T-79.231 **Autumn 2001**
Parallel and distributed digital systems
Answers to Tutorial 5
Oct 23, 2001

1. Let us create the incidence matrix of the net (the columns indicate the changes in the place markings by transitions are)

$$C = \begin{pmatrix} -1 & 0 & 0 & 1 & 1 \\ 0 & -1 & 0 & 1 & 0 \\ 1 & -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \end{pmatrix}$$

The vector representation of the column ι is an invariant iff $C^T \cdot \iota = 0$, where 0 is a [column] vector.

a) Our task is to prove the following with invariants: $M(p_4) \leq 2$.
Let us make the guess that one of the invariants is $\iota = M(p_2) + M(p_4)$. Hence, we obtain the ι vector, where the multipliers for both places are 1. Let us check that our vector really is an invariant:

$$C^T \cdot \iota = \begin{pmatrix} -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 1 & 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

So $\iota = M(p_2) + M(p_4)$ really is an invariant. Then we will compute, using the initial marking M_0, the value of the invariant.

$$\iota^T \cdot M_0 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 5 \\ 2 \\ 0 \\ 0 \\ 0 \end{pmatrix} = 2$$

The result was $M(p_2) + M(p_4) = 2 \Rightarrow M(p_4) \leq 2$, which means that at most 2 processors can access the memory concurrently.

b) First we will prove that $\iota = M(p_1) + M(p_3) + M(p_4) + M(p_5)$ is an invariant. Let us check it:

$$C^T \cdot \iota = \begin{pmatrix} -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & -1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 1 \\ 1 & 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

As the next step we will compute the value of the invariant, again basing on the initial marking:

$$\iota^T \cdot M_0 = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} 5 \\ 2 \\ 0 \\ 0 \\ 0 \end{pmatrix} = 5$$

Hence we have proved the following formula to be the invariant: $M(p_1) + M(p_3) + M(p_4) + M(p_5) = 5$.

2. A `Maria` description of the TicketME algorithm:

```
int nprocs=2;
int NIL=0;

typedef int(1..nprocs) n_of_processes;
typedef enum {REM, ENTER, TEST, LEAVE_T, CRIT, RESET, LEAVE_E} process_state;
typedef struct {
  n_of_processes x,
  process_state y
} process_t;
typedef struct {
  n_of_processes x,
  n_of_processes y
} turn_t;
typedef struct {
  n_of_processes x,
  int(0..nprocs) y
} ticket_t;


place ids n_of_processes : n_of_processes x : x;
place turn turn_t : {1,1};
place procs process_t : n_of_processes x : {x,REM};
place tickets ticket_t : n_of_processes x : {x,NIL};

trans try
   in  { place procs: {pid,REM}; }
   out { place procs: {pid,ENTER}; };

trans enter
   in  { place procs: {pid,ENTER}; place turn: {next,granted};
         place tickets: {pid,tick}; }
   out { place procs: (next == granted)?{pid,LEAVE_T}:{pid,TEST};
         place turn: {+next,granted};
         place tickets: {pid,next}; };

trans testtick
   in  { place procs: {pid,TEST}; place turn: {next,granted};
         place tickets: {pid,tick}; }
   out { place procs:
         (is n_of_processes tick == granted)?{pid,LEAVE_T}:{pid,TEST};
         place turn: {next,granted};
         place tickets: {pid,tick}; };

trans crit
   in  { place procs: {pid,LEAVE_T}; }
   out { place procs: {pid,CRIT}; };

trans exit
   in  { place procs: {pid,CRIT}; }
   out { place procs: {pid,RESET}; };

trans reset
   in  { place procs: {pid,RESET}; place turn: {next,granted};
         place tickets: {pid,tick}; }
   out { place procs: {pid,LEAVE_E}; place turn: {next,+granted};
         place tickets: {pid,NIL}; };

trans rem
   in  { place procs: {pid,LEAVE_E}; }
```

```
out { place procs: {pid,REM}; };
```

(a) Let the above Maria description be in the file `TicketME.pn`. Let us present:

- The property as an LTL formula,
- The property as a Maria LTL formula,
- Is the property valid?
- An example of a model checking run.
- Two processes are never in the critical section concurrently:

$$\Box(\neg(procs(\langle 1, CRIT\rangle) \wedge procs(\langle 2, CRIT\rangle)))$$

  `[]!(is process_t ({1,CRIT},{2,CRIT}) subset place procs)`

  The property holds.

```
path 1 /lhome/jhonkola/ticket $./maria -d TicketME.pn
"TicketME.pn": 82 states, 164 arcs
@0$translator "./lbt"
@0$[]!(is process_t ({1,CRIT},{2,CRIT}) subset place procs)
(command line):2:property holds
@0$
```

- The process 1 never makes it to the critical section:

$$\Box(\neg(procs(\langle 1, CRIT\rangle)))$$

  `[]!(is process_t {1,CRIT} subset place procs)`

  The property does not hold.

```
@0$[]!(is process_t {1,CRIT} subset place procs)
(command line):2:constructing counterexample
(command line):2:counterexample path:
(command line):2: @0 @1 @3 @27 @56 @57 @57 @44 @45 @46 @47 @48 @48
@0$show @27
state (
 ids:
  1,2
 turn:
  {2,1}
 procs:
  {2,ENTER},{1,LEAVE_T}
 tickets:
  {2,0},{1,1}
)
3 predecessors
2 successors
@0$show @56
state (
 ids:
  1,2
 turn:
  {1,1}
 procs:
  {2,TEST},{1,LEAVE_T}
 tickets:
  {1,1},{2,2}
)
3 predecessors
```

```
2 successors
@0$show @57
state (
 ids:
  1,2
 turn:
  {1,1}
 procs:
  {2,TEST},{1,CRIT}
 tickets:
  {1,1},{2,2}
)
3 predecessors
2 successors
@0$path (is process_t {1,CRIT} subset place procs) @0
shortest path from @0 to condition (4 nodes):
 @0 @1 @26 @28
@0$path @0 (is process_t {1,CRIT} subset place procs)
shortest path from condition to @0 (4 nodes):
 @77 @78 @38 @0
@0$exit
path 2 /lhome/jhonkola/ticket $
```

- If the process 1 wants access to the critical section, it will make it there:

$$\Box(procs(\langle 1, ENTER\rangle)\implies \Diamond procs(\langle 1, CRIT\rangle))$$

```
[]((is process_t {1,ENTER} subset place procs) => \
<> (is process_t {1,CRIT} subset place procs))
```

The property does not hold in the model. Maria gives a counter example where the process 2 is continuously firing the transition `testtick` and the process 1 cannot make any progress.

```
path 2 /lhome/jhonkola/ticket $./prop_3
./prop_3:5:constructing counterexample
./prop_3:5:counterexample path:
./prop_3:5: @0 @1 @3 @27 @56
@0$show @27
state (
 ids:
  1,2
 turn:
  {2,1}
 procs:
  {2,ENTER},{1,LEAVE_T}
 tickets:
  {2,0},{1,1}
)
3 predecessors
2 successors
@0$show @3
state (
 ids:
  1,2
 turn:
  {1,1}
 procs:
```

```
  {1,ENTER},{2,ENTER}
 tickets:
  {1,0},{2,0}
)
2 predecessors
2 successors
@0$show @56
state (
 ids:
  1,2
 turn:
  {1,1}
 procs:
  {2,TEST},{1,LEAVE_T}
 tickets:
  {1,1},{2,2}
)
3 predecessors
2 successors
@0$succ @56
transition testtick->@56
<
 pid:2
 next:1
 granted:1
 tick:2
>
transition crit->@57
<
 pid:1
>
@0$
```

- If the process 2 is in the critical section, it will eventually leave the critical section:

$$\neg\Diamond\Box(procs(\langle 2, CRIT\rangle))$$

`!<>[](is process_t {2,CRIT} subset place procs)`

The property does not hold. Maria gives a counter example where the process 2 is in the critical section and the process 1 is continuously firing the transition `testick`.

```
@0$!<>[](is process_t {2,CRIT} subset place procs)
(command line):2:constructing counterexample
(command line):2:counterexample path:
(command line):2: @0 @1 @3 @27 @56 @57 @44 @45 @34 @35 @18 @19 @50
@0$show @50
state (
 ids:
  1,2
 turn:
  {2,2}
 procs:
  {1,TEST},{2,CRIT}
 tickets:
  {1,1},{2,2}
)
```

```
3 predecessors
2 successors
@0$succ @50
transition testtick->@50
<
 pid:1
 next:2
 granted:2
 tick:1
>
transition exit->@51
<
 pid:2
>
@0$exit
path 3 /lhome/jhonkola/ticket $
```

(b) Let us add a scheduler to the Maria description file `TicketMES.pn`, listed below:

```
int nprocs=2;
int NIL=0;

typedef int(1..nprocs) n_of_processes;
typedef enum {REM, ENTER, TEST, LEAVE_T, CRIT, RESET, LEAVE_E} process_state;
typedef struct {
  n_of_processes x,
  process_state y
} process_t;
typedef struct {
  n_of_processes x,
  n_of_processes y
} turn_t;
typedef struct {
  n_of_processes x,
  int(0..nprocs) y
} ticket_t;


place ids n_of_processes : n_of_processes x : x;
place turn turn_t : {1,1};
place procs process_t : n_of_processes x : {x,REM};
place tickets ticket_t : n_of_processes x : {x,NIL};
place scheduler int(0..nprocs) : NIL;

trans schedule
   in  { place ids: pid; place scheduler: NIL;}
   out { place ids: pid; place scheduler: pid;};

trans try
   in  { place procs: {pid,REM}; place scheduler: pid;}
   out { place procs: {pid,ENTER}; place scheduler: NIL;};

trans enter
   in  { place procs: {pid,ENTER}; place turn: {next,granted};
         place tickets: {pid,tick}; place scheduler: pid;}
   out { place procs: (next == granted)?{pid,LEAVE_T}:{pid,TEST};
         place turn: {+next,granted};
         place tickets: {pid,next}; place scheduler: NIL;};

trans testtick
   in  { place procs: {pid,TEST}; place turn: {next,granted};
         place tickets: {pid,tick}; place scheduler: pid;}
   out { place procs:
         (is n_of_processes tick == granted)?{pid,LEAVE_T}:{pid,TEST};
         place turn: {next,granted};
         place tickets: {pid,tick}; place scheduler: NIL;};

trans crit
   in  { place procs: {pid,LEAVE_T}; place scheduler: pid;}
   out { place procs: {pid,CRIT}; place scheduler: NIL;};

trans exit
   in  { place procs: {pid,CRIT}; place scheduler: pid;}
   out { place procs: {pid,RESET}; place scheduler: NIL;};

trans reset
   in  { place procs: {pid,RESET}; place turn: {next,granted};
         place tickets: {pid,tick}; place scheduler: pid;}
   out { place procs: {pid,LEAVE_E}; place turn: {next,+granted};
         place tickets: {pid,NIL}; place scheduler: NIL;};

trans rem
   in  { place procs: {pid,LEAVE_E}; place scheduler: pid;}
   out { place procs: {pid,REM}; place scheduler: NIL;};
```

The fairness assumption has to be included separately in each verifiable formula. The assumption is as follows:

$$(\Box\Diamond scheduler(\langle 1\rangle) \wedge \Box\Diamond scheduler(\langle 2\rangle))$$

which results in the following Maria LTL formula:

```
[]<>(1 subset place scheduler) && []<> (2 subset place scheduler)
```

Now we will be able to do the model checking of our formulas w.r.t. the fairness assumption:

- If the process 1 wants access to the critical section , it will eventually make it there:

$$\begin{aligned}&(\Box\Diamond scheduler(\langle 1\rangle) \wedge \Box\Diamond scheduler(\langle 2\rangle)) \implies \\ &(\Box(procs(\langle 1, ENTER\rangle) \implies \Diamond procs(\langle 1, CRIT\rangle)))\end{aligned}$$

```
([]<>(1 subset place scheduler) && []<> (2 subset place scheduler)) => \
        ([]((is process_t {1,ENTER} subset place procs) =>\
        (<>(is process_t {1,CRIT} subset place procs))));
```

The property holds with the fairness assumption:

```
path 3 /lhome/jhonkola/ticket $cat prop_5
#!./maria
graph "TicketMES";
translator "./lbt";
([]<>(1 subset place scheduler) && []<> (2 subset place scheduler)) => \
        ([](is process_t {1,ENTER} subset place procs) =>\
        (<>(is process_t {1,CRIT} subset place procs)));
path 4 /lhome/jhonkola/ticket $ls -la TicketMES.rg?
-rw-rw-r--   1 jhonkola jhonkola     2542 Oct 22 20:42 TicketMES.rga
-rw-rw-r--   1 jhonkola jhonkola     2995 Oct 22 20:42 TicketMES.rgd
-rw-rw-r--   1 jhonkola jhonkola     2048 Oct 22 20:42 TicketMES.rgh
-rw-rw-r--   1 jhonkola jhonkola     2624 Oct 22 20:42 TicketMES.rgp
-rw-rw-r--   1 jhonkola jhonkola     1476 Oct 22 20:42 TicketMES.rgs
path 5 /lhome/jhonkola/ticket $./prop_5
./prop_5:6:property holds
@0$exit
path 6 /lhome/jhonkola/ticket $
```

(c) The formula is valid in the model. A Maria example run is given below:

```
path 6 /lhome/jhonkola/ticket $cat prop_6
#!./maria
graph "TicketMES";
translator "./lbt";
(<>[]!(1 subset place scheduler) && (<>[]!(2 subset place scheduler))) =>\
      (<>(is process_t ({1,CRIT},{2,CRIT}) subset place procs));
path 7 /lhome/jhonkola/ticket $./prop_6
./prop_6:5:property holds
@0$exit
path 8 /lhome/jhonkola/ticket $
```

Also the following formula holds in the model:

$$(\Diamond\Box\neg scheduler(\langle 1\rangle) \wedge \Diamond\Box\neg scheduler(\langle 2\rangle)) \implies (1 = 0)$$

The formula is an example of a fairness assumption which does not hold for any execution. It claims that fair runs are the ones in which the scheduling of both the process 1 and 2 is terminated. Such runs do not exist in the model!