PROMELA – programski jezik

Promela se sastoji od:

- procesa
- kanala
- varijabli

- izrazi i uvjetni izrazi

```
while (a != b)
        sleep;

bool zastavica;   // 0 ili 1
int stanje;
byte poruka;    //0-255
short nesto;

byte stanje[n];
stanje[0] = stanje[3] + stanje[5];
```

PROCESI

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

razdvajanje izraza: **;** ili **→**

Primjer:

```
byte stanje = 2;

proctype A() {
    (stanje == 1) → stanje = 3
}

proctype B() {
    stanje = stanje – 1;
}

run A();
run B();
```

```
init {
      run A();
      run B();
}
```
- inicijalni proces
- uvijek se izvodi prvi

Primjer:

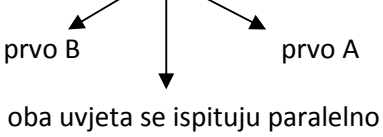
```
proctype A (byte stanje, short fcc) {
      (stanje == 1) → stanje = fcc
}

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

Primjer:

```
byte stanje = 1;

proctype A() {
      (stanje == 1) → stanje = stanje + 1
}

proctype B() {
      (stanje == 1) → stanje = stanje – 1
}

init { run A(); run B() }
```

svi procesi se pokreću paralelno (moguće je i da se niti jedan ne izvrši)

- varijable stanje može imati 3 različite vrijednosti: 0, 1, 2



oba uvjeta se ispituju paralelno

Primjer:

```
# define true 1
#define false 0
#define Aturn false
#define Bturn true

bool x, y, t;

proctype A(){
  x = true;  t = Bturn;
  (y == false  ||  t == Aturn);
-----------------------------------------------------
  x = false                K.O.
-----------------------------------------------------
}

proctype B() {
  y = true;
  t = Aturn;
  (x == false ||  t == Bturn);
-----------------------------------------------------
  y = false                K.O.
-----------------------------------------------------
}

init { run A();  run B() }
```

s ovime osiguravamo kritični odsječak

atomic { } - unutar izraza atomic nema prekida (zaštita kritičnog odsječka)

PROCESI KOMUNICIRAJU PORUKAMA PREKO KOMUNIKACIJSKOG KANALA

```
chan  qname = [16]  of  { short }
```

na kanalu može biti 16 poruka

tip poruke

```
chan qname [16] of {byte, int, byte}
                    (16, 8, 3)  ⟶  primjer poruke
```

qname ! izraz → na kanal qname ŠALJI (!) izraz

qname ? izraz → primi poruku s kanala qname i spremi je u varijablu izraz

- kanal radi na principu FIFO (first in – first out)

chan qname = [16] of { short }

qname ! var1, var2, var3

poruka se sastoji od više varijabli

qname ! var1(var2, var3) *identično je izrazu* qname ! var1, var2, var3

Primjer:

```
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
}
```

ne može se primiti nešto s kanala ako na kanalu nema ničega (prvo se mora poslati nešto na kanal)

~~(qname ? var == 0)~~ → ne može se ovako provjeravati, nego kao u donjem primjeru

qname ? [var1, var2] → provjera nepraznosti kanala

chan part = [0] of {byte}

sinkroni kanal
(ne može se čuvati poruku na kanalu nego mora odmah biti aktivan drugi proces koji prima poruku)

Primjer:

```
#define msqtype 33

chan name = [0] of {byte, byte}

proctype A() {
      name ! msqtype (124);
      name ! msqtype (121);
}

proctype B() {
      byte state;
      name ? msqtype (state)
}

init {
      atomic {run A();  run B(); }
}
```

msqtpye, 124
33,124

neće se nikada izvršiti;
kanal je sinkroni pa se na kanal može staviti poruka samo ako je neki drugi proces prima (u ovom slučaju proces B prima samo jednu poruku)

timeout → izvodi se kad se ništa drugo ne izvodi

KONTROLA TOKA

```
if
  :: (a != b) → opcija 1
  :: (a == b) → opcija 2
  :: else →
fi
```

-izvodi se prvo naredba koja je istinita;
-prva nije ona koja je u kodu napisana nego bilo koja istinita (simulator sam odluči koju treba izvesti – slučajni odabir);

- ako nema else dijela, a ni jedan uvjet nije ispunjen, čeka

Primjer:

```
#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()
	}
}
```

Primjer:

```
byte count;

proctype counter {
	if
	  :: count = count + 1
	  :: count = count – 1
	fi
}
```

50 % vjerojatnosti izvršavanja neke od navedenih naredbi

```
do
    :: count = count + 1
    :: count = count – 1
    :: (count == 0) → break
od
```

```
do
:: (count != 0) →
    if
      :: count = count + 1
      :: count = count – 1
    fi
:: (count == 0) → break    //goto end (end je labela)
od
```

Upute za labos:

1. **Kod**
2. **Check syntax**
3. **Simulation**
4. **Verification**