Design and Analysis of Distributed Interacting Systems
Summerterm 2013
Assignment 3

Note: Solutions must be submitted in groups of two or three students. Submissions by just one student will not be graded!

To be submitted on May 9, 2013, 8am via e-mail.

Submissions and further questions to Raphael Pham:

Raphael.Pham@inf.uni-hannover.de

Exercise 1. Understand LTL Properties (6 Marks)
Textually describe the meaning of the following LTL formulae (from the coffee machine example in the lecture). Describe the meaning as intuitively as possible, but not at the expense of being imprecise. Furthermore, sketch two example runs of a Kripke structure for each formula, one that satisfies the formula and one that does not.

a. $G(money\_inserted \Rightarrow F(money\_back \lor tea\_delivered \lor coffee\_delivered))$

b. $G(money\_inserted \Rightarrow (F coffee\_chosen \Rightarrow (F coffee\_delivered)))$

c. $G(\neg(cleaning \land money\_inserted) \lor X money\_back)$

Exercise 2. Specify LTL Properties (4 Marks)
Give an LTL formula corresponding to each of the following textual statements. The set of atomic propositions is $AP = \{p, q, r\}$.

a. Whenever $p$ holds, $q$ must not hold.

b. $p$ and $q$ must never hold at the same time.

c. If $p$ and $q$ together hold infinitely often, also $r$ must hold infinitely often.

d. If from some point on $p$ holds always, then $q$ must hold eventually.
Exercise 3. Equivalences of LTL Properties (6 Marks)
Consider the following definition of **equivalence** of LTL formulae.

**Definition 1.** Two LTL formulae $\varphi$ and $\psi$ are said to be equivalent, written $\varphi \equiv \psi$, iff for all Kripke structures $M$ we have $M \models \varphi \iff M \models \psi$.

Prove or disprove the statements below. (Hint: Equivalence can be shown on the basis of any possible run of any Kripke structure $\pi$, so take a look at Def. 3 in the lecture slides. If an equivalence in not correct, giving a counter-example is sufficient.)

a. $F \varphi \equiv true \ U \varphi$

   (Sample solution:)

   *Proof.* We consider any run of any Kripke structure $\pi$ and show that $\pi \models F \varphi \iff \pi \models true \ U \varphi$ holds. This proves that $F \varphi \equiv true \ U \varphi$.

   \[
   F \varphi \iff \exists i \geq 0 : \pi^i \models \varphi \\
   \iff \exists i \geq 0 : \pi^i \models \varphi \\
   \land \ \forall j, 0 \leq j < i : \pi^j \models true \\
   \iff \pi \models true \ U \varphi
   \]

   (by Def. $F$)

   (true holds in all states)

   (by Def. $U$)

   \[\square\]

b. $\neg G \varphi \equiv F \neg \varphi$

c. $F(\varphi \land \psi) \equiv F \varphi \land F \psi$

d. $F(\varphi \lor \psi) \equiv F \varphi \lor F \psi$

Exercise 3. First steps with the SPIN model-checker (4 Marks)
Download and install the Spin model checker and the iSpin graphical user interface (see \[\text{http://spinroot.com/spin/Man/README.html}\]). To run Spin, you need a C-compiler, and TCL/TK is required to run iSpin.

There are some videos describing the installation if MinGW, TCL/TK, Spin, iSpin on Youtube:

1. \[\text{http://www.youtube.com/watch?v=vc7JRspeYUo}\]
2. \[\text{http://www.youtube.com/watch?v=hHJwSS3tpFQ}\]
3. \[\text{http://www.youtube.com/watch?v=NB3N1g2FvCw}\]

Start iSpin and load the light switch model `light_simple.pml`, which is also shown in the listing below. This model corresponds to the light switch example that was shown in the lecture previously to introduce other modeling formalisms.
mtype = {press, hold};

chan c = [0] of { mtype };

active proctype light(){
    OFF:
        if :: c?press; goto LOW
        fi;
    LOW:
        if :: c?press; goto OFF
        :: c?hold; goto HIGH
        fi;
    HIGH:
        if :: c?press; goto OFF
        :: c?hold; goto LOW
        fi;
}

active proctype button(){
    RELEASED:
        if :: c?press; goto PRESSED
        fi;
    PRESSED:
        if :: c?hold; goto PRESSED
        :: goto RELEASED
        fi;
}

Now perform the following steps.

1. Try to understand the Promela code. Refer to the manual [http://spinroot.com/spin/Man/Manual.html](http://spinroot.com/spin/Man/Manual.html) or other resources.

2. Simulate the model: In iSpin, go to the tab “Simulate / Replay” and hit “(Re)Run” and observe what happens. (You can stop the execution after a while.) Re-run the simulation using the mode “Interactive (for resolution of all non-determinism)”.

3. In the “Edit/View” tab, change line 3 of the code to

   chan c = [1] of { mtype };

   (change the “0” to “1”) and save the change.

4. Simulate the model again using the random mode and observe what happens.

5. Switch to the “Verification” tab. Make sure that “safety” and “invalid end-states” is selected and hit “Run”. In the output generated in the console, you should see a line “pan:1: invalid end state (at depth 6)” (or similar); moreover, the last statement in the output console should say “To replay the error-trail, goto Simulate/Replay and select "Run".”
6. Do the latter, i.e., again go to tab “Simulate / Replay” and hit “(Re)Run”, and try to understand what is being executed.

Now answer the following questions:

1. What does the change above in Step 3 mean?

2. Explain the “invalid end state” that the verification shows and how the program execution ends up in this state. Why does the model before the change not have a deadlock state? Supply the .trail file created by the verification with your submission.