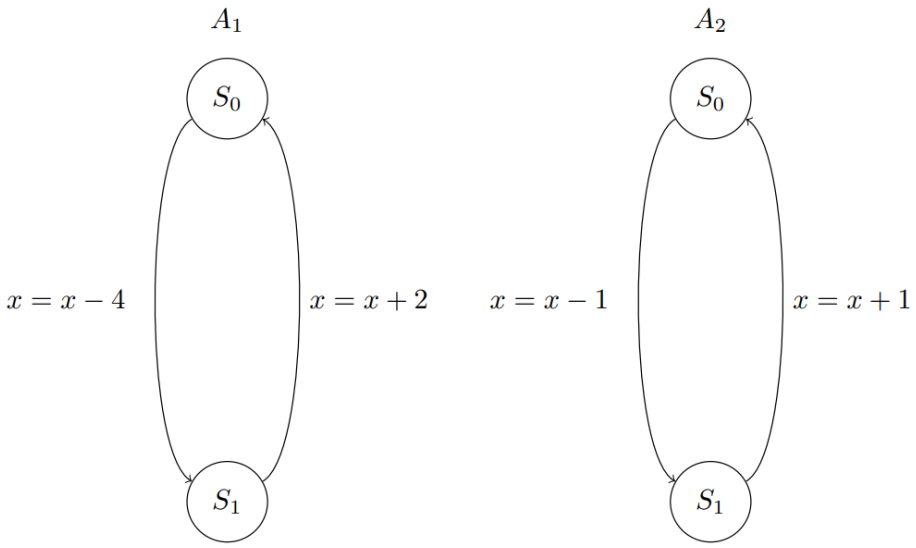
IME I PREZIME:_____ Ak. god. 2019./2020.

JMBAG:_____

3. domaća zadaća iz Formalnih metoda u oblikovanju sustava

Prvi dio

Zadana su dva konačna diskretna automata A_1 i A_2 prema slici: (početna stanja su uvijek S_0, a završna S_1)



Slika 1: FSA A_1 i A_2

U okviru domaće zadaće potrebno je:

a) Detaljno opisati strukturu automata A_1 i A_2 prema definiciji FSA A = (S, s_0, L, T, F) (odrediti elemente svakog od skupova S, s_0, L, . . .)

```
FSA A1 = ({S0, S1}, {S0}, {x = x - 4, x = x + 2}, {(S0, x = x - 4,
S1),(S1, x = x + 2, S0)}, {S1})

A1.S = {S0, S1}
A1.s0 = {S0}
A1.L = {x = x - 4, x = x + 2}
A1.T = {(S0, x = x - 4, S1),(S1, x = x + 2, S0)}
A1.F = {S1}
```

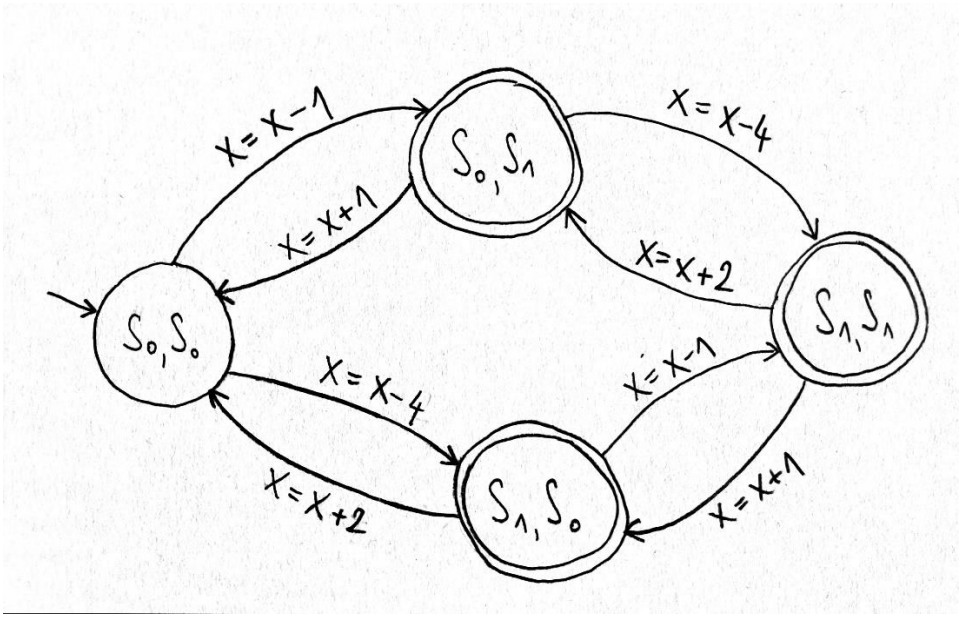
FSA A_2 = ({S_0, S_1}, {S_0}, {x = x - 1, x = x + 1}, {(S_0, x = x - 1, S_1),(S_1, x = x + 1, S_0)}, {S_1})

A_2.S = {S_0, S_1}
A_2.s_0 = {S_0}
A_2.L = {x = x - 1, x = x + 1}
A_2.T = {(S_0, x = x - 1, S_1),(S_1, x = x + 1, S_0)}
A_2.F = {S_1}

b) Odrediti asinkroni produkt automata A_1 i A_2 i nacrtati ga.

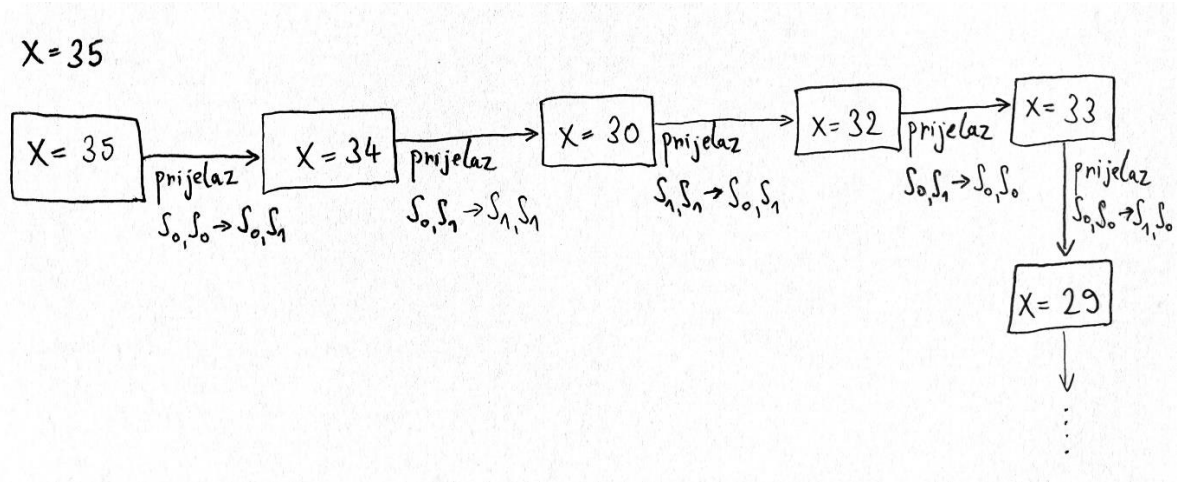
A = A_1 X A_2

A.S = {(S_0, S_0), (S_0, S_1), (S_1, S_0), (S_1, S_1)}
A.s_0 = {(S_0, S_0)}
A.L = {x = x − 4, x = x + 2, x = x − 1, x = x + 1}
A.T = {{(S_0, S_0), x = x − 1, (S_0, S_1)}, {(S_0, S_1), x = x + 1, (S_0, S_0)}, {(S_0, S_0), x = x − 4, (S_1, S_0)}, {(S_1, S_0), x = x + 2, (S_0, S_0)}, {(S_0, S_1), x = x − 4, (S_1, S_1)}, {(S_1, S_1), x = x + 2, (S_0, S_1)}, {(S_1, S_0), x = x − 1, (S_1, S_1)}, {(S_1, S_1), x = x + 1, (S_1, S_0)}}
A.F = {(S_0, S_1), (S_1, S_0), (S_1, S_1)}

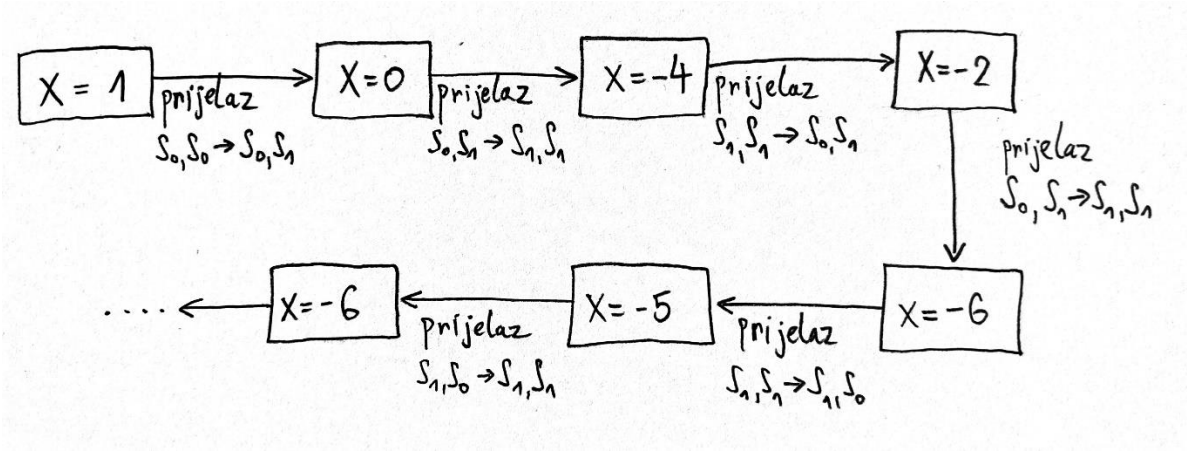


c) Odrediti ekspandirani asinkroni produkt za prvih 5–10 po volji odabranih članova i nacrtati ga.

Na stranici #7 zadana je vrijednost N = 35 koja će biti korištena za početnu vrijednost x.



Zbog lakšeg određivanja istinitosti formula z zadacima d) i e) nacrtan je još jedan ekspandirani asinkroni produkt za x = 1.



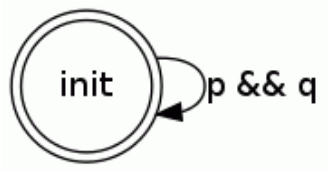
d) Pomoću ekspandiranog produkta odrediti istinitost LTL formule ◊□p ako je p ≡ x ≤ 0. Obrazložiti rješenje, posebice komentirati mogućnost rješavanja bez primjene programskih alata.

Iz druge sekvence u zadatku c) vidljivo je da je LTL formula ◊□p istinita. Od drugog člana sekvence (x = 0), koji zadovoljava uvjet x ≤ 0, svaki idući član također zadovoljava uvjet x ≤ 0, što pokazuje istinitost formule ◊□p.

e) Pomoću ekspandiranog produkta odrediti istinitost LTL formule ◊p ako je p ≡ x < 0. Obrazložiti rješenje, posebice komentirati mogućnost rješavanja bez primjene programskih alata.

Iz druge sekvence u zadatku c) vidljivo je da je LTL formula ◊p istinita. Primjer, kod trećeg člana sekvence (x = -4) uvjet x < 0 je zadovoljen, što pokazuje istinitost formule ◊p.

f) Nacrtati moguću realizaciju Büchi automata za LTL formulu: □(p ∧ q) .



Drugi dio

1) Instalirajte programski alat Spin (http://spinroot.com/spin/Bin/index.html). Instalacije se svodi na kopiranje izvršnog programa.
Za one koji hoće više, sve instrukcije jezika Promela možete pronaći na http://spinroot.com/spin/Man/promela.html kao i službene upute ("manual") na http://spinroot.com/spin/Man/Manual.html).

2) Editirajte automate A_1 i A_2 kao promela procese A i B (vidjeti predložak na strani 7).
Napomena: Promela file nazvati prezime.prm (npr. blaskovic.prm). Polazeći od zadanog Promela modela koji se sastoji od dva procesa A i B analizirati će se LTL formula ◊p gdje je p ≡ (x <= 0).

3) Pokrenite simulaciju: spin -u20 -p -c -g prezime.prm.
Prepišite prvih 12 članova. Pismeno obrazložite istovjetnosti i razlike između ekspandiranog asinkronog produkta iz domaće zadaće i rezultata simulacije.

```
1:    proc  1 (B:1) prezime.prm:22 (state 1) [(1)]
2:    proc  0 (A:1) prezime.prm:10 (state 1) [(1)]
3:    proc  1 (B:1) prezime.prm:24 (state 2) [x = (x-1)]
                x = 34
4:    proc  1 (B:1) prezime.prm:24 (state 3) [goto end_S1]
5:    proc  1 (B:1) prezime.prm:27 (state 4) <valid end state>
[x = (x+1)]
                x = 35
6:    proc  1 (B:1) prezime.prm:27 (state 5) [goto S0]
7:    proc  1 (B:1) prezime.prm:24 (state 2) [x = (x-1)]
                x = 34
8:    proc  0 (A:1) prezime.prm:12 (state 2) [x = (x-4)]
                x = 30
```

```
9:    proc  0 (A:1) prezime.prm:12 (state 3) [goto end_S1]
10:    proc  0 (A:1) prezime.prm:15 (state 4) <valid end state>
[x = (x+2)]
                x = 32
11:    proc  0 (A:1) prezime.prm:15 (state 5) [goto S0]
12:    proc  1 (B:1) prezime.prm:24 (state 3) [goto end_S1]
```

Kako se u simulaciji nasumično određuje u koje iduće stanje će automat prijeći, svakim pokretanjem simulacije dobije se drugačiji ishod. Isto tako su prijelazi u prvoj sekvenci u zadatku c) birani proizvoljno, stoga je i tu drugačiji ishod ovisno o odabranim prijelazima. Možemo primjerice usporediti 5. član iz sekvence i simulacije. 5. član iz sekvence je x = 33, dok je 5. član iz simulacije x = 30.

4) Generirajte analizator: spin -a prezime.prm.

5) Prevedite u izvršni oblik npr.: gcc -o pan pan.c.

6) Pozovite analizator: pan -a ili ./pan -e. Pismeno obrazložite da li je uvjet p zadovoljen ?

U ispisu analizatora vraćena je „greška" (errors: 1), što znači da je uvjet p zadovoljen.

```
State-vector 24 byte, depth reached 128, errors: 1
```

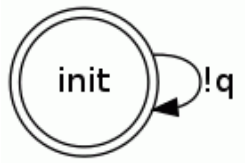
7) Pokrenite "error trail" opciju (pronalaženje protuprimjera) sa spin -t -p -c -g prezime.prm.
Da li postoji sekvenca u kojoj varijabla x na kraju poprima vrijednost $x \leq 0$?
Koliko koraka (eng. "steps") sadrži ?

U ispisu „error trail" opcije vraćeno je da postoji sekvenca u kojoj varijabla x na kraju poprima vrijednost $x \leq 0$, a ta sekvenca sadrži 129 koraka.

```
spin: prezime.prm:34, Error: assertion violated
spin: text of failed assertion: assert(!((x<=0)))
Never claim moves to line 34    [assert(!((x<=0)))]
spin: trail ends after 129 steps
```

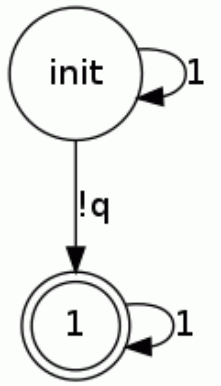
8) Prepišite instrukcije za Büchi automat koje generira Spin spin -f '!<>q'.
Nacrtajte pripadni Büchi automat.

```
never  {    /* !<>q */
accept_init:
T0_init:
        do
        :: (! ((q))) -> goto T0_init
        od;
}
```



9) Prepišite instrukcije za Büchi automat koje generira Spin spin -f '![]q'.
Nacrtajte pripadni Büchi automat.

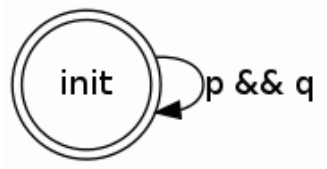
```
never  {    /* ![]q */
T0_init:
        do
        :: atomic { (! ((q))) -> assert(!(! ((q)))) }
        :: (1) -> goto T0_init
        od;
accept_all:
        skip
}
```



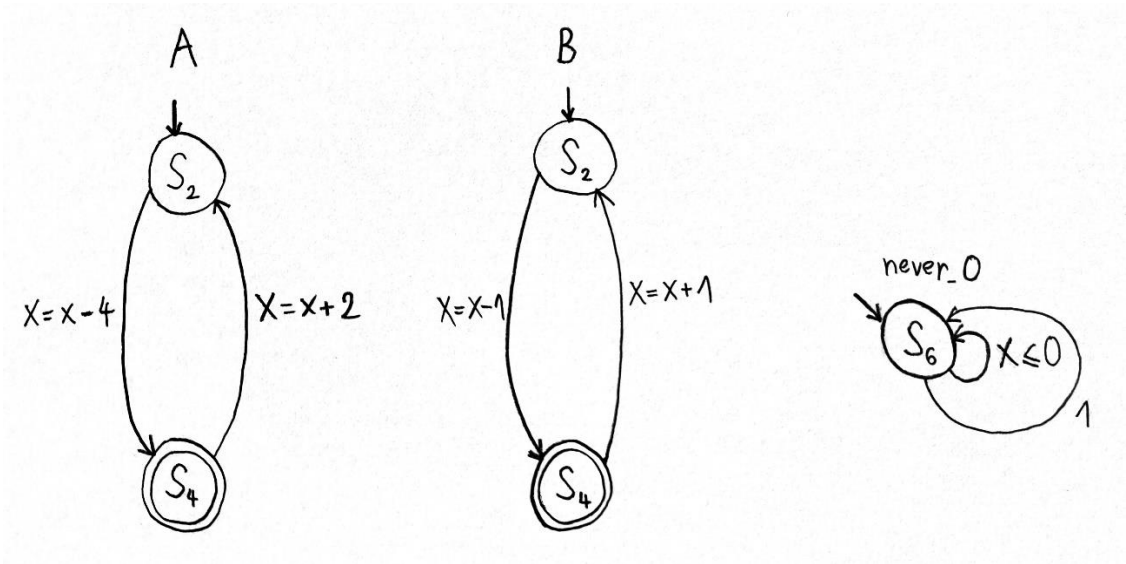
10) Na isti način koristeći Spin nacrtajte automat iz Vaše domaće zadaće (pitanje f)).

Pomoću naredbe *spin -f '[](p && q)'* generira se sljedeći kod:

```
never  {    /* [](p && q) */
accept_init:
T0_init:
        do
        :: ((p && q)) -> goto T0_init
        od;
}
```



11) Generirajte analizator sa spin -a -o3 prezime.prm, prevedite te pozovite analizator sa pan -d. Precrtajte tako dobivene FSA. Objasnite razlike kao i istovjetnosti prema automatima iz domaće zadaće ? Usporedite stanja prema slici na stranici 1 i stanja dobivena s opcijom pan -d. U čemu je razlika ?



Osim što imaju drugačije nazive stanja (S_2 umjesto S_0 i S_4 umjesto S_1), dobiveni automati A i B jednaki su zadanim automatima A_1 i A_2 iz domaće zadaće. Prijelazi između stanja S_2 i S_4 automata A i B jednaki su prijelazima između stanja S_0 i S_1 automata A_1 i A_2. Novost koju je analizator donio je automat never_0 za zadanu LTL formulu ◊p, gdje je p ≡ (x <= 0). Automat

never_0 ima jedno stanje S_6 koje ima dva prijelaza u samo sebe za uvjete 1 i $x \leq 0$.

Odgovorite na sljedeća pitanja:
Ponovite postupak za ◊□p (modificirati ”never claim” spin -f ’<>[]p’ na kraju prezime.prm datoteke.

p1) Da li postoji sekvenca u kojoj varijabla x na kraju poprima vrijednost $x \leq 0$?

U ispisu analizatora vraćena je „greška“ (errors: 1), što znači da je uvjet p zadovoljen, odnosno da varijabla x na kraju poprima vrijednost $x \leq 0$ i da svaki sljedeći član također ima vrijednost $x \leq 0$.

State-vector 24 byte, depth reached 141, errors: 1

p2) Koliko koraka (eng. ”steps”) sadrži ?

U ispisu „error trail“ opcije vraćeno je da sekvenca sadrži 142 koraka, a na 140. koraku počinje ciklus.

```
  <<<<<START OF CYCLE>>>>>
Never claim moves to line 39    [((x<=0))]
140:    proc  1 (B:1) prezime.prm:24 (state 2) [x = (x-1)]
                x = -2
142:    proc  1 (B:1) prezime.prm:27 (state 4) <valid end state>
[x = (x+1)]
                x = -1
spin: trail ends after 142 steps
```

p3) Da li je moguće problem riješiti bez LTL formule samo pomoću assert naredbi ? Obrazložite odgovor.

Pomoću assert naredbe nije moguće riješiti problem, jer je uvjet zadovoljen tek u određenog trenutka, što nije moguće zadati assert naredbi.

p4) Da li je moguće problem riješiti bez LTL formule samo pomoću simulacije ? Obrazložite odgovor.

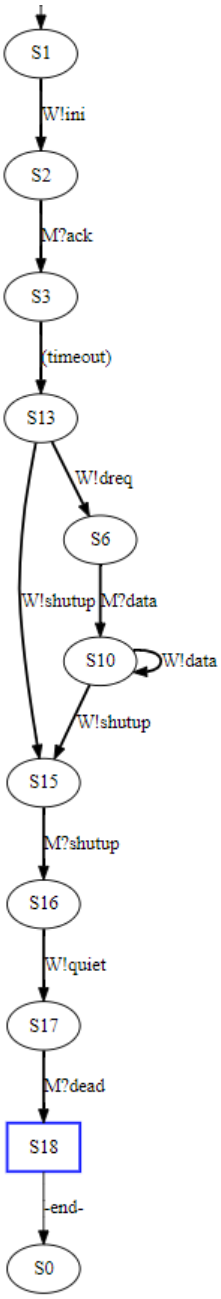
Simulacijom se može riješiti problem, no nije preporučljivo, jer se simulacije izvode slučajnim odabirom, što potencijalno može uvelike oduljiti postupak.

Stranica #5

Zadan je Promela model komunikacijskog protokola koji opisuje dio moguće realizacije protokola za preuzimanje datoteka (eng. ”download”). Ako je N = 16, pomoću programskog alata Spin odredite:

a) stanja iz kojih nema napretka (eng. ”deadlock”)

Stanje iz kojeg nema napretka je S_{10}, što se može vidjeti na generiranom grafu.



b) dolazi li protokol u završno stanje ?

Protokol ne dolazi u završno stanje, jer ni u jednom od ispisanih koraka ne vidimo napisan *<valid end state>*.

```
  1   W!ini
  1:    proc  0 (Mproc:1) download.prm:10 (state 1)     [W!ini]
                queue 1 (W): [ini]
  1   .   W?ini
  2:    proc  1 (Wproc:1) download.prm:32 (state 1)     [W?ini]
                queue 1 (W):
  2   .   M!ack
  3:    proc  1 (Wproc:1) download.prm:33 (state 2)     [M!ack]
                queue 2 (M): [ack]
                queue 1 (W):
  2   M?ack
  4:    proc  0 (Mproc:1) download.prm:11 (state 2)     [M?ack]
                queue 2 (M):
                queue 1 (W):
  5:    proc  1 (Wproc:1) download.prm:49 (state 11)    [.(goto)]
                queue 2 (M):
                queue 1 (W):
      timeout
  6:    proc  0 (Mproc:1) download.prm:13 (state 3)     [(timeout)]
                queue 2 (M):
                queue 1 (W):
  1   W!dreq
  7:    proc  0 (Mproc:1) download.prm:16 (state 5)     [W!dreq]
                queue 2 (M):
                queue 1 (W): [dreq]
  1   .   W?dreq
  8:    proc  1 (Wproc:1) download.prm:36 (state 3)     [W?dreq]
                queue 2 (M):
                queue 1 (W):
  2   .   M!data
  9:    proc  1 (Wproc:1) download.prm:37 (state 4)     [M!data]
                queue 2 (M): [data]
                queue 1 (W):
 10:    proc  1 (Wproc:1) download.prm:49 (state 11)    [.(goto)]
                queue 2 (M): [data]
                queue 1 (W):
  2   M?data
 11:    proc  0 (Mproc:1) download.prm:17 (state 6)     [M?data]
                queue 2 (M):
                queue 1 (W):
 12:    proc  0 (Mproc:1) download.prm:23 (state 11)    [.(goto)]
                queue 2 (M):
                queue 1 (W):
  1   W!data
 13:    proc  0 (Mproc:1) download.prm:19 (state 7)     [W!data]
                queue 2 (M):
                queue 1 (W): [data]
  1   .   W?data
 14:    proc  1 (Wproc:1) download.prm:38 (state 5)     [W?data]
```

```
                queue 2 (M):
                queue 1 (W):
15:    proc  0 (Mproc:1) download.prm:23 (state 11)    [.(goto)]
                queue 2 (M):
                queue 1 (W):
 2   .   M!data
16:    proc  1 (Wproc:1) download.prm:40 (state 6)     [M!data]
                queue 2 (M): [data]
                queue 1 (W):
17:    proc  1 (Wproc:1) download.prm:49 (state 11)    [.(goto)]
                queue 2 (M): [data]
                queue 1 (W):
 1   W!shutup
18:    proc  0 (Mproc:1) download.prm:20 (state 8)     [W!shutup]
                queue 2 (M): [data]
                queue 1 (W): [shutup]
19:    proc  0 (Mproc:1) download.prm:21 (state 9)     [goto :b0]
                queue 2 (M): [data]
                queue 1 (W): [shutup]
 1   .   W?shutup
20:    proc  1 (Wproc:1) download.prm:44 (state 7)     [W?shutup]
                queue 2 (M): [data]
                queue 1 (W):
 2   .   M!shutup
21:    proc  1 (Wproc:1) download.prm:45 (state 8)     [M!shutup]
                queue 2 (M): [data][shutup]
                queue 1 (W):
22:    proc  1 (Wproc:1) download.prm:46 (state 9)     [goto :b1]
                queue 2 (M): [data][shutup]
                queue 1 (W):
     timeout
```