Introduction to Model Checking
Exercises WS 2009/2010

René Thiemann

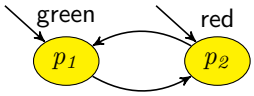
Institute of Computer Science

December 10, 2009

1. Formalize the following requirements in LTL over $AP = \{\mathsf{red}, \mathsf{orange}, \mathsf{green}\}$. (increasing difficulty)
   (a) The traffic light is never red and green at the same time.
   (b) Under the assumption that the traffic light is orange infinitely often, it is green infinitely often and red infinitely often.
   (c) The sequence of lights is exactly $(\mathsf{red}, \mathsf{red}\,\mathsf{orange}, \mathsf{green}, \mathsf{orange})^\omega$.
   (d) Whenever the traffic light shows red, at some moment strictly before, both red and orange have been shown.
   (e) The sequence of lights is of the form $(\mathsf{red}^+\,\mathsf{green}^+\,\mathsf{orange}^+)^\omega$.
   (f) The sequence of lights is of the form $(\mathsf{red}^+\,\mathsf{orange}^+\,\mathsf{green}^+\,\mathsf{orange}^+)^\omega$.
2. Prove the following equivalences
   - $\mathsf{G}\,(\varphi \wedge \psi) \quad \equiv \quad \mathsf{G}\,\varphi \;\wedge\; \mathsf{G}\,\psi$
   - $\mathsf{G}\,\mathsf{F}\,\mathsf{G}\,\varphi \quad \equiv \quad \mathsf{F}\,\mathsf{G}\,\varphi$
   - $\varphi\,\mathsf{U}\,(\varphi\,\mathsf{U}\,\psi) \quad \equiv \quad \varphi\,\mathsf{U}\,\psi$
3. Let $\Sigma = \{A, B, C\}$. Construct GNBAs for the languages:
   - $\{w \mid w$ contains only finitely many A's or infinitely many B's$\}$
   - $\{w \mid w$ starts with an even number of A's followed by a $B\}$
   - $\{w \mid w$ contains infinitely many A's and between every two A's there is an odd number of B's$\}$
4. Show that GNBAs are closed under union, i.e., given two GNBAs $\mathcal{A}_1$ and $\mathcal{A}_2$ over the same alphabet Σ, construct a new GNBA $\mathcal{A}$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$.
5. For each of the requirements in Exercise 1 build a corresponding GNBA that accepts the allowed behaviours.

6. Prove the soundness of the automaton for the intersection of two NBAs. To be more precise, you have to show for an arbitrary ω-word w that $w \in \mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2)$ iff $w \in \mathcal{L}(\mathcal{A}_{\mathcal{A}_1 \cap \mathcal{A}_2})$.

7. Use the algorithm depicted on Slide 29 of Chapter 3 to determine $TS \models \varphi$ where $\varphi = \mathsf{G}\,\neg(\mathsf{red} \wedge \mathsf{green})$ and TS is the following transition system. Here, you should perform the computation of $\mathcal{A}_{\neg\varphi}$ and of the intersection automaton lazily, i.e., only construct the reachable part of the intersection automaton and only construct those parts of $\mathcal{A}_{\neg\varphi}$ that are needed!



8. Compute the GNBA for the LTL formula $\mathsf{G}\,(\mathsf{req} \Rightarrow \mathsf{F}\,\mathsf{X}\,\mathsf{resp})$ using the improved translation (Slide 42, Chapter 3) where $cl' = \mathsf{req}, \mathsf{resp}, \ldots$. To this end first define the GNBA symbolically and then determine all outgoing transitions of state $(0,0,0,0,1)^T$.

9. Confirm yourself that it is not a good idea to use asynchronous communication for the traffic lights. To this end, create the transition system for the channel system

$$[\mathit{TrafficLight} \mid \mathit{TrafficLight} \mid \mathit{Starter}]$$

where c has a buffer of size 1.

10. The following nanoPromela code models the alternating bit protocol.

```
-------------  SENDER  -------------
atomic { b := 0; snd := true } ;
do
  :: snd     =>  if :: true => c ! <m,b>
                    :: true => skip
                 fi ;
                 tmr_on ! dummy ;
                 snd := false;
  :: not snd => if :: true => timeout ? y ;
                              snd := true
                :: true => d ? x ;
                           if :: x = 1-b => skip
                              :: x = b   => tmr_off ! dummy ;
                                            snd := true;
                                            b := 1 - b
                           fi
              fi
od
```

```
------------ RECEIVER -------------
b := 0;
do
  :: true => c ? <m,y> ;
              if :: y = 1-b => skip
                 :: y =  b  => d ! b;
                               b := 1-b
              fi
od

--------------  TIMER  -------------
do
  :: true => tmr_on ? x ;
              if :: true => timeout ! dummy
                 :: true => tmr_off ? x
              fi
od
```

Construct the program graphs for these statements according to the formal semantics. For the timer process also derive the transitions by building an inference tree as demonstrated during the lecture on Slide 44 of Chapter 4.

11. Consider 3 processes (with separate registers r) which each increase a shared variable x by 1 for 6 times.

```
LOOP 6 TIMES
  LOAD x INTO r
  INCR r
  STORE r INTO x
```

- Model these processes in Spin.
- Determine whether it is possible that the variable has value 2 when all processes have been executed, assuming that the value of x is initially 0. To this end, formalize a corresponding property in LTL and let Spin determine the answer.

  (To see how Spin is used, cf. Chapter 5)