CS552: Models of Software Systems
Qualifying Examination

July 2014

1	/25
2	/ 15
3	/ 30
4	/ 30
Total	/ 100

* Total number of pages: 6 pages

1. Logic (25 points)

1. Let P be a predicate over elements of type T. Assume you can test elements for equality. Write predicates to capture the following English sentences.
   - There is exactly one element of T that satisfies P.

   - There are at least two elements of T that satisfy P.

2. Give English translations for each of the following assertions and explain how the two are related, assuming P is some predicate.
   (a) $\forall\, x \bullet \exists y \bullet P(x,y)$
   (b) $\exists y \bullet \forall x \bullet P(x,y)$

3. Give English translations for each of the following assertions and explain how the two are related, assuming P is some predicate.
   (a) $\Diamond\Box P$
   (b) $\Box\Diamond P$

2. Petri Nets (15 points)

Two processes, P and Q, each repeatedly perform a computation that requires a critical section. Additionally, P and Q can jointly (simultaneously) move to a termination state, but not while either one is in its critical section. Model the interaction of P and Q with a Petri Net. Be sure to specify the initial marking.

2 Z (30 Points)

In the following questions multiple interpretations may be possible. To be sure it is clear what interpretation you are adopting, you may find it helpful to state your assumptions. However, lengthy explanations are not needed.

Consider the problem of building a system to keep track of a sports league. Each league consists of a set of TEAMs, and each team has a number of PLAYERs that play for that team. One of the league rules is that no team may have more than 25 players associated with it. If we let *players* represent the set of PLAYERs that play on some TEAM in the League, the system might be partially specified by:

[PLAYER, TEAM]

Leage

$teams: \mathbb{F}\ TEAM$

$players: \mathbb{F}\ PLAYER$

$plays\_for: PLAYER \rightarrow\!\!\!\!\!\!\rightarrow TEAM$

---

dom *plays_for* = *players*

ran *plays_for* = *teams*

1. Complete the state invariant for this schema so that it is consistent with the description above.

2. Based on the invariant specified in part 1, answer the following short questions, and *briefly* justify.

(a) Can any of a League's set of *players* play for more than one team in that League?

(b) Can any of a League's set of *players* not be on any team?

(c) Can some of a League's set of *teams* have no players?

(d) Is it possible for there to be no TEAMs in a League?

4. SPIN (30 points)

Mr. Problema wanted to model and analyze a protocol in SPIN. He produced a Promela model with two channels and two processes, which contained the following declarations and statements.

```
mtype = {m0, m1, ack}

proctype Sender
{    (* Code for sender *)
}
proctype Receiver
{    (* Code for Receiver *)
}
init
{    chan StoR = [5] of {mtype};
     chan RtoS = [5] of {mtype};
     atomic {  StoR! m0;
               run Sender;
               run Receiver
     }
}
```

He found that the model was too big (too many reachable states). He then wanted to make a more abstract model, where instead of representing the state of a channel by a sequence of messages, the model only remembers how many messages are in the channel. He then replaced the above declarations by

```
mtype = { m0, m1, ack } ;
byte  StoR, RtoS ;

proctype Sender
{    (* Modified Code for Sender *)
}
proctype Receiver
{    (* Modified Code for Receiver *)
}
init
{    atomic {  StoR = 1 ;
               run Sender ;
               run Receiver
     }
}
```

In the code for the Sender and for the Receiver, he modified send and receive statements as follows:

- A send statements of form **`StoR! m0`** is replaced by **`StoR = StoR + 1`**
- A receive statement of form **`StoR? m0`** is replaced by
  `StoR = (StoR > 0 -> (StoR – 1) : StoR)`
- Analogously for send and receive statements to other channels and with other messages.

(1) What is wrong with the above abstraction?

(2) How should it be done properly?