija = MAX želi <u>maksimizirati</u> dobitak a MIN želi <u>minimizirati</u> gubitak

Uz pretpostavku optimalne igre oba igrača, <u>minimax</u> vrijednost svakog čvora (stanja) s je

$$\text{MINIMAX}(s) = \begin{cases} \text{VRIJEDNOST IGRE}(s), & \text{ako je } s \text{ završno stanje} \\ \underline{\max}\{\text{MINIMAX}(t) \mid t \in succ(s)\}, & \text{ako je } s = \text{MAX čvor} \\ \underline{\min}\{\text{MINIMAX}(t) \mid t \in succ(s)\}, & \text{ako je } s = \text{MIN čvor} \end{cases}$$

(a)

- Minimax algoritam nalazi ove vrijednosti pretragom u dubinu (DFS)

D = MAX minimax(D) = max{3, 5, 11} = 11

B = MIN minimax(B) = min{2, 6, D=11} = 2

E = MAX minimax(E) = max{8, 4, 12} = 12

C = MIN minimax(C) = min{1, 7, E=12} = 1

A = MAX minimax(A) = max{B=2, C=1} = 2

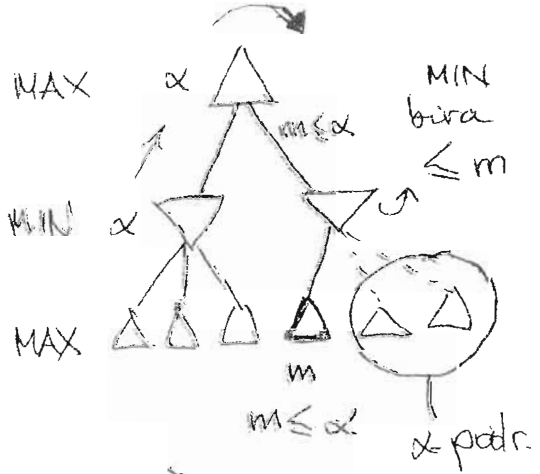
Garantirani dobitak za MAX je 2, pripadni potezi su:
MAX igra B, MIN igra 2.

(b)

Ideja α-β podrezivanja

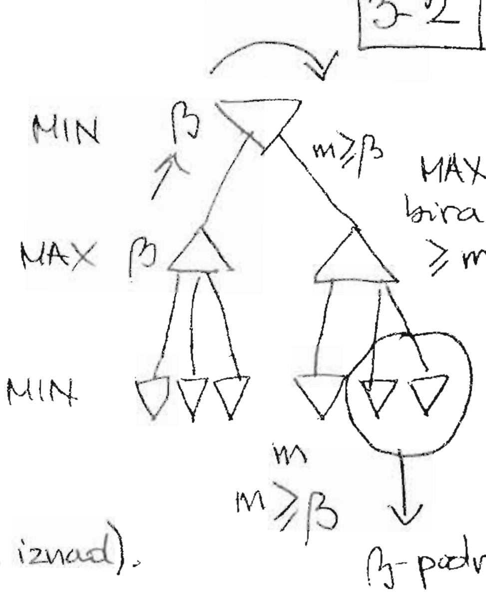
α-podrezivanje za MAX čvor:

- Ako je α <u>najveća</u> dotad nađena vrijednost za MAX;
- u nastavku, ispod MIN vidimo vrijednost $m \leq \alpha$ za MAX
- podrezujemo sve MAX čvorove ISPOD MIN (MAX to <u>neće</u> igrati)
- podrezivanje ide u tom MIN čvoru (za MAX iznad)



β- podrezivanje za MIN čvora:

- Ako je β najmanja dotad nađena vrijednost za MIN,
- u nastavku, ispod MAX vidimo vrijednost $m \geq \beta$ za MIN
- podrezujemo sve ostale MIN čvorove ispod MAX (MIN to neće igrati)
- podrezivanje radi u tom MAX čvoru (za MIN iznad).



Detaljan opis zaključaka α-β podrezivanja kod pretrage minimax algoritmom "u dubinu, slijeva udesno"

A(MAX) → B(MIN) $\quad \alpha = -\infty, \beta = +\infty$

B(MIN) → 2 → B $\quad \alpha = -\infty, \beta = 2 \quad$ (minimax(B) ≤ 2)

B(MIN) → 6 → B $\quad \alpha = -\infty, \beta = 2 \quad$ (— " —)

B(MIN) → D(MAX)

D(MAX) → 3 → D $\quad \alpha = -\infty, \beta = 2 \quad m = 3 \geq \beta = 2$

⇒ minimax(D) ≥ 3

⇒ MIN radi β-podrez. u MAX čvoru D ⟺ sigurno NEĆE igrati D granu.

β-podrezivanje čvorova ∇_5 ; ∇_{11}

~~povratak iz D → B, α=-∞, β=2, minimax(B) ≤ 2~~

~~povratak iz B → A, α=-∞, β=2, minimax(A)~~

povratak iz D → B $\quad \alpha = -\infty, \beta = 2 \quad$ (α ostaje −∞, zbog β podrezivanja!)

minimax(B) = 2

povratak iz B → A $\quad \alpha = 2, \beta = 2$, znamo minimax(A) ≥ 2.
[Gotov obilazak lijeve grane ispod (A)] $\quad$ ‖ $\quad \alpha$

Spuštanje u desnu granu (trenutno $(\alpha,\beta)=(2,2)$) B-3

$A(MAX) \to C(MIN)$

$C(MIN) \to 1 \to C \qquad \alpha=2,\ \beta=2$

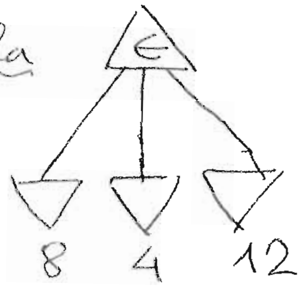
no vrijednost $m = 1 \leq \alpha = 2$

$\Rightarrow$ minimax $(C) \leq 1$ (jer je C MIN čvor)

$\Rightarrow$ "MAX" radi α-podrezivanje u MIN čvoru C

$\Leftrightarrow$ sigurno NEĆE igrati C granu, jer već ima garantiran bolji/veći dobitak.

α-podrezivanje čvorova $\underset{7}{\triangle}$ i stabla



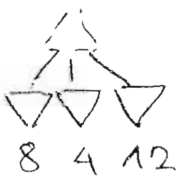
<u>povratak</u>

$C(MIN) \to A$, <u>gotovo</u>!

(c) O preskočenim (podrezanim) završnim čvorovima:

$\underset{5}{\triangledown}\ \underset{11}{\triangledown}$ za β-podrezivanje

$\underset{7}{\triangle}$



za α-podrezivanje

vrijednosti igre mogu biti <u>bilo koje</u> (proizvoljne) a da podrezivanje ostane ISTO (baš zato i <u>smijemo</u> podrezati

- Odluka o <u>podrezivanju</u> je donesena na osnovu <u>vrijednosti (minimax)</u> u <u>prethodnom</u> čvoru na odgovarajućoj dubini – to su $\underset{3}{\triangledown}$ (za β) i $\underset{1}{\triangle}$ (za α)
- Za β: u $\underset{3}{\triangledown}$ smije biti bilo što ≥ 2.

  Za α: u $\underset{1}{\triangle}$ smije biti bilo što ≤ 2.

4. zad. - rješenje

Osnovni pojmovi:

$l_1 \vee l_2 \vee \ldots \vee l_n$

klauzula = disjunkcija literala

gdje je l_i literal, tj. osnovni simbol ili njegova negacija
(simbol → pozitivni literal,
negacija simbola → negativni literal)

konjunktivna normalna forma (CNF) = konjunkcija klauzula

$c_1 \wedge c_2 \wedge \ldots \wedge c_n$, c_i = klauzula (v. gore)

osnovno pravilo rezolucije (jedna od klauzula je jedinična = literal)

$$\frac{l_1 \vee \ldots \vee l_i \vee \ldots \vee l_k \qquad m}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k} \quad \leftarrow \text{rezolventa}$$

gdje su l_i i m **komplementarni** literali (jedan je negacija drugog

"opće" pravilo rezolucije (za dvije klauzule)

rezolventa

$$\frac{l_1 \vee \ldots \vee l_i \vee \ldots \vee l_k \qquad m_1 \vee \ldots \vee m_j \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

gdje su l_i i m_j **komplementarni** literali

Pojednostavljeni zapis - spajanjem blokova koji se "ne krate"

$$\frac{A \vee F \qquad \neg A \vee G}{F \vee G}$$

A literal, F, G klauzule ili bilo što

Hornova klauzula - disjunkcija literala u kojoj je <u>najviše jedan</u> literal pozitivan

definitna klauzula - točno jedan literal pozitivan

ciljna — " — - nema pozitivnih literala

Digresija

Hornove klauzule odgovaraju implikacijama u kojima je premisa = konjunkcija pozitivnih literala, a zaključak je jedan literal (pozitivan) - ako je klauzula definitna, odnosno laž za ciljnu klauzulu.

— definitna klauzula:

$$\underbrace{\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n}_{\text{premisa}} \Rightarrow \beta$$ je ekvivalentno $\neg(\ldots) \lor \beta$ + de Morgan

$$\underbrace{\neg\alpha_1 \lor \neg\alpha_2 \lor \ldots \lor \neg\alpha_n}_{\text{tijelo}} \lor \underbrace{\beta}_{\text{glava}}$$ ← klauzalni oblik

Poseban slučaj kad nema negativnih literala, već samo pozitivni literal odgovara implikaciji

$\text{True} \Rightarrow \beta$ ili samo β

i zove se činjenica

— ciljna klauzula - nema pozitivnih literala, tj. "fali" joj zaključak — uzima se kao False

$$\alpha_1 \land \alpha_2 \land \ldots \land \alpha_n \Rightarrow \text{False} \quad (\text{neutral za } \lor)$$

$$\neg\alpha_1 \lor \neg\alpha_2 \lor \ldots \lor \neg\alpha_n$$ ← klauzalni oblik

Ide se na obaranje premise, što odgovara rezoluciji opovrgavanjem, tj. dokazu da premise (zajedno s negiranom željenom tvrdnjom daju proturječnu ili neispunjivu formulu).

- Rezolucija unaprijed (direktnim dokazivanjem) da

$F_1, \ldots, F_n \models G$ (ili $KB \models \alpha$)

nije potpuna. Umjesto toga, koristimo tzv. rezoluciju opovrgavanjem - negiramo cilj, dodamo to u premise i dokazujemo da je

$F_1, \ldots, F_n, \neg G$

proturječna formula (rezolucija daje praznu klauzulu kao rezolventu).

Zadana rečenica (formula)

lijeva strana (LS) desna strana (DS)

$$\left[(\text{Hrana} \Rightarrow \text{Zabava}) \vee (\text{Piće} \Rightarrow \text{Zabava})\right] \Rightarrow \left[(\text{Hrana} \wedge \text{Piće}) \Rightarrow \text{Zabava}\right]$$

ima tri simbola Hrana, Piće, Zabava, koje skraćeno pišemo H, P, Z.

(a) <u>Enumeracija</u> s 3 simbola ima 8 mogućnosti. Tablica

H	P	Z	$H \Rightarrow Z$	$P \Rightarrow Z$	LS	$H \wedge P$	DS $(H \wedge P) \Rightarrow Z$	$LS \Rightarrow DS$
0	0	0	1	1	1	0	1	1
0	0	1	1	1	1	0	1	1
0	1	0	1	0	1	0	1	1
0	1	1	1	1	1	0	1	1
1	0	0	0	1	1	0	1	1
1	0	1	1	1	1	0	1	1
1	1	0	0	0	0	1	0	1
1	1	1	1	1	1	1	1	1

isto, tj. <u>ekvivalentno</u>

Dakle, formula je valjana. Čak jače, valjana je i formula $LS \Leftrightarrow DS$, uz $LS = (H \Rightarrow Z) \vee (P \Rightarrow Z)$, $DS = (H \wedge P) \Rightarrow Z$

(b) <u>Pretvorba u CNF</u>:

– Lijeva strana: $(H \Rightarrow Z) \vee (P \Rightarrow Z)$

eliminacija $\Rightarrow$: $(\neg H \vee Z) \vee (\neg P \vee Z)$

eliminacija zagrada: $\neg H \vee Z \vee \neg P \vee Z$

spajanje i faktorizacija: ($Z \vee Z = Z$) $\neg H \vee \neg P \vee Z$ <u>jedna klauzula</u>

– Desna strana: $(H \wedge P) \Rightarrow Z$

eliminacija $\Rightarrow$: $\neg(H \wedge P) \vee Z$

potiskivanje $\neg$ (deMorgan) $(\neg H \vee \neg P) \vee Z$

eliminacija zagrada $\neg H \vee \neg P \vee Z$ <u>jedna klauzula</u>

Obje strane daju ISTU CNF, pa polazna rečenica ima oblik $A \Rightarrow A$ i <u>valjana</u> je (kao i $A \Leftrightarrow A$) za <u>bilo koji</u> A.

Tvrdimo da je formula <u>valjana</u> (bez ikakvih premisa). Za dokaz opovrgavanjem, treba pokazati da je njezina <u>negacija neispunjiva</u>.

Dakle, negiramo ju, pretvorimo to u CNF i koristimo rezoluciju da dokažemo "kontradikciju", tj moramo dobiti <u>praznu rezolventu</u>.

Negacija formule:

$$\neg\{[(H\Rightarrow Z)\vee(P\Rightarrow Z)]\Rightarrow[(H\wedge P)\Rightarrow Z]\}$$

Koristimo $\neg(a\Rightarrow b)\Leftrightarrow\neg\neg(a\wedge\neg b)\Leftrightarrow a\wedge\neg b$ $(a=LS,\ b=DS)$

Nakon eliminacije implikacije izlazi

$$[(H\Rightarrow Z)\vee(P\Rightarrow Z)]\wedge\neg[(H\wedge P)\Rightarrow Z] \qquad LS\wedge\neg DS$$

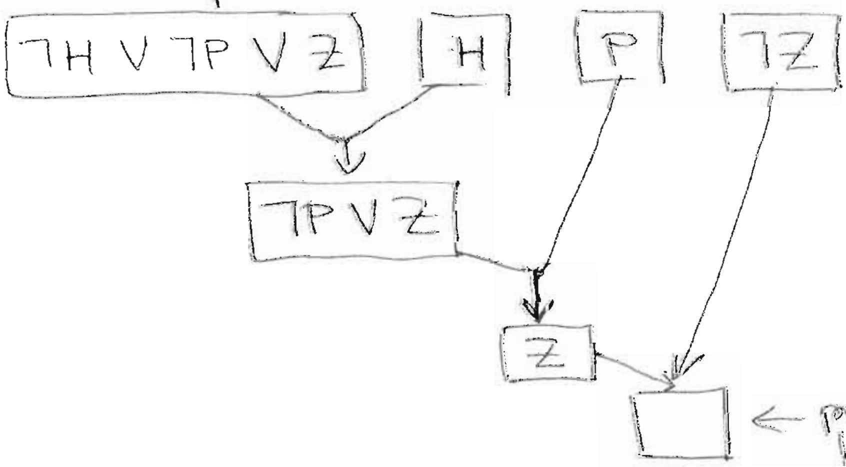
Onda iskoristimo već nađene CNF za lijevu (LS) i desnu stranu (DS). Izlazi

$$(\neg H\vee\neg P\vee Z)\wedge\neg[\neg H\vee\neg P\vee Z]$$

Na kraju, po de Morganu sredimo desni dio, uz eliminaciju dvostruke negacije

CNF: $\boxed{(\neg H\vee\neg P\vee Z)\wedge H\wedge P\wedge\neg Z}$ 4 klauzule, zadnje 3 su jedinične.

Svaku od zadnje 3 jedinične klauzule možemo razlučiti (rezolvirati) s pripadnim komplementarnim literalom u prvoj. Korak po korak, redom kako piše (⟶)



← prazna klauzula kao rezolventa.

Dakle, polazna formula je <u>valjana</u>.

5. zad. - rješenje

U terminima logike predikata (FOL) koja ima objekte, relacije i funkcije, zaključivanje nastojimo svesti na zaključivanje u logici sudova - tzv. postupak propozicioniranja (ili propozicionalizacije)

Za to koristimo tzv. supstitucije
= zamjene varijabli u rečenicama, osnovnim simbolima (objektima) u tzv. bazi znanja (KB).

- Neformalno, unifikacija rečenica p i q je traženje supstitucija tako supstitucijom dobijemo iste rečenice.

Neka je θ neka supstitucija (zamjena varijabli osnovnim objektima).
Supstitucija θ je unifikacija rečenica p i q, u oznaci

$$\text{UNIFY}(p, q) = \theta$$

ako nakon supstitucije θ u p i q dobivamo ISTE rečenice

$$\boxed{p\theta = q\theta, \text{ ili } \text{SUBST}(\theta, p) = \text{SUBST}(\theta, q)}$$

- Pravilo zaključivanja GMP opisuje zaključivanje na osnovu supstitucija:
Neformalno: supstituiramo θ u sve premise
i premise povlače neki zaključak
onda slijedi zaključak sa supstitucijom θ

Za atomarne rečenice p_i, p_i' i q, ako postoji supstitucija θ takva da je

$$\boxed{p_i'\theta = p_i\theta \text{ odn. } \text{SUBST}(\theta, p_i') = \text{SUBST}(\theta, p_i)} \quad K_i$$

za sve i, onda

$$\boxed{\frac{p_1', p_2', \ldots, p_n', \quad (p_1 \wedge p_2 \wedge \ldots \wedge p_n \Rightarrow q)}{q\theta = \text{SUBST}(\theta, q)}}$$

~~supstitucija~~ unifikacija svih premisa povlači supstituirani zaključak.

Sintaktički i semantički korektni prijevod rečenice
"Svaki pas koji voli nekog od svoje braće je sretan".

U ovom kontekstu, "nekog" može značiti da pas uopće nema braće (odnosno, možemo ga čak smatrati svojim vlastitim bratom - nije bitno), ali i da može imati puno braće.

Dakle, pojam "Brat" sigurno NE smije biti funkcija, jer bi to značilo da pas ima točno jednog brata, a to nije korektna semantika polazne rečenice.

(1) nije korektna - koristi Brat kao funkciju

(2) $\boxed{\text{je}}$ korektna
koristi egzistencijalni kvantifikator za postojanje Brata od x i kojeg x Voli, ali lokalno

(3) $\boxed{\text{je}}$ korektna
koristi globalni univerzalni kvantifikator za brata(y), ali su svi predikati u konjunkciji unutar premise.

(4) nije korektna - klasična greška
egzistencija $\exists$ i $\Rightarrow$ umjesto $\wedge$

(5) nije korektna - klasična greška
univerzalnost $\forall$ (po x) i $\wedge$ umjesto $\Rightarrow$

Jednostavni (ne strogo formalni) argument za ekvivalenciju (2) i (3):

- Ako u (2) zaista postoji y (ovisan o danom x) koji ispunjava Brat(y, x) $\wedge$ Voli(x, y), onda taj isti y ispunjava i taj isti dio u (3)
- Obratno, krenemo od (3) i ako je (Brat(y,x) $\wedge$ Voli(x,y)) istina za neke x, y, onda $\exists$y u (2) da je i (2) ispunjena.

Za zaključivanje u FOL (posebno metodama FC, BC i u Prologu) pogodniji je oblik tvrdnje (3) u kojem su SVI kvantifikatori UNIVERZALNI (ne treba Skolemizacija)

Dakle, krećemo (kao i kod pukovnika Westa) od zapisa (3)

$$\boxed{\forall x,y \;\; Pas(x) \land Brat(y,x) \land Voli(x,y) \Rightarrow Sretan(x)}$$

Ovo je jedini "aksiom" za zaključivanje!

Sad dodamo navedene 3 činjenice

Pas(Fifi), Brat(Rex, Fifi), Voli(Fifi, Rex)

No, to je ovdje trivijalno, jer imamo samo JEDNU implikaciju. Ulančavanje unaprijed (FC) ide od poznatih činjenica.

(1) Pas(Fifi) zadovoljava Pas(x) s {x/Fifi}
(i nema druge supstitucije koja bi to zadovoljila)

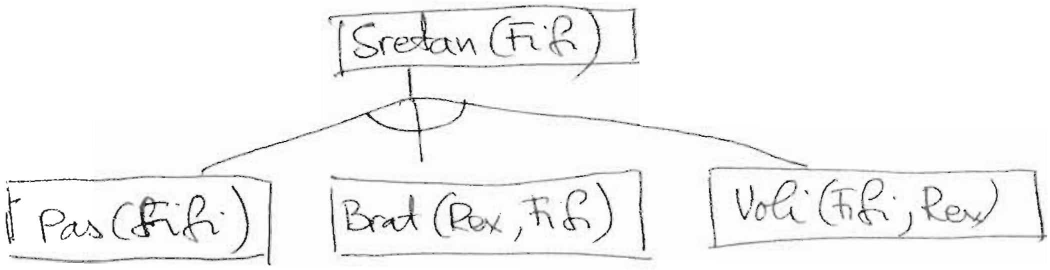
(2) Brat(Rex, Fifi) zadovoljava Brat(y, x) s {x/Fifi, y/Rex} – kompatibilno s prvim zaključkom

(3) Voli(Fifi, Rex) zadovoljava Voli(x, y) opet s {x/Fifi, y/Rex} – kompatibilno s prethodna dva zaključka

⇒ Sve tri premise smo zadovoljili s {x/Fifi, y/Rex} pa slijedi zaključak Sretan(x) uz {x/Fifi}, tj

$$\boxed{Sretan(Fifi)}$$

Grafički, FC ima samo prvu iteraciju



Vrlo jednostavno ide i ulančavanje unatrag (BC).

Upit Sretan(Fifi) može slijediti samo iz one jedne implikacije

$$Pas(x) \wedge Brat(y,x) \wedge Voli(x,y) \Rightarrow Sretan(x)$$

pa dobivamo vezivanje = supstituciju $\{x/Fifi\}$

Nakon te supstitucije zaključak se svodi na

$$Pas(Fifi) \wedge Brat(y,Fifi) \wedge Voli(Fifi,y) \Rightarrow Sretan(Fifi)$$

1. Unštavanje u Pas(x) daje poznatu činjenicu Pas(Fifi), tj. prva premisa je dokazana.

2. Gledamo Brat(y, Fifi). Poznata činjenica u bazi znanja Brat(Rex, Fifi) daje supstituciju $\{y/Rex\}$, čime je i druga premisa dokazana.

3. Ostaje Voli(x,y) s $\{x/Fifi, y/Rex\}$, tj. Voli(Fifi, Rex). I to je poznata činjenica u bazi, pa vrijedi i zadnja premisa,

$\Rightarrow$ cilj je dokazan $\boxed{Sretan(Fifi)}$

6. zad. – rješenje (v. zad6ok.pro)

U Prologu se atomi pišu malim slovima, a varijable velikim.

Prvi aksiom iz 5. zad:

sretan(X) :- pas(X), brat(Y,X), voli(X,Y).

Kod drugog aksioma, iz 6. zad., treba paziti, jer Prolog koristi ulančavanje unatrag s pretragom u dubinu, i to slijeva nadesno.

Zato NE VALJA izravni prijevod

pas(Y) :- pas(X), brat(Y,X).

jer već upit sretan(fifi). izaziva beskonačnu rekurziju.

Ispravni zapis je:

pas(Y) :- brat(Y,X), pas(X).

tako da se rekurzija makne s lijeve strane (front) na desnu (tail).

Činjenice:

pas(fifi).
brat(rex, fifi).
voli(fifi, rex).

Upiti i odgovori:

(a)	sretan(fifi).	odgovor:	yes
(b)	sretan(X).	odgovor:	X = fifi
(c)	pas(X).	odgovori:	X = fifi (; i a) X = rex