

**SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Projektni zadatak

KOMUNIKACIJSKI PROTOKOLI

G-bear, 0036000000

Zagreb, 3.12.2012.

Projektni zadatak

Projektni zadatak je bio specificirati protokol za prijenos podataka tako da se definiraju predajnik, međuspremnik i prijamnik koji međusobno sinkrono komuniciraju. Predajnik i prijamnik u slučaju pogreške moraju izvršiti retransmisiju podatka. Prijamnik provodi provjeru ispravnosti svake primljene poruke te šalje potvrdu predajniku. Pogreška može doći samo u međuspremniku i to samo kod podatka, a ne i kod potvrde.

Potrebno je pokrenuti dva predajnika i dva prijamnika. Svaki predajnik šalje po 10 poruka predajnicima preko međuspremnika. Kada međuspremnik primi po jednu poruku od svakog predajnika, odmah ih prosljeđuje jednom prijamniku, a sljedeće dvije primljene poruke prosljeđuje drugom prijamniku. Međuspremnik po principu slučajnog odabira prima poruke od predajnika, i po principu slučajnog odabira ih prosljeđuje prijamnicima.

Promela/Spin

Potrebno je specificirati protokol u Promeli:

```
init {  
  chan k1 = [0] of {int};  
  chan k2 = [0] of {int};  
  chan k3 = [0] of {int, int};  
  chan k4 = [0] of {int, int};
```

```
  run Sender(k1);  
  run Sender(k2);  
  run Buffer(k1,k2,k3,k4);  
  run Receiver(k3);  
  run Receiver(k4);  
}
```

```
proctype Sender(chan k1){  
  int count = 1;  
  int confirm = 0;
```

```
  do  
    :: count <= 10 ->  
      k1!count;  
      k1?confirm;  
  
      if  
        :: confirm != count -> skip;  
        :: confirm == count -> count = count + 1;  
      fi;
```

```
  :: else -> break  
od;  
}
```

```

proctype Buffer (chan k1; chan k2; chan k3; chan k4){
  int count = 1;
  int confirm1 = 0;
  int confirm2 = 0;
  int error = -1;
  int txorder = 0;
  int rxorder = 0;
  int data1 = 0;
  int data2 = 0;

  if
    :: rxorder = 1; txorder = 1;
    :: rxorder = 1; txorder = 2;
    :: rxorder = 2; txorder = 1;
    :: rxorder = 2; txorder = 2;
  fi;

  do
    :: count <= 10 ->

      if
        ::(txorder == 1) ->
          k1?data1;
          k2?data2;
        ::(txorder == 2) ->
          k2?data1;
          k1?data2;
        :: else -> skip;
      fi;

      if
        :: (rxorder == 1) ->
          if
            ::k3!data1, data2;
            ::k3!error, error;
          fi;

          k3?confirm1, confirm2;

          if
            :: confirm1 == error -> skip;
            :: confirm1 != error -> count = count + 1; rxorder = rxorder + 1;
          fi;

          k1!confirm1;
          k2!confirm2;

```

```
:: (rxorder == 2) ->
    if
        ::k4!data1, data2;
        ::k4!error, error;
    fi;

    k4?confirm1, confirm2;

    if
        :: confirm1 == error -> skip;
        :: confirm1 != error -> count = count + 1; rxorder = rxorder - 1;
        :: else -> skip
    fi;

    k1!confirm1;
    k2!confirm2;

:: else -> skip;

fi;

::else -> break
od
}

proctype Receiver(chan k3) {
    int count = 1;
    int error = -1;
    int data1;
    int data2;

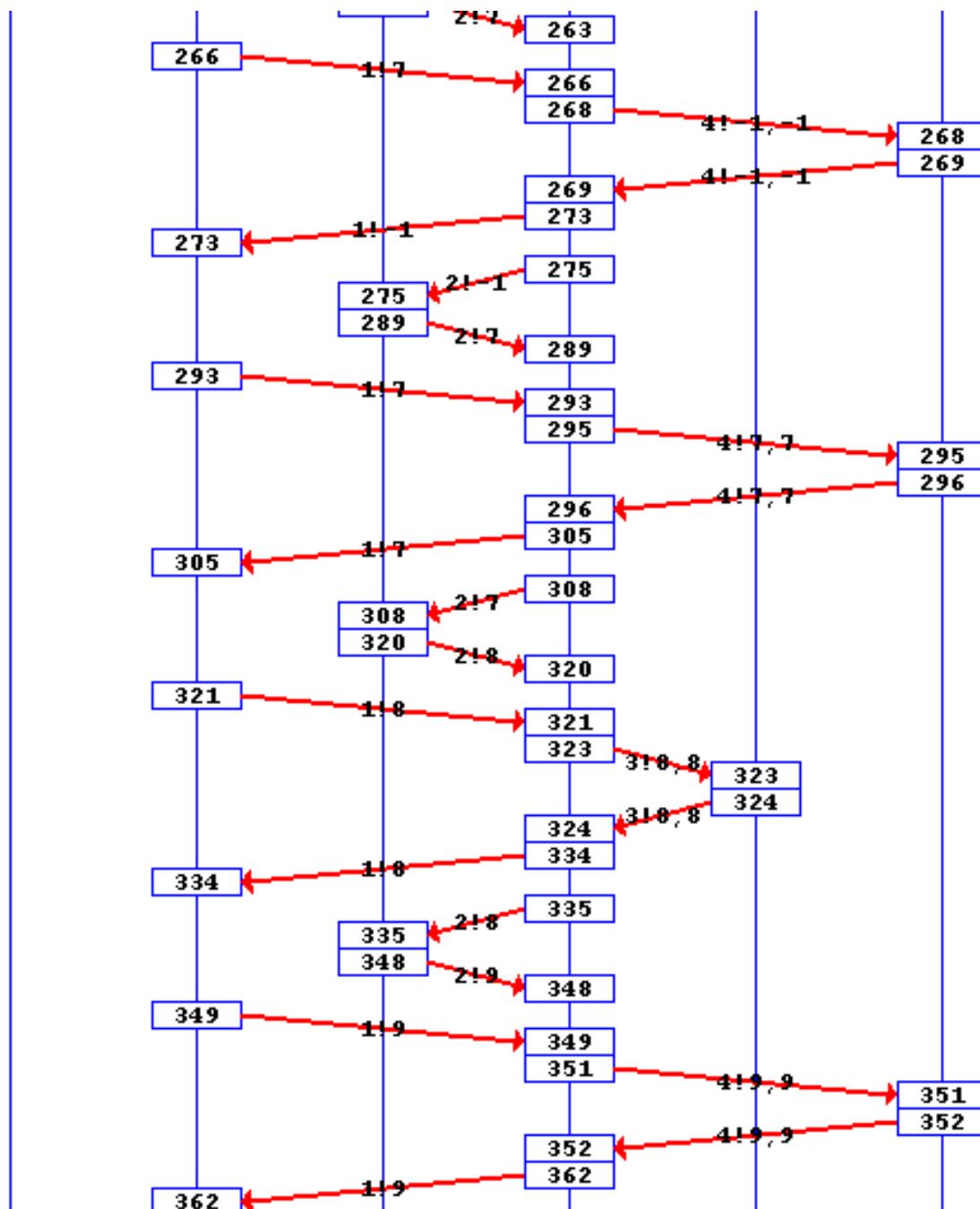
    do
        :: count <= 5 ->
            k3?data1, data2;
            k3!data1, data2;

            if
                :: data1 == error -> skip;
                :: data1 != error -> count = count + 1;
            fi;

        :: else -> break

    od
}
```

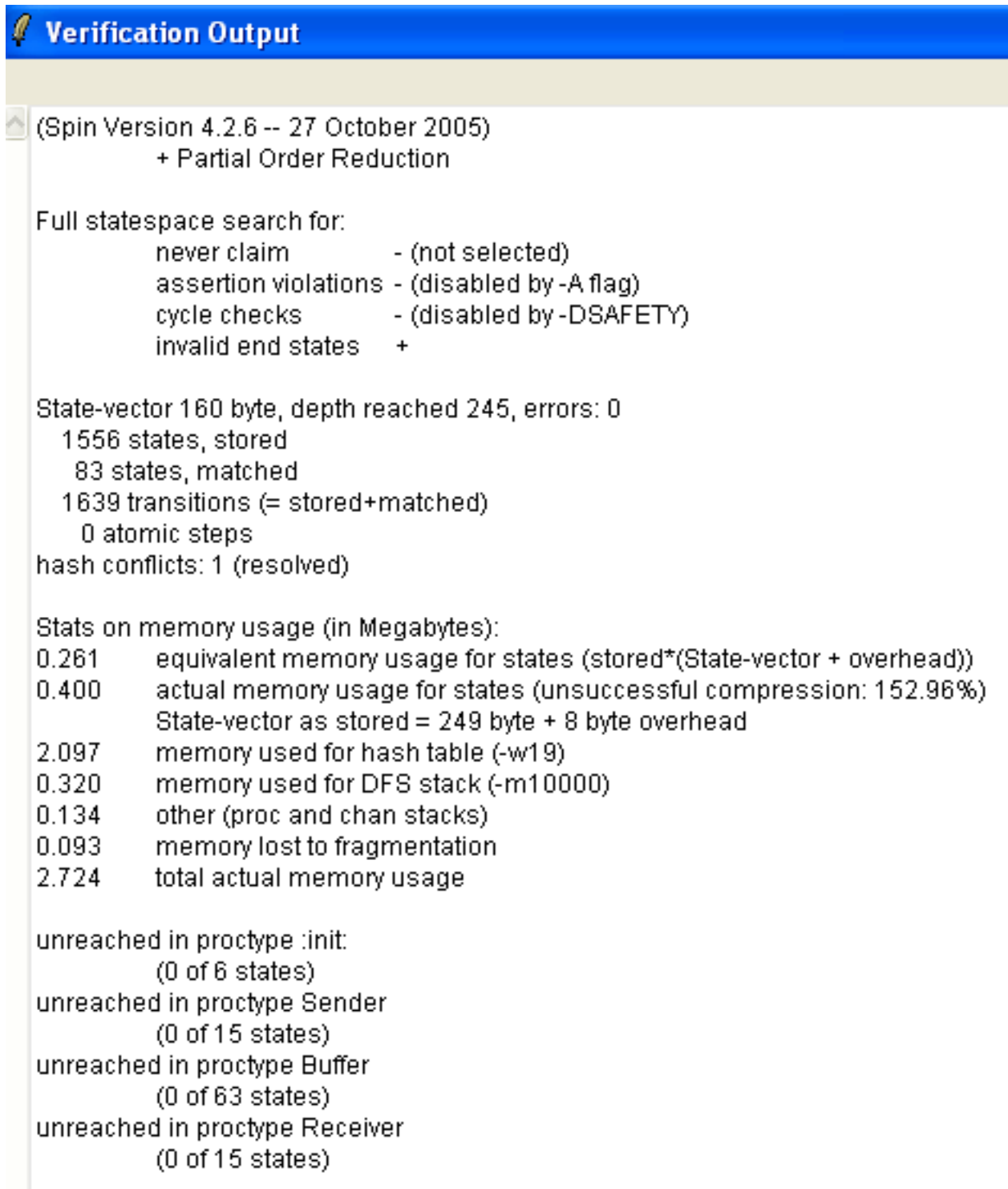
Nakon što se provjere sintaksne greške i utvrdi se da ih nema, pokreće se simulacija. Rezultat te simulacije je slijedni dijagram.



Slika 1. Slijedni dijagram

Ovdje se vidi i primjer retransmisije, te uspješna slanja. *Seed* za randomizaciju je slučajno odabran broj te se za različite *seedove* dobiju drugačiji rezultati.

Nakon uspješne simulacije, u kojoj su svi procesi završili, obavlja se verifikacija. Kod verifikacije, najvažnije je da su sva stanja dohvatljiva, to jest da nema nedohvatljivih stanja.



Verification Output

(Spin Version 4.2.6 -- 27 October 2005)
+ Partial Order Reduction

Full statespace search for:

- never claim - (not selected)
- assertion violations - (disabled by -A flag)
- cycle checks - (disabled by -DSAFETY)
- invalid end states +

State-vector 160 byte, depth reached 245, errors: 0
 1556 states, stored
 83 states, matched
 1639 transitions (= stored+matched)
 0 atomic steps
 hash conflicts: 1 (resolved)

Stats on memory usage (in Megabytes):

0.261	equivalent memory usage for states (stored*(State-vector + overhead))
0.400	actual memory usage for states (unsuccessful compression: 152.96%)
	State-vector as stored = 249 byte + 8 byte overhead
2.097	memory used for hash table (-w19)
0.320	memory used for DFS stack (-m10000)
0.134	other (proc and chan stacks)
0.093	memory lost to fragmentation
2.724	total actual memory usage

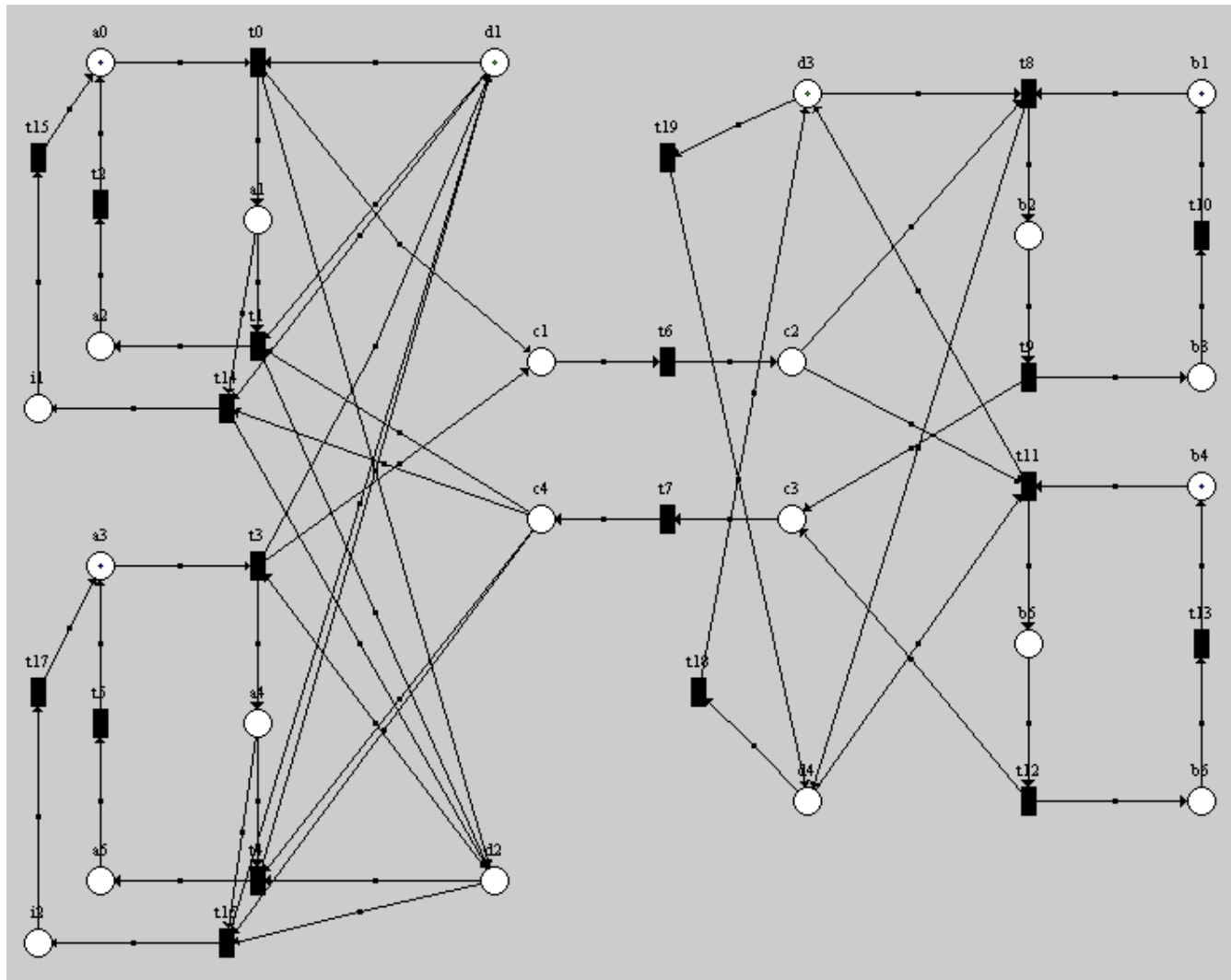
unreached in proctype :init:
 (0 of 6 states)
 unreached in proctype Sender
 (0 of 15 states)
 unreached in proctype Buffer
 (0 of 63 states)
 unreached in proctype Receiver
 (0 of 15 states)

Slika 2. Rezultat verifikacije

Ovdje se vidi da niti u jednom procesu nema nedohvatljivih stanja.

DaNAMiCS

Model protokola modeliran u Petrijevoj mreži može se vidjeti na slici 3. Izmodeliran je pomoću obojene Pertijeve mreže da se naznači razlika između ispravnih poruka i grešaka.



Slika 3. Obojena Petrijeva mreža

Značenje stanja:

- a0 – predajnik spreman za slanje poruke
- a1 – predajnik čeka potvrdu
- a2 – potvrda primljena
- a3 – predajnik spreman za slanje poruke
- a4 – predajnik čeka potvrdu
- a5 – potvrda primljena
- b1 – prijatelj spreman za primanje poruke
- b2 – poruka primljena, spreman za slanje potvrde
- b3 – potvrda poslana
- b4 – prijatelj spreman za primanje poruke
- b5 – poruka primljena, spreman za slanje potvrde

b6 – potvrda poslana
 c1 – poruke stigle u spremnik
 c2 – spreman za slanje poruka prijateljima
 c3 – potvrde stigle
 c4 – spreman za slanje potvrda prijateljima
 d1 – odabran prvi predajnik za slanje poruka
 d2 – odabran drugi predajnik za slanje poruka
 d3 – odabran prvi prijatelj za primanje poruka
 d4 – odabran drugi prijatelj za primanje poruka
 i1 – zahtjev za retransmisijom poruke
 i2 – zahtjev za retransmisijom poruke

Prijelazi:

t0 – predajnik šalje poruku
 t1 – predajnik prima potvrdu
 t2 – predajnik priprema poruku za slanje
 t3 – predajnik šalje poruku
 t4 – predajnik prima potvrdu
 t5 – predajnik priprema poruku za slanje
 t6 – slanje ispravnih ili neispravnih poruka
 t7 – priprema za slanje potvrda prijateljima
 t8 – primanje poruke
 t9 – slanje potvrde
 t10 – priprema za primanje poruke
 t11 – primanje poruke
 t12 – slanje potvrde
 t13 – pripremanje za primanje poruke
 t14 – primanje potvrde za retransmisiju
 t15 – priprema poruke za retransmisiju
 t16 – primanje potvrde za retransmisiju
 t17 – priprema poruke za retransmisiju
 t18 – vraćanje prava na primanje poruke zbog retransmisije
 t19 – vraćanje prava na primanje poruke zbog retransmisije

t0 (t3)

a0 (a3)	d1 (d2)	a1 (a4)	c1	d2 (d1)
•	•	•	•	•
•	•	•	••	•

t1 (t4)

a1 (a4)	d1 (d2)	c4	a2 (a5)	d2 (d1)
•	•	•	•	•
•	•	••	•	•

t14 (t16)

a1 (a4)	d1 (d2)	c4	i1 (i2)	d2 (d1)
•	•	•	•	•
•	•	••	•	•

t2 (t5)

a2 (a5)	a0 (a3)
•	•

t15 (t17)

i1 (i2)	a0 (a3)
•	•

t6

c1	c2
••	••
••	••

t7

c3	c4
••	••
••	••

t8 (t11)

c2	d3 (d4)	b1 (b4)	b2 (b5)	d4 (d3)
••	•	•	••	•
••	•	•	••	•

t9 (t12)

b2 (b5)	c3	b3 (b5)
••	••	•
••	••	•

t10 (t13)

b3 (b6)	b1 (b4)
•	•

t18

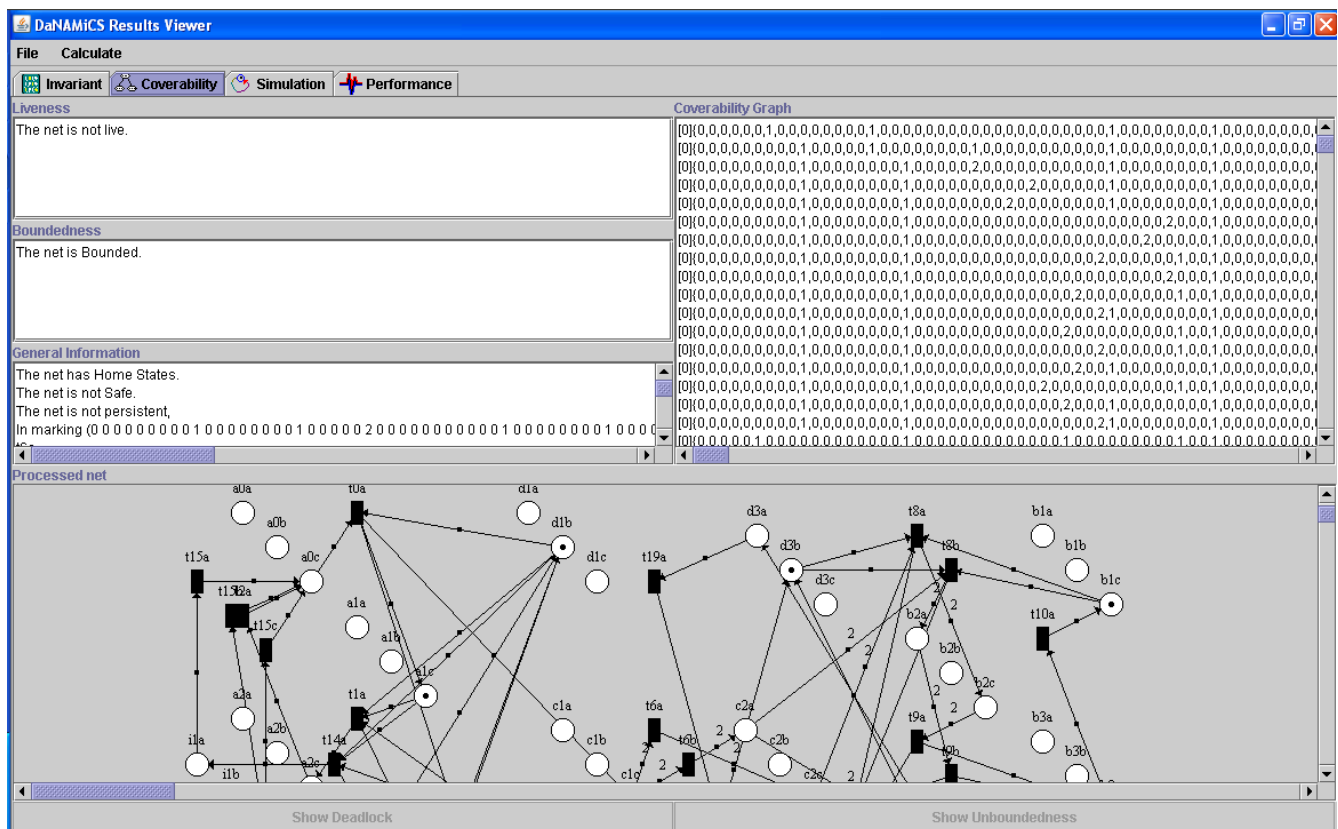
d4	d3
•	•

t19

d3	d4
•	•

Nakon što je mreža izmodelirana, potrebno ju je provjeriti. To se čini pokretanjem animacije koja se nalazi u izborniku *Animate* i pokreće se sa *Play*. Animacija se odvija korak po korak na način da sami biramo koji će se od mogućih prijelaza izvršiti. Također, na mjestima gdje je to moguće, biramo da li se kroz međuspremnik šalje ispravna poruka ili greška.

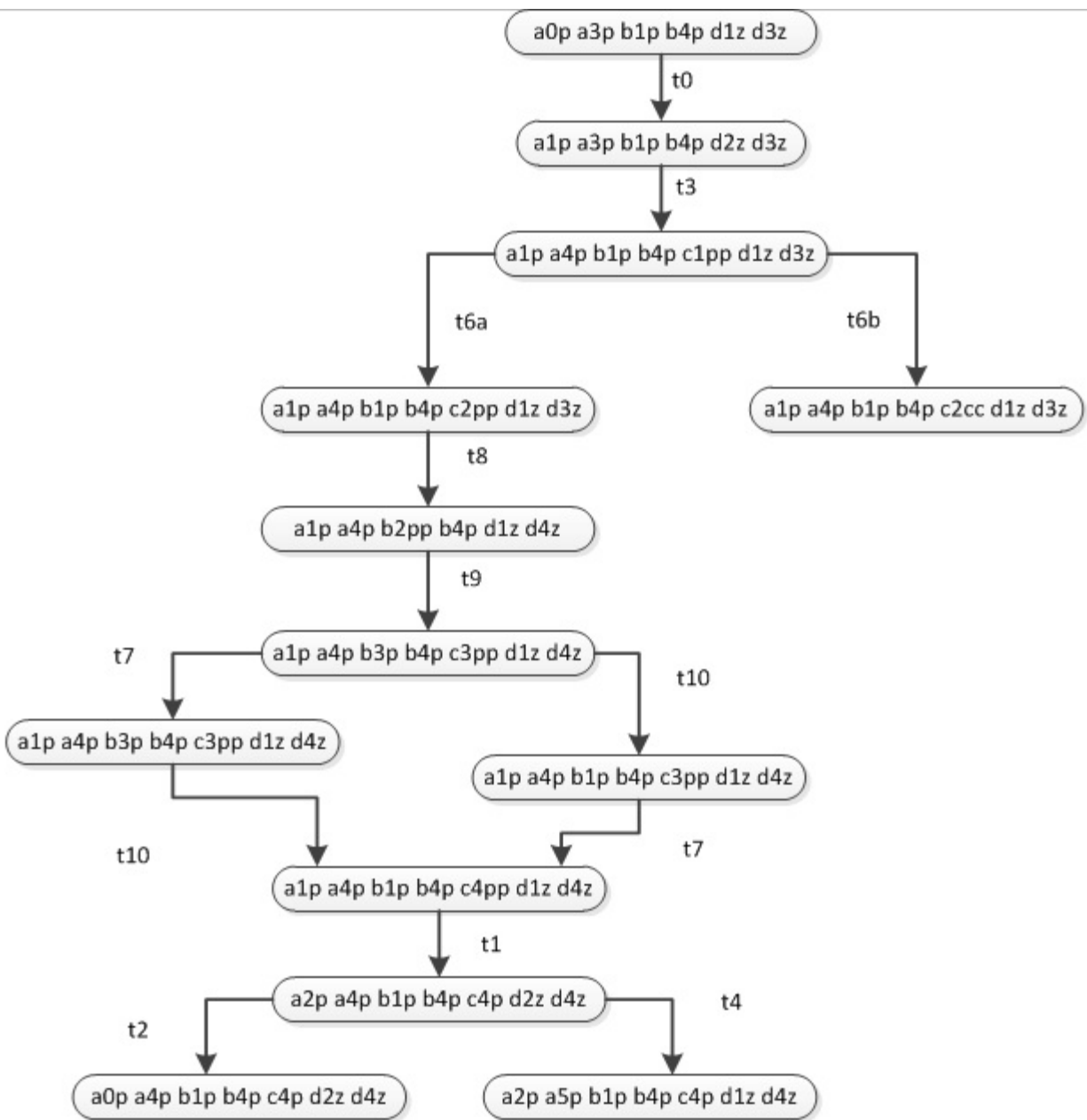
Nakon izvršenja animacije, potrebno je pokrenuti i analizu svojstava Petrijeve mreže. To se radi tako da se u izborniku odabere *Analysis* pa *Coverability graph*. Otvorit će se prozor koji će omogućiti neke izbore. Treba ih sve označiti i pokrenuti analizu. Rezultat je prikazan slikom 4.



Slika 4. Analiza mreže

Ovdje se vide neka svojstva mreže. Mreža je živa*, ograničena, reverzibilna, ali nije sigurna niti je perzistentna. Tu se vide i graf procesa petrijeve mreže, te graf pokrivenosti.

**iako piše da mreža nije živa, ona to je. Bila je jedna greška koju sam ispravio, odsimulirao sam još puno puta svaki korak i nisam naišao na blokiranje. Tako da pretpostavljam da je stvar buga u programu.*



Slika 5. Graf stanja

Na slici 5 prikazan je graf stanja, točnije njegov početak. Nacrtati cijeli graf stanja bilo bi prekomplikirano zbog velikog broja njegovih stanja. Slovo i broj označavaju stanje, a znak ili skupina znakova pored označavaju broj oznaka i njihove boje (p – plava, c – crvena, z – zelena). Primjerice, `a1p` označava da se u stanju `a1` nalazi jedna oznaka plave boje, dok `c3cc` označava da se u stanju `c3` nalaze dvije oznake crvene boje.

Usporedba rezultata

Cilj ove vježbe bio je modeliranje zadanog protokola u Spinu i DaNAMiCS-u tako da na kraju imamo isti rezultat, odnosno da imamo dva protokola razvijena u dva različita programska alata koji u suštini rade istu stvar i daju isti krajnji rezultat. Prijelazi u Petrijevoj mreži odgovaraju onima u koji se izvode u slijednom dijagramu koji smo dobili simulacijom u Spinu.

Dok je u Spinu napisano programsko rješenje, DaNAMiCS je više usmjeren na modeliranje mreže i njezin grafički prikaz, te nam prikazuje unutarnje prijelaze koji se prilikom izvođenja simulacije u Spinu ne vide.

Iako su oba programska jezika vrlo jednostavna za korištenje, postoje neke njihove poteškoće. U Spinu je to nepreglednost pri javljanju pogrešaka, odnosno potrebno je jako dobro poznavanje njegova rada i programskog jezika da se razumije gdje se javlja pogreška, pogotovo u verifikaciji. Također, randomiziranje kod *if*-a je na prvu malo neobičan, no korisnik se s vremenom privikne. DaNAMiCS, s druge strane, nema tih problema, no ima dosta *bugova*. Onaj najizraženiji je taj da se gube sve postavke pri gašenju programa, pa ga nije preporučljivo gasiti sve do kraja projekta i njegove prezentacije. Također, neke stvari rade tek nakon gašenja programa te ponovnog pokretanja istog.

LITERATURA

- [1] LOVREK, I. *Modeli telekomunikacijskih procesa*, Zagreb: Školska knjiga, 1997.
- [2] HOLZMAN, G.J., *Basic Spin Manual*, New Jersey, AT&T Bell Laboratories
- [3] CHANGUION, DAVIES, NELTE, *DaNAMiCS: Modelling Concurrent Systems*, 1998.