Pregled zadataka za završni ispit iz Formalnih metoda u oblikovanju sustava

FPORS pismeni 16.06.2003.

1. Što mora svaki "always" blok u Verilog programu imati kako se ne bi vrtio u beskonačnoj petlji?

Mora imati kašnjenje.
n
@ (event)
wait ()

2. Ekspanzijom s lijeve strane (Shannon) dokaži:
$f(\overline{x}+\overline{y})+f(xy)=f(0)+f(1)$
Pomoć: $(x'+y')=g, xy=h$

Ekspanzija po Shannonu: $f(x)=xf_x+\overline{x}f_{\overline{x}}$

$f(\overline{x}+\overline{y})+f(xy)$

$= x[f(0+\overline{y})+f(y)]+\overline{x}[f(1+\overline{y})+f(0)]$ $\quad 1+\overline{y}=1$
$= x[f(\overline{y})+f(y)]+\overline{x}[f(0)+f(1)]$ $\quad f(\overline{y})+f(y) \Leftrightarrow f(0)+f(1)$
$= x[f(0)+f(1)]+\overline{x}[f(0)+f(1)]$ $\quad 1+\overline{y}=1$
$= [f(0)+f(1)][x+\overline{x}]$ $\quad x+\overline{x}=1$
$= f(0)+f(1)$

3. Navedi dvije osnovne razlike između CTL i LTL vremenske logike

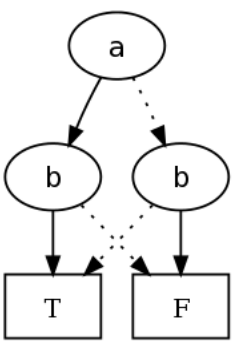
Kod CTL kvantifikatori su A (Always) ili E (Exists, Eventually).
Kod LTL kvantifikator A je implicitan, tj. nema kvantifikatora E.
CTL predstavlja stablo (*branching time*), a LTL linearno vrijeme (*linear time*).
Jedan nije podskup drugog. Zajedno su dio CTL*.

LTL dozvoljava ugnježđivanje booleovih vezica i modaliteta.
CTL ne dozvoljava booleove kombinacije formula puta ni ugnježđivanje modaliteta.

4. Zadan je dvobitni komparator (izlaz=1 za jednake vrijednosti ulaza).
Napiši logičku funkciju za navedeni sklop.
Nacrtaj odgovarajući reducirani BDD dijagram uz uređenost varijabli: a1, b1, a2, b2

Logička funkcija izlaza (Z) za 1-bit komparator (na slici):
$K=(a \Leftrightarrow b)$

Dvobitni komparator se razlikuje (ima 4 ulaza).
http://en.wikipedia.org/wiki/Digital_comparator



5. Uz koje argumente *ite* algoritam izračunava logičku AND funkciju $(F \wedge G) = ite(?, ?, ?)$ te koja je složenost izračunavanja AND dviju logičkih funkcija predstavljenih BDD dijagramima sa M odnosno N čvorova, *ite* algoritmom, pod pretpostavkom korištenja izračunske tablice (*computed table*) i beskonačne memorije?

$(F \wedge G) = ite(F, G, 0)$

Za *ite* funkciju s dva operanda (npr. logičke funkcije AND, OR, XOR...) složenost je $O(|f| \times |g|)$ u najgorem slučaju, pri čemu je $|f|$ broj čvorova u ROBDD-u za *f*.
Za M i N čvorova, složenost je tada $O(M \times N)$.

6. Pod pretpostavkama:
- skupovi stanja predstavljeni su BDD dijagramima
- postoji algoritam za izračunavanje skupa stanja: $BDD\ EX(BDD\ F)$
- poznata je CTL ekvivalencija $EG\ \varphi \equiv \varphi \wedge EX\ EG\ \varphi$

Skiciraj algoritam za izračunavanje skupa stanja: $BDD\ EG(BDD\ F)$

FPORS pismeni, 13.05.2003.

7. U Verilog programskom primjeru analiziranom na vježbama postoji naredba poput: `assign var1 = $ND(0, 1);` Objasni ovu naredbu.

Tom se naredbom modelira nedeterminizam (ND), odnosno varijabla `var1` nedeterministički poprima vrijednost 0 ili 1.

8. Objasni razliku između logičke posljedice i ekvivalencije između dviju formula.

Dvije formule A i B su semantički ekvivalentne ($A \equiv B$ ili $A \Leftrightarrow B$) akko vrijede (istinite su) logičke posljedice: $A| = B$ i $B| = A$

9. Navedi koje su ispravno napisane CTL formule (*well-formed*), a koje nisu:

A [p U EFr]	*ispravna*
AEFr	*neispravna*
FG r	*neispravna*
AF [(rUQ)^(pUr)	*neispravna*

10. Napiši u CTL formalizmu:
"Štogod da se dogodi, izvjestan proces će konačno završiti u trajnoj blokadi (*deadlock*)"

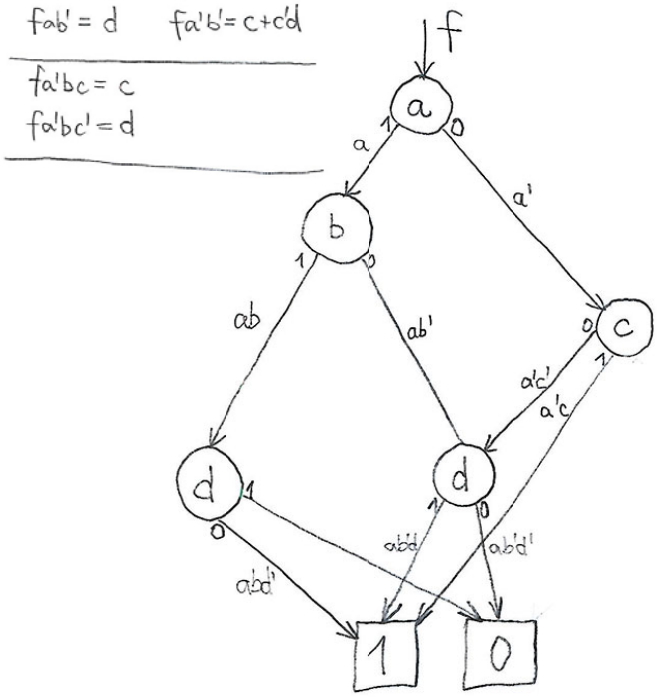
AG (AF deadlock)

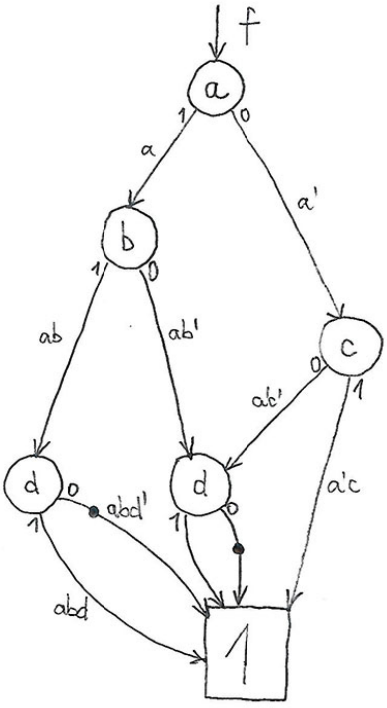
11. Za funkciju $f = abd' + ab'd + a'c + a'c'd$ uz uređenje $a < b < c < d$ nacrtaj ROBDD s komplementarnim lukovima

$f_a = bd' + b'd$
$f_{a'} = c + c'd$

$f_{ab} = d'$ $f_{a'b} = c + c'd$
$f_{ab'} = d$ $f_{a'b'} = c + c'd$

$f_{a'bc} = c$
$f_{a'bc'} = d$





12. Uz koje argumente *ite* algoritam izračunava:
$NAND(f,g) = ite(?,?,?)$ $XOR(f,g) = ite(?,?,?)$

$NAND(f,g) = ite(f,g',1)$
$XOR(f,g) = ite(f,g',g)$

13. Zanima nas da li vrijedi $F \Rightarrow G$ (pri čemu je $\Rightarrow$ znak implikacije). To je ekvivalentno provjeri da li $ite(F,G,1) = 1$. Izračunavanje se može izvesti algoritmima $ite(F,G,1)$ i $ite\_constant(F,G,1)$. Obrazloži zašto je efikasnije koristiti $ite\_constant$?

(F ⇒ G) = (F' ∨ G) = ite(F, G, 1) = 1 ?

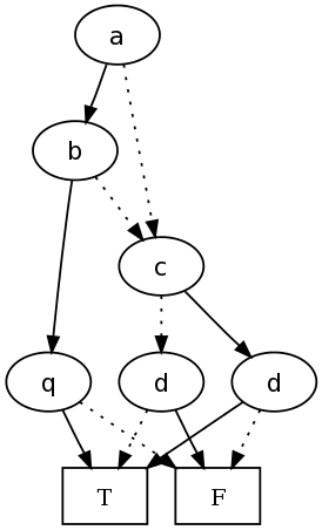
To se može izvesti izvođenjem standardnog ***ite*** algoritma i provjerom rezultata. No postoji i brži postupak:

```
ITE_constant(F, G, H){
    if (trivial case) {
        return result (0,1, or NC);
    } else if (cache table has entry for (F, G, H))
        return result;
```

npr. za *ite(F,G,1)* odmah se vraća *NC* ako *H≠1*

Algoritam je brži od standardnog **samo ako funkcija nije konstanta** (raniji izlazak). Da bi funkcija bila konstanta, *then* i *else* strana u svakoj iteraciji moraju biti jednake.

14. Neka je rezultat izračunavanja: $ite(F, G, H) = (a, (b, q, (c, d, d')), (c, d, d'))$
Nacrtaj odgovarajući ROBDD. Pojednostavi BDD dijagram uz pretpostavku da se lukovima mogu dodati obilježja komplementa.



15. Objasni riječima: $M, s \models (\varphi \Rightarrow \omega)$
gdje je $\Rightarrow$ znak implikacije, $M = (S, R, L)$ Kripke struktura, stanje $s \in S$, te φ i ω CTL formule.

U modelu M, za stanje $s \in S$ vrijedi: ako je φ istinita formula (u CTL logici), onda je istinita i ω.

16. Nacrtaj reducirani BDD dijagram za funkciju sume kod digitalnog sklopa punog zbrajala (*full adder*).

http://en.wikipedia.org/wiki/Adder_(electronics)#Full_adder

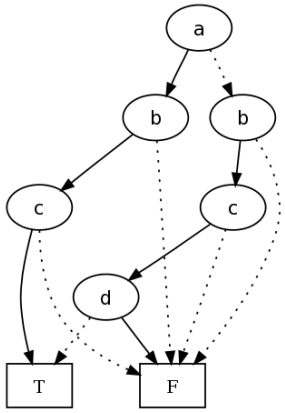
Puno zbrajalo sadrži 2 bita na ulazu i 1 carry bit.

17. Jedinstvena tablica (*unique table*) i izračunska tablica (*computed table*) koriste "hash" funkciju koja daje ključ (lokaciju u tablici) za pohranu podataka. Što predstavlja ključ u jedinstvenoj tablici, a što u izračunskoj tablici?

Jedinstvena tablica – ključ je pokazivač na čvorove ROBDD-a (opis čvora je struktura podataka koja sadrži indeks varijable, indeks *then* strane, indeks *else* strane, pokazivač na sljedeću strukturu u kolizijskom lancu).

Izračunska tablica – ključ je pokazivač na rezultate *ite* funkcije

18. Nacrtaj BDD dijagrame za funkciju: $f = abc + bcd'$



19. Objasni riječima CTL formulu: $AG(p \rightarrow AX\ AG(\neg q\ \vee\ A[\neg r\ U\ t]))$

Uvijek vrijedi (AG, Always Globally), ako p vrijedi (implikacija), tada u svakom sljedećem stanju (AX, Always neXt) uvijek vrijedi (AG), ili $\neg q$, ili $\neg r$ dok (U, until) t ne postane istinit.

20. Objasni razliku između:
"q je logička posljedica p"
"q je ekvivalentno p"

21. Napiši CTL formulu koja izražava
"Iz svakog stanja moguće je doći do početnog stanja."

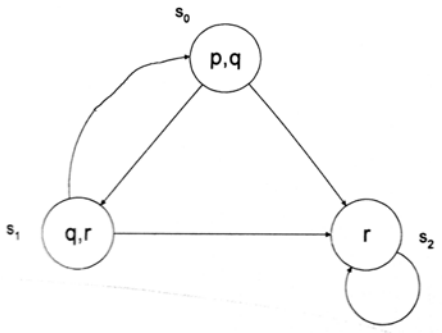
AG(EF pocetno)

FPORS pismeni, 13.05.2008.

22. Potrebno je izgraditi jedan Verilog modul koji zadovoljava sljedeće CTL specifikacije:
$AG(p \rightarrow EXq)$
$AG\big(q \rightarrow EX(EG\ r)\big)$
Zadovoljava li izgrađeni modul specifikaciju $AG(p \rightarrow EF\ r)$

23. Navedi tri načina pretvorbe Mooreovog automata u Kripke strukturu. Za svaki od načina procijeni promjenu broja stanja Kripke strukture dodavanjem nedeterminističkog ulaza veličine jednog bita Mooreovom automatu. Broj stanja Mooreovog automata se ne mijenja dodavanjem novog ulaza.

24. Skiciraj algoritam za izračunavanje $Q(EG\ f)$ bez korištenja BDD struktura. Poznat je algoritam za izračunavanje $Q(EX\ f)$. Primijeniti algoritam na primjeru sa slike za sve atomske propozicije (p, q, r). S0 je početno stanje sustava.



25. Odredi složenost za SAT, te za ispitivanje tautologije za logičke funkcije prikazane ROBDD dijagramima.

26. Ekspanzijom (Shannon) lijeve strane po varijabli x, dokaži:
$f(x+y)+f(xy)=f(x)+f(y)$

Ekspanzija po Shannonu: $f(x)=xf_x+\overline{x}f_{\overline{x}}$

$f(x+y)+f(xy)$

$= x[f(1+y)+f(y)]+\overline{x}[f(y)+f(0)]$ $\quad y+1=1,\ x\overline{x}=0$
$= xf(1)+xf(y)+\overline{x}f(y)+\overline{x}f(0)$
$= xf(1)+\overline{x}f(0)+xf(y)+\overline{x}f(y)$ $\quad f(x)=xf(1)+\overline{x}f(0)$
$= f(x)+f(y)(x+\overline{x})$ $\quad x+\overline{x}=1$
$= f(x)+f(y)$

27. Može li se SMV-om i Verilogom modelirati nedeterminizam i na koji se način modelira (ako ga je moguće modelirati)?

Nedeterminizam u SMV i Verilog se **može** modelirati.

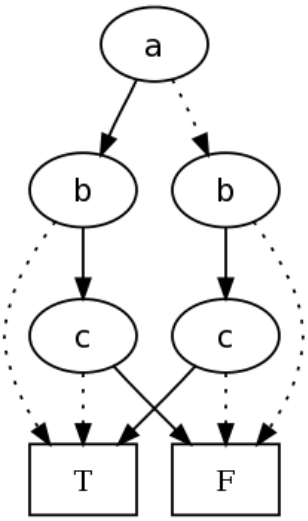
Verilog – eksplicitni nedeterminizam `x=$ND(0,1)` ili implicitni nedeterminziam (izostavljanje prijelaza neke varijable implicira da ta varijabla može poprimiti bilo koju vrijednost)

Kod SMV-a nekoj se varijabli također može dodijeliti nedeterministička vrijednost iz skupa vrijednosti. Primjer: `signal := {a,b,c,d}`

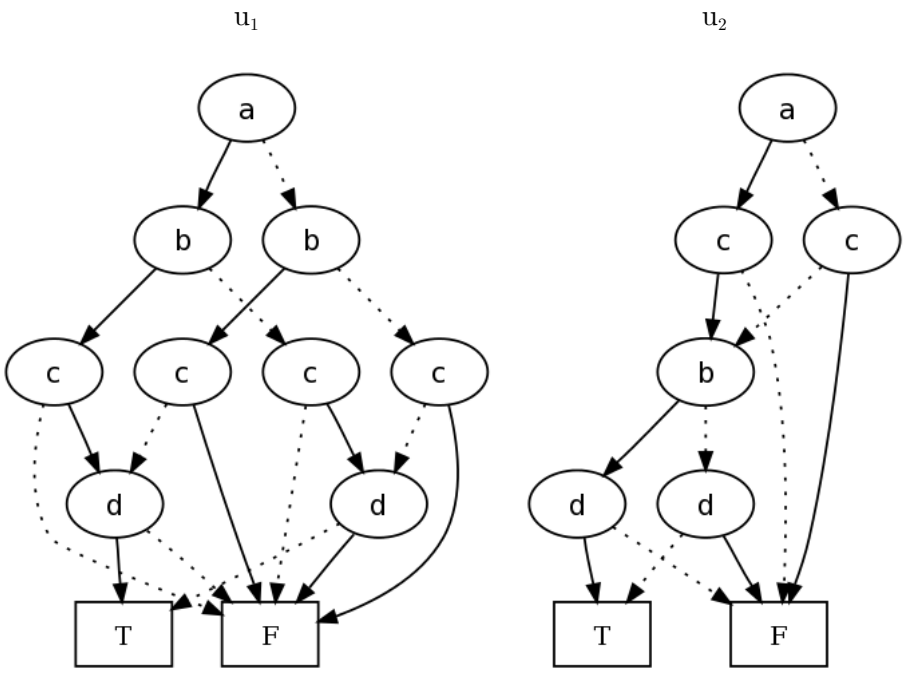
28. Objasnite pojam nepristranosti (*fairness*) u provjeri modela i sintaksu kojom se uključuje u SMV-u.

FAIRNESS running / JUSTICE running
Ograničenja pravednosti rješavaju probleme blokiranja kod ulaska u kritični odsječak.

29. Izgradite BDD (bez komplementarnih lukova) za funkciju:
$f = a\ XOR\ (b\ AND\ c)$
uz uređenje varijabli $a < b < c$.



30. Izgradite ROBDD (bez komplementarnih lukova) za funkciju:
$f = (a \Leftrightarrow c) \wedge (b \Leftrightarrow d)$
za uređaje varijabli
$u_1 = a < b < c < d$ i $u_2 = a < c < b < d$ te odredite koji je uređaj bolji.



Uređaj u_2 je bolji jer je poredak varijabli a<c i b<d sukladan ekvivalencijama $a \Leftrightarrow c$ i $b \Leftrightarrow d$, što producira ROBDD s manjim brojem čvorova (6 naspram 9), jer se ekvivalencije mogu najprije za sebe riješiti, a zatim graf povezati u cjelinu, što nije slučaj kod u_1.

31. Objasnite razliku u semantici oznaka |= i |– kod propozicijske logike i predikatne logike prvog reda. Navedite što su kompletnost i ispravnost logike, te u kojim logikama vrijede.

$\Gamma \vdash_L \omega_i$ Sekvencija formula ili pojedina formula ω_i je teorem (dokaz, dedukcija) iz skupa formula Γ, ako se nalazi u tom skupu, ili se iz njega može izvesti korištenjem pravila zaključivanja L.

$\Gamma \models \omega$ Skup formula Γ implicira ili povlači (*entails*) formulu ω, ako je svaki model od Γ ujedno i model od ω. Formula ω je tada logička posljedica skupa formula Γ.

$\varphi \models \psi$ akko je $(\varphi \wedge \neg\psi)$ nezadovoljiva

Logika je **ispravna** ako je svaka pravilima dokazana formula ujedno i logička posljedica skupa Γ.

Logika je **kompletna** ako je svaku logičku posljedicu skupa Γ moguće dokazati pravilima L.

32. Skiciraj sekvence stanja koja odgovara obilježju u LTL logici:
    a) GF p, tj. beskonačno često (*infinitely often* p)
    b) FG p, tj. konačno globalno p (*finally globally* p, *almost everywhere* p)

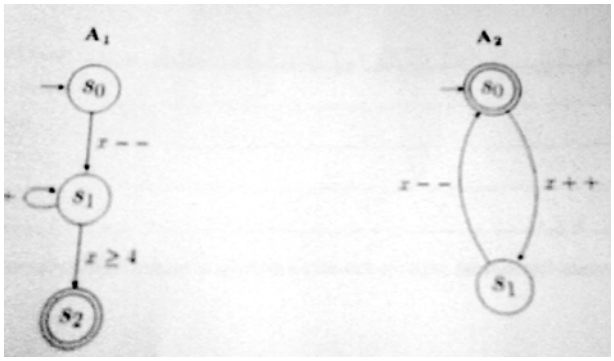
33. Kojoj skupini vremenskih logika (CTL, LTL, CTL*) pripada formula E(GF p). Objasni zašto?

Formula E(GF p) pripada skupini CTL*.
- u skupini CTL kvantifikator A ili E mora biti eksplicitan (GF p pripada LTL).
- u skupini LTL kvantifikator A je implicitan, tj. ne postoji kvantifikator E.

34. Izrazi CTL formule:
    a) AF p
    b) EG q
    pomoću AU operatora.

A (**T** U p)
A (**T** U (AG p))

35. Na slici su prikazana dva FSA, A1 i A2 za koje treba:



a) napisati pripadne naredbe za Promela procese (varijabla x neka je globalna varijabla tipa *byte* početne vrijednosti *x=1*)

```
proctype A1 () {
```

```
proctype A2 () {
```

b) nacrtati asinkroni produkt automata C_{FSA}=A1=A2 te odrediti sve komponente C_{FSA}.=(C.S, C.s_0, C.L, C.T, C.F)

c) za LTL formulu $\Diamond\Box p$ napisati pripadne Promela instrukcije te nacrtati Buchi automat

never {

d) Postoji li sekvenca ekspandiranog produkta koja je beskonačne duljine? Što se pri tome dešava s vrijednostima varijable x?

36. Zadani su Promela procesi Thread1 i Thread2.

```
byte x,t1,t2;
#define dif(a,b) ((a>=b) -> a-b; b-a) /* |a-b| a-b uvijek pozitivno */

proctype Thread1() {
    do
            ::      t1=x;
                    t2=x;
                    x=dif(t1,t2);
    od
}



proctype Thread2() {
    do
            ::      t1=x;
                    t2=x;
                    x=t1=t2;
    od
}

init
{
    x = 1;
    run Thread2();
    run Thread1();
}
```

a) Nacrtajte konačne automate za oba Promela procesa

b) Opisati postupak kako bez primjene LTL formule odrediti istinitost sljedeće tvrdnje: "Tijekom izvođenja varijabla x može poprimiti vrijednost $x > 255$".

Tvrdnja je istinita. Moguć je konstantan odabir procesa Thread2 dok x ne postane 255 ili više.

c) Opisati postupak kako primjenom LTL formule odrediti istinitost sljedeće tvrdnje te postupak pronalaženja protuprimjera
"Tijekom izvođenja varijabla x može poprimiti vrijednost $x > 255$".

37. Ekspanzijom po Shannonu dokaži:
$f(x+y) + f(xy) = f(x) + f(y)$
Ekspanziju provedi za prvu funkciju $f(x+y)$ po $(x+y)$, a za drugu funkciju $f(xy)$ po (xy). Ekspanzija se naime može provesti ne samo po nekoj varijabli nego i po funkciji, tj. vrijedi: $g(h(x)) = h(x)g(1) = h'(x)g(0)$

38. Odredi parametre *ite* funkcije za izračunavanje sljedećih logičkih funkcija:
a) AND(f, g)
b) OR(f, g)
c) NAND(f, g)
d) XOR(f, g)

Name:	Notation:	ITE form:
not	$\overline{F}$	ITE (F, 0, 1)
and	$F \cdot G$	ITE (F, G, 0)
xor	$F \oplus G$	ITE (F, $\overline{G}$, G)
or	$F \vee G$	ITE (F, 1, G)
nor	$\overline{F \vee G}$	ITE (F, 0, $\overline{G}$)
equiv	$F \leftrightarrow G$	ITE (F, G, $\overline{G}$)
implies	$F \rightarrow G$	ITE (F, G, 1)
nand	$\overline{F \cdot G}$	ITE (F, $\overline{G}$, 1)

39. Navedi **dvije** temeljne razlike između jedinstvene i izračunske tablice u postupku izgradnje ROBDD-a.

Jedinstvena (*unique*) tablica: zapisuje pokazivače na čvorove ROBDD-a; njome se ostvaruje kanonski zapis (za neki čvor jedan i samo jedan pokazivač); kod *hash* kolizija formira se kolizijski lanac;

Izračunska (*computed*) tablica: zapisuje pokazivače na rezultate *ite* funkcije; odgovara na pitanje "Postoji li jedinstveni čvor *(v,g,h)*?", odnosno "Jesmo li već izračunali nešto?"; ne koristi kolizijski lanac (kod kolizije stariji podatak se odbacuje);

40. Skiciraj algoritam za izračun *ite(f, g, h)* funkcije uz uporabu jedinstvene (*unique*) i izračunske (*computed*) tablice.

Terminal cases: $(0, g, f) = (1, f, g) = f$
$ite(f, g, g) = g$

```
ite(f,g,h)
    if(terminal case){
        return result;
    } else if (computed-table has entry (f,g,h){
        return result;
    }else{
        let v be the top variable of (f,g,h);
        f~ ← ite(f_v, g_v, h_v);
        g~ ← ite(f_v̄, g_v̄, h_v̄);
        if (f~ equals g~) return g~;
        R ←find_or_add_unique_table(v, f~, g~);
        insert_computed_table({f,g,h}, R);
        return R; } }
```

možda je elementarno, ako nije pogledaj ća li je to već računato

rekurzivno izračunavanje THEN i ELSE strane

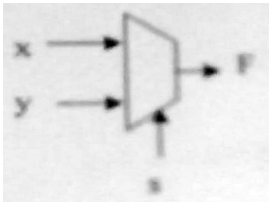
postoji ili novi čvor R

dodaj argumente i rezultat R u izračunsku tablicu

28

41. Multipleksori

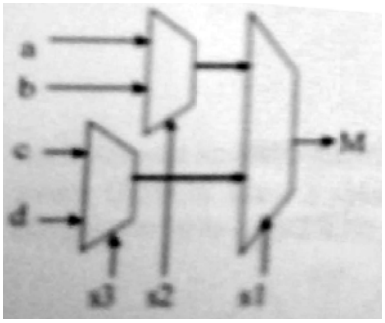
a) Napiši logičku funkciju izlaza F za osnovni digitalni multipleksor na slici:

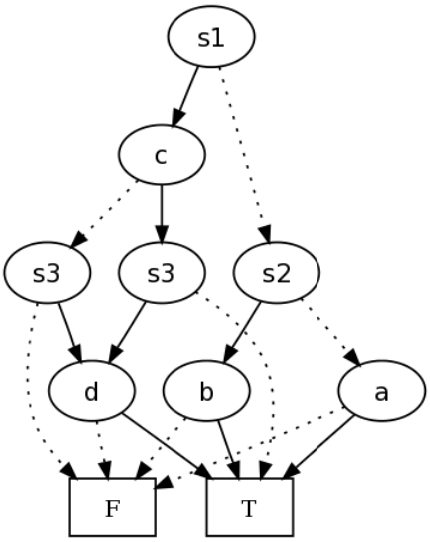


$F = \overline{s}x + sy$

Varijabla s je selektor – ako je s=*0* na izlaz F postavlja se x, u protivnom se postavlja y.

b) Napiši logičku funkciju izlaza M i nacrtaj ROBDD za sklop tri multipleksora prema slici uz redoslijed varijabli $s1, s2, a, b, c, s3, d$. Nije potrebno koristiti komplementarne lukove.





Izlazna funkcija M:

$M = s_1 s_3 d + s_1 c \overline{s}_3 + \overline{s}_1 s_2 b + \overline{s}_1 \overline{s}_2 a$

Referenca: http://www.cs.uc.edu/~weaversa/BDD_Visualizer.html

```
order(d,s3,c,b,a,s2,s1) ; pazi: order se navodi obrnutim redoslijedom
or(and(s1,s3,d),and(s1,c,not(s3)),and(not(s1),s2,b),and(not(s1),not(s2),a))
print($1)
```

42. Neka je dan skup početnih stanja S_0 sa svojim ROBDD-om. Neka je također dana funkcija sljedećih stanja H (veliko eta) koja prima ROBDD stanja i vraća ROBDD stanja dostupnih u jednom koraku. Skiciraj algoritam izračunavanja ROBDD-a svih dosezljivih stanja (*reachability analysis*), počevši od S_0.