Introduction to Model Checking
Exercises WS 2010/2011

René Thiemann

Institute of Computer Science

December 3, 2010

1. Consider the requirements 1–4 on Slide 27, Chapter 2. For each of these requirements construct a GNBA which accepts the ω-words that violate the requirement.

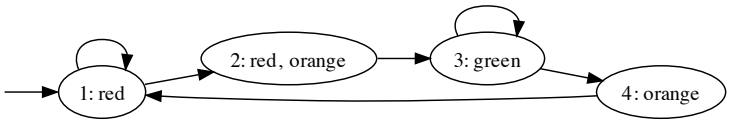
2. Let $\Sigma = \{A, B, C\}$. Construct GNBAs for the languages:

   - $\{w \mid w$ 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$\}$

3. 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 $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$. You only have to give a precise (formal) construction, a correctness proof is not required.

4. 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 $w \in \mathcal{L}(\mathcal{A}_{\mathcal{A}_1 \cap \mathcal{A}_2})$.

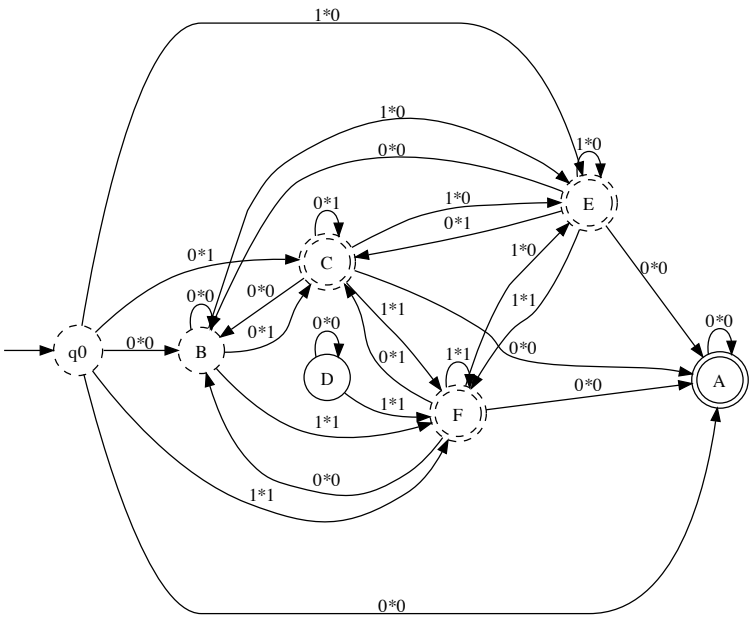
5. Consider the following transition system for a traffic light over $AP = \{\mathsf{red}, \mathsf{orange}, \mathsf{green}\}$:



   (a) Construct a GNBA to specify the forbidden traces w.r.t. the requirement "there is no direct switch from red to green". Does the traffic

light system satisfy this requirement? Do you see any problems with this requirement?

(b) For another requirement the following GNBA was automatically generated which specifies the set of forbidden traces.



Here, the alphabet consists of vectors $(red, orange, green)$. Moreover, the double-circled states form one final state set ($\{A, C, E, F\}$) and another final state set is given by the solid lined states ($\{A, D\}$).

- Decide whether the traffic light systems satisfies the requirements using the algorithms of Chapter 2. Here, for the intersection GNBA, only construct the reachable states!
- Optional: Describe the requirement in words.

6. 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) The sequence of lights is of the form $(\mathsf{red}^+\,\mathsf{green}^+\,\mathsf{orange}^+)^\omega$.

   (e) The sequence of lights is of the form $(\mathsf{red}^+\,\mathsf{orange}^+\,\mathsf{green}^+\,\mathsf{orange}^+)^\omega$.

7. Prove the following equivalences

   - $\mathsf{F}\,\mathsf{X}\,\varphi \quad \equiv \quad \mathsf{X}\,\mathsf{F}\,\varphi$
   - $\mathsf{G}\,\mathsf{F}\,\mathsf{G}\,\varphi \quad \equiv \quad \mathsf{F}\,\mathsf{G}\,\varphi$

8. Consider the LTL-formula

   $$\mathsf{G}\,(\mathsf{green} \Rightarrow (\neg\mathsf{orange} \wedge \mathsf{X}\,(\mathsf{green}\,\mathsf{U}\,\mathsf{orange})))$$
   $$\equiv \neg(\mathrm{true}\,\mathsf{U}\,(\mathsf{green} \wedge \neg(\neg\mathsf{orange} \wedge \mathsf{X}\,(\mathsf{green}\,\mathsf{U}\,\mathsf{orange})))) = \psi$$

   Compute $\mathcal{A}_\psi$ by using the improved translation where $cl'(\varphi) = \mathsf{green}, \mathsf{red}, \ldots$:

   (a) compute the transition function symbolically in the form of constraints

   (b) compute the sets of final states symbolically

   (c) compute all outgoing transitions of state $(0,0,0,0,1)^T$ explicitly and for each of the states you encounter determine to which final state sets it belongs

9. Consider 3 processes (where each process has its own register r) which each increases 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 x 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.

10. Show that LTL model checking is in PSPACE. Here, you may use the fact that PSPACE = NPSPACE. (This equality implies that it suffices to give a non-deterministic algorithm for the question "$TS \not\models \varphi$" which only requires polynomial space.)

    Only give a construction without the proof.

11. 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.

    (Study slides 29 and 30 of Chapter 5 to see the missing inference rules for asynchronous communication.)

12. 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 5.