/Zavod za telekomunikacije

Komunikacijski protokoli

Modeliranje komunikacijskih protokola programskim alatom Promela/Spin

/Zavod za telekomunikacije

- **slobodno smijete:**
  - **dijeliti** — umnožavati, distribuirati i javnosti priopćavati djelo
  - **remiksirati** — prerađivati djelo

- **pod sljedećim uvjetima:**
  - **imenovanje**. Morate priznati i označiti autorstvo djela na način kako je specificirao autor ili davatelj licence (ali ne način koji bi sugerirao da Vi ili Vaše korištenje njegova djela imate njegovu izravnu podršku).
  - **nekomercijalno**. Ovo djelo ne smijete koristiti u komercijalne svrhe.
  - **dijeli pod istim uvjetima**. Ako ovo djelo izmijenite, preoblikujete ili stvarate koristeći ga, preradu možete distribuirati samo pod licencom koja je ista ili slična ovoj.

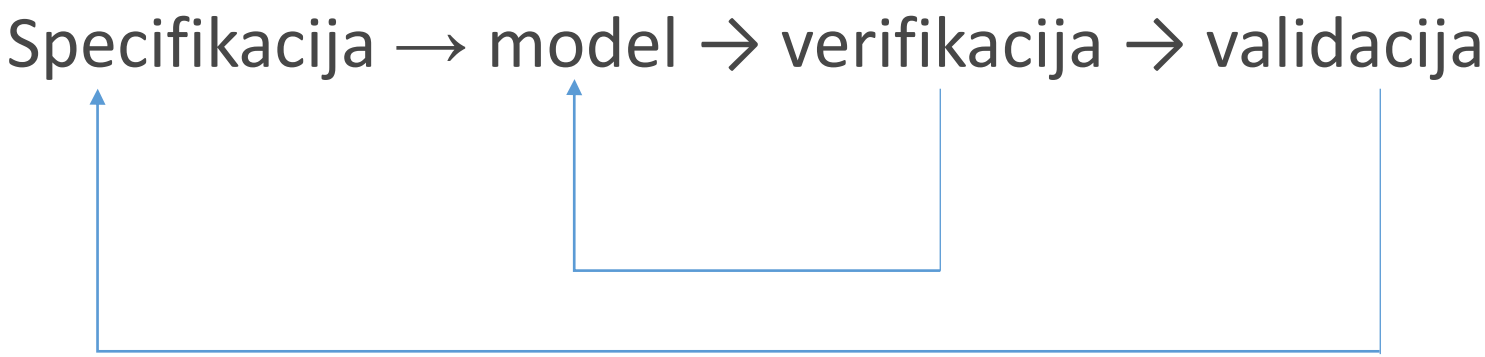
U slučaju daljnjeg korištenja ili distribuiranja morate drugima jasno dati do znanja licencne uvjete ovog djela.
Najbolji način da to učinite je linkom na ovu internetsku stranicu.
Od svakog od gornjih uvjeta moguće je odstupiti, ako dobijete dopuštenje nositelja autorskog prava.
Ništa u ovoj licenci ne narušava ili ograničava autorova moralna prava.

Tekst licencije preuzet je s http://creativecommons.org/.

Sadržaj predavanja

- Specifikacija, modeliranje i verifikacija protokola
- Programski alat Promela/Spin
- Jezik Promela – detalji

Modeliranje protokola



Promela/Spin

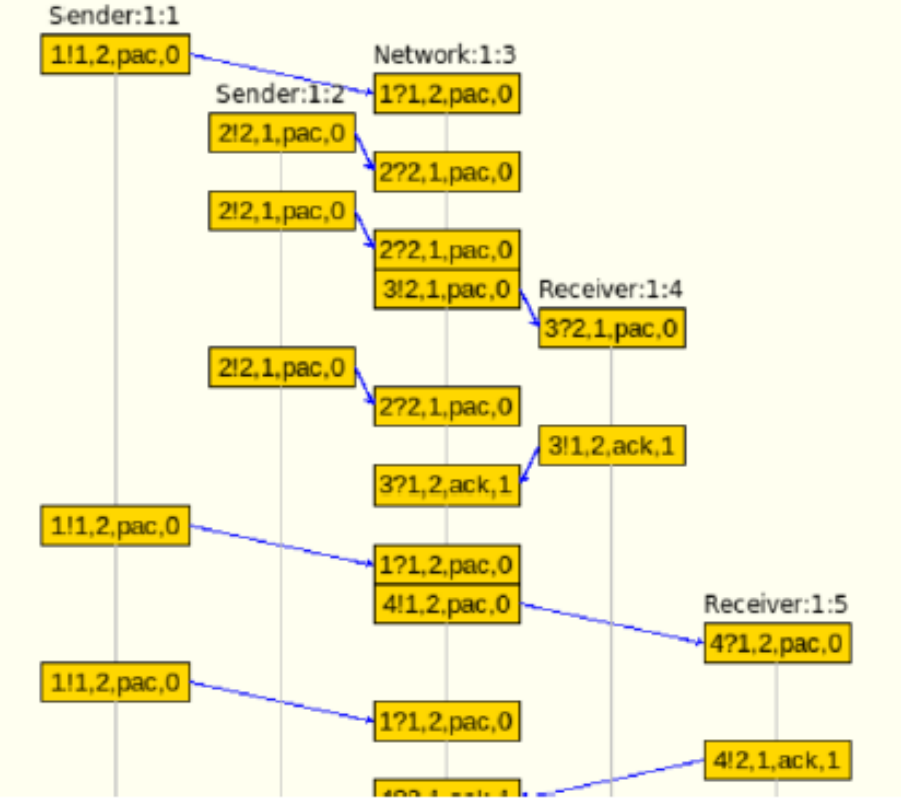
/Zavod za telekomunikacije

- http://spinroot.com/spin/Man/Manual.html
- Promela
  - Jezik za modeliranje komunikacijskih protokola i raspodijeljenih sustava
  - Modelira se komunikacija
  - Ostali detalji (npr. podaci koji se prenose) se zanemaruju
- Spin
  - Alat za verifikaciju raspodijeljenih sustava i komunikacijskih protokola
  - Provjeravaju se odsustvo zastoja (*deadlock*), nedefinirani primatelji poruka, nepotrebna stanja

Spin

/Zavod za telekomunikacije

Redoslijed poruka



Verifikacija

unreached in init
(0 of 6 states)
unreached in proctype Sender
(0 of 15 states)
unreached in proctype Network
(0 of 73 states)
unreached in proctype Receiver
(0 of 11 states)

pan: elapsed time 0.23 seconds
No errors found -- did you verify all claims?

Promela

- Osnovni elementi jezika
  - Procesi – globalni
  - Kanali – globalni/lokalni
  - Varijable – globalni/lokalni

Izvodivost

- Nema razlike između uvjeta (*conditions*) i izraza (*statements*)
- Uvjeti (npr. `a==b`) se mogu koristiti kao izrazi
- Svaki izraz je izvodiv ili blokiran
- Izvodivost je osnovni način sinkronizacije

```
while (a!=b)
  skip
```

Isto se postiže i izrazom: `(a==b)`

Tipovi podataka (1)

Ime	Veličina (bitovi)		Raspon
bit	1	nepredznačni	0..1
bool	1	nepredznačni	0..1
byte	8	nepredznačni	0..255
mtype	8	nepredznačni	0..255
short	16	predznačni	-2^15..2^15-1
int	32	predznačni	-2^32..2^32-1

Tipovi podataka (2)

```
mtype = {ack, err};
=
#define ack 2;
#define err 1;


byte state[N];
state[0] = state[4]+8*state[4*2/n];
```

Procesi (1)

- Opisuju ponašanje kojim se mijenja stanje varijabli i komunikacijskih kanala

```
proctype A(){
  byte state;
  state=3
}
```

- Za odvajanje izraza može se koristiti ; ili ->

Procesi (2)

```
byte state = 2;

proctype A() {
  (state == 1) -> state=3
}

proctype B() {
  state = state - 1
}
```

Inicijalizacija procesa (1)

- Pri pokretanju Promela modela, izvršava se proces `init`

```
int state;
init {
  state=3;
  run A();
  run B();
}
```

Inicijalizacija procesa (2)

```
proctype A (byte state; short foo) {
  (state == 1) -> state = foo
}

init {
  run A(1,3)
}
```

Izvođenje procesa

- Čitanje i izmjena globalne varijable

```
byte state = 1;
proctype A() {
  (state==1) -> state = state +1
}
proctype B() {
  (state==1) -> state = state -1
}
init {
  run A(); run B()
}
```

Naredba `atomic`

- Naredbe u bloku `atomic` se izvode slijedno i ništa ih ne može prekinuti

```
byte state = 1;
proctype A() {
  atomic {
       (state==1) -> state = state +1
  }
}
proctype B() {
  atomic {
       (state==1) -> state = state -1
  }
}
init {run A(); run B()}
```

Kanali (1)

- Služe za modeliranje prijenosa poruka između procesa

```
chan qname = [16] of { short }
chan qname = [16] of { byte, int, chan, byte}
```

- Slanje poruke: `qname!expr`
- Primanje poruke: `qname?msg`
- Prijenos poruka po FIFO redoslijedu

Kanali (2)

- Moguće je slanje i primanje više podataka

```
qname!expr1,expr2,expr3
qname?var1,var2,var3
=
qname!expr1(expr2,expr3)
qname?var1(var2,var3)
```

Kanali (3)

```
proctype A(chan q1) {
  chan q2;
  q1?q2;
  q2!123;
}
Proctype B(chan qforb) {
  int x;
  qforb?x;
  printf("x = %d\n", x)
}
Init {
  chan qname = [1] of {chan};
  chan qforb = [1] of {int};
  run A(qname);
  run B(qforb);
  qname!qforb
}
```

Kanali (4)

- Broj poruka u kanalu: `len(qname)`
- Provjera nalazi li se u kanalu podatak: `qname?[var]`
  - 1 – ako se nalazi (naredba je izvršiva)
  - 0 – ako se ne nalazi (naredba nije izvršiva)

- Sinkrona komunikacija – kanal ne pohranjuje podatke

```
chan qname = [0] of {byte}
```

- Asinkrona komunikacija – kanal može pohraniti podatke

```
chan qname = [N] of {byte}
```

Komunikacija između procesa

```
#define msgtype 33;
chan name = [0] of {byte, byte, byte};
proctype A() {
  name!msgtype(124,123);
  name!msgtype(121,125)
}
proctype B() {
  byte state1,state2;
  name?msgtype(state1, state2)
}
init {
  atomic {
        run A();
        run B();
  }
}
```

Kontrola toka

- Slijed izraza

```
(a==b) -> state=3
```

- Paralelno izvođenje procesa
- Blok `atomic`

- Odabir slučaja (naredba `if`)
- Ponavljanje (petlje)
- Uvjetni skokovi

Odabir slučaja (1)

```
if
:: (a != b) -> option1
:: (a == b) -> option2
fi
```

- Izvršava se samo jedan slijed naredbi
- Ako se ne može izvesti niti jedan slijed naredbi, proces je blokiran

```
if
:: (a == 3) -> option1
:: (b == 5) -> option2
fi
```

Odabir slučaja (2)

```
#define a 1
#define b 2
chan ch = [1] of { byte };

proctype A() { ch!a }
proctype B() { ch!b }
proctype C() {
  if
  :: ch?a
  :: ch?b
  fi
}
init { atomic { run A(); run B(); run C() }}
```

Odabir slučaja (3)

- Ako se može izvesti više sljedova naredbi, onaj koji će se izvesti se bira slučajnim odabirom

```
byte count;

proctype counter() {
  if
  :: count = count + 1
  :: count = count - 1
  fi
}
```

Petlje

```
byte count;

proctype counter() {
  do
  :: count = count + 1
  :: count = count - 1
  :: (count == 0) -> break
  od
}
```

Petlje – izlaz (1)

```
byte count;

proctype counter() {
  do
       :: (count!=0) ->
            if
                 :: count = count + 1
                 :: count = count - 1
            fi
       :: (count==0) -> break
  od
}
```

Petlje – izlaz (2)

```
byte count;

proctype counter() {
  do
       :: (count!=0) ->
            if
                 :: count = count + 1
                 :: count = count - 1
            fi
       :: (count==0) -> goto done
  od
done:
  skip
}
```

Primjer 1

- Napisati program koji filtrira poruke u kanalu
  - Vrijednosti veće od 128 se prenose kanalom `large` do primatelja A
  - Vrijednosti manje od 128 se prenose kanalom `small` do primatelja B
  - Nakon što primatelji prime poruke, ponovno ih šalju ispočetka

Primjer 1 – proces koji filtrira poruke

```
#define N 128
#define size 16
chan in = [size] of { short };
chan large = [size] of { short };
chan small = [size] of { short };

proctype split() {
  short cargo;
  do
  :: in?cargo ->
       if
       :: (cargo >= N) -> large!cargo
       :: (cargo < N) -> small!cargo
       fi
  od
```

Primjer 1 – procesi A i B

```
proctype A() {
  short cargo;

  do
  :: large?cargo; in!cargo
  od
}
proctype B() {
  short cargo;

  do
  :: small?cargo; in!cargo
  od
}
```

Primjer 1 – pokretanje procesa

```
init {
 in!345; in!12; in!677; in!32; in!0;
 run split();
 run A();
 run B()
}
```

Primjer 2 (1)

Sinkronizacija procesa pomoću semafora

```
chan sema = [0] of { bit }
proctype dijkstra() {
  byte count = 1;
  do
  :: sema!0
  :: sema?1
  od
}

proctype user() {
  do
  ::  sema?0; /* critical section */
       sema!1; /* non-critical section */
  od
}
```

Primjer 2 (2)

Sinkronizacija procesa pomoću semafora

```
chan sema = [0] of { bit }
proctype dijkstra() {
  byte count = 1;
  do
  :: (count == 1) -> sema!0; count = 0
  :: (count == 0) -> sema?1; count = 1
  od
}

proctype user() {
  do
  ::  sema?0; /* critical section */
       sema!1; /* non-critical section */
  od
}
```

Prekid procesa

- Istek vremena

```
proctype watchdog() {
  do
  :: ch?a
  :: timeout -> guard!reset
  od
}
```

- Izvršava se kad ostale naredbe u `do` bloku nisu izvršive
- Samostalno treba implementirati što činiti u takvom slučaju!
- `timer`, `set`, `expire`…

Laboratorij iz komunikacijskih protokola

Specifikacija, modeliranje i verifikacija protokola

1. Dio – Promela/Spin

modeliranje protokola u jeziku Promela (Protocol Modeling Language) i verifikacija modela primjenom alata Spin

Predavanje: 15.10.

2. Dio – CPN Tools

modeliranje protokola pomoću Petrijeve mreže, analiza mreže primjenom alata CPN (Colored Petri Nets) Tools

Predavanje: 22.10.

Podjela zadataka: 22.10.

Predaja rješenja: 15.11. - 16.11.

Laboratorij iz komunikacijskih protokola

- Predaja projekta
  - Predaja dokumentacije preko web-a (14.11.)
  - Demonstracija u labosu – 7. tjedan 1. ciklusa predavanja (15. - 16.11.)
    - Termini su objavljeni u *Nastavnim aktivnostima*
- Konzultacije
  - Uz prethodnu najavu na mail pavle.skocir@fer.hr dan ranije!
    - Uživo ili online (MS Teams)
    - Po dogovoru