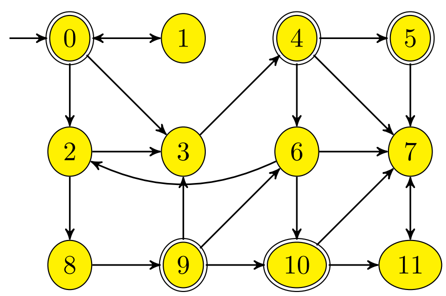
Exercises for the Lecture *Model Checking* (703521, SS09)

May 5, 2009

1. Consider the following NBA $\mathcal{A}$ (where input letters are omitted).

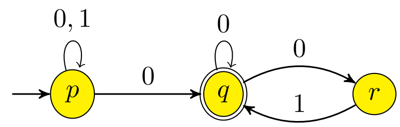


   Apply the linear on-the-fly algorithm to check $\mathcal{L}(\mathcal{A}) = \varnothing$. In the algorithm successors with small numbers should be taken first. Which accepting run is detected by the algorithm? Which states are marked, which are flagged?

2. Translate the formula $(a \mathsf{U} b) \mathsf{U} \mathsf{X} (a \wedge b)$ into F1S.

3. Recall the construction of the S1S formulas $\varphi_{\mathcal{A}}$ for some NBA $\mathcal{A}$. Currently it uses m additional second-order variables $\mathsf{b}_1, \ldots, \mathsf{b}_\mathsf{m}$ where m is the number of states of $\mathcal{A}$. Improve this construction and show that $log(m)$ additional variables suffice. You can assume that the number of states is a power of 2.

4. Construct the S1S-formula $\varphi_{\mathcal{A}}(\mathsf{a})$ for the following NBA $\mathcal{A}$ using the construction from the lecture.



5. Show that every S1S$_0$-formula can be translated into S1S (only give the construction).

6. Give F1S- or S1S-formulas for the following languages. Try to use F1S whenever possible.

   - $\begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix}^* \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right)^\omega$
   - $\left(\begin{pmatrix} 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right)^* \left(\begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right)^\omega$

7. Transform the following formula to S1S$_0$ using the transformation of the lecture.
$$\forall x : 0 < x \vee \mathsf{c}(x)$$

8. Consider the NBA $\mathcal{A}$ of Exercise 4.

   - Compute the $\mathcal{A}$-equivalence classes $U_1, \ldots, U_n$ by giving their shortest representatives and the corresponding transition profiles.
   - For each of the words $w_1 = 101001000100001000001\ldots$ and $w_2 = 011011011011011\ldots$ find U_i and U_j such that $w_{1/2} \in U_i \cdot U_j^\omega$.

9. Transform the following S1S$_0$-formula to an NBA using the construction of the lecture.
$$\exists \mathsf{c} : \mathsf{succ}(\mathsf{c}, \mathsf{a}) \vee \mathsf{a} \subseteq \mathsf{b}$$

10. Construct an NBA for the language $b(aa)^* \cdot (bb^* + a)^\omega$ using the construction from the lecture. Start with constructing NFAs for $b(aa)^*$ and $bb^* + a$ intuitively.

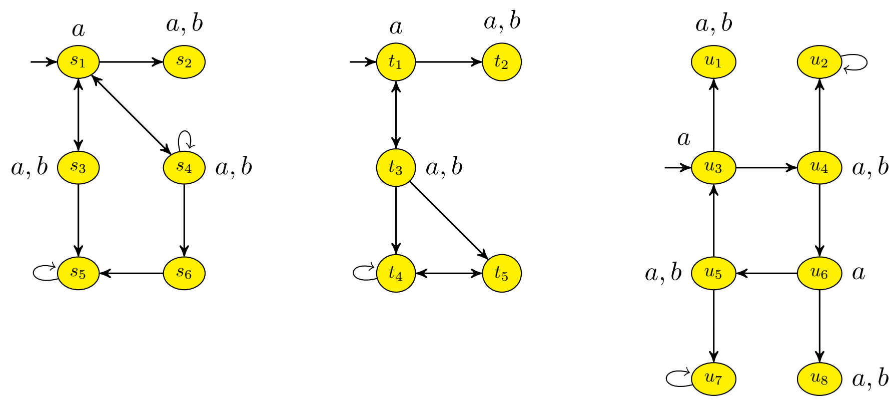
11. Prove $\sim_{TS} \subseteq \equiv_{CTL^*}$ by showing the following statements.

    - If $s \sim_{TS} t$ then for all CTL*-state-formulas Φ: $s \models \Phi$ iff $t \models \Phi$
    - If $\pi \sim_{TS} \pi'$ then for all CTL*-path-formulas ψ: $\pi \models \psi$ iff $\pi' \models \psi$

12. In the lecture, $\equiv_{CTL}$ is a relation between states. One can also interpret $\equiv_{CTL}$ as a relation between transition systems, where $TS_1 \equiv_{CTL} TS_2$ iff TS_1 and TS_2 satisfy the same CTL-formulas. Prove or disprove:
$$\equiv_{CTL} \subseteq \sim$$
(every two non-bisimilar systems can be distinguished by a CTL formula)

13. Prove that every system is bisimilar to its quotient, i.e., $TS \sim (TS/\sim)$.

14. Use the partitioning algorithm to decide $TS_i \sim TS_j$ for $i \neq j$ for the following transition systems where $TS_1 = (\{s_1, \ldots, s_6\}, \ldots)$, $TS_2 = (\{t_1, \ldots, t_5\}, \ldots)$, and $TS_3 = (\{u_1, \ldots, u_8\}, \ldots)$. In case of $TS_i \not\sim TS_j$ try to find a CTL-formula which can distinguish the systems (cf. Exercise 12). Note, that one can easily extend the CTL-semantics to transition systems with terminal states, e.g., $\mathsf{E}\, a \,\mathsf{U}\, b$ means that there is some finite or infinite path such that some state in this path satisfies b and all states before satisfy a. Moreover, $\mathsf{A}\,\mathsf{X}\,\varphi$ is always true in a terminal state whereas $\mathsf{E}\,\mathsf{X}\,\varphi$ is never satisfied in a terminal state.



15. Consider the three transition systems from the previous exercise. Decide for all $i \neq j$ whether $TS_i \preceq TS_j$. Of course, you can use the results of the previous exercise.

16. Explicitly construct the complete timed automaton for the train-controller-gate example, i.e., construct

$$(\mathit{Train} \,||_{H_1}\, \mathit{Controller}) \,||_{H_2}\, \mathit{Gate}$$

where $H_1 = \{\text{approach,exit}\}$ and $H_2 = \{\text{lower,raise}\}$.

17. Prove or disprove: $||_H$ is associative, i.e.,

$$TA_1 \,||_{H_1}\, (TA_2 \,||_{H_2}\, TA_3) = (TA_1 \,||_{H_1}\, TA_2) \,||_{H_2}\, TA_3$$

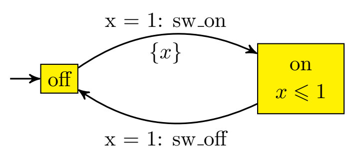
Of course, in the resulting timed automata, one has to identify the locations $(\ell_1, (\ell_2, \ell_3))$ with $((\ell_1, \ell_2), \ell_3)$ where one just can write (ℓ_1, ℓ_2, ℓ_3).

18. Four robbers are on a nightly escape and have to pass a small bridge. The bridge can only carry two robbers at the time and it is necessary to use a flashlight to pass the bridge. Unfortunately, the robbers only possess one

flashlight with batteries which last for one hour. Moreover, the robbers need different times to pass the bridge (5, 8, 25, and 31 minutes). Of course, if two of the robbers pass the bridge then they will need the time of the slower robber. The question now is, whether all four robbers can pass the bridge within the night.

- Model the question as a TCTL-model checking problem where you have to provide both the timed automaton and the formula.
- Answer the question and provide evidence in form of a compressed path.

19. Consider the following timed automaton *TA*.



In the lecture, $RTS(TA, \dots)$ and parts of $RTS(TA \uplus \{z\}, \dots z < 1 \dots)$ have been constructed (Slides 42 and 46) which were sufficient to prove $TA \not\models \mathsf{A}\,\mathsf{G}^{<1}\text{on}$.

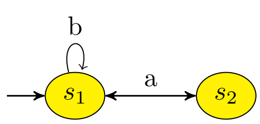
Compute $Sat(\mathsf{A}\,\mathsf{G}^{<1}\text{on}) \subseteq \{A, \dots, G\}$ where $A, \dots, G$ are the states of $RTS(TA, \dots)$, cf. Slide 42. For this, you should construct the reachable part of $RTS(TA \uplus \{z\}, \dots z < 1 \dots)$ starting from all states that are required to determine $Sat(\mathsf{A}\,\mathsf{G}^{<1}\text{on})$.

20. Prove $[\![\neg\nu x.\varphi]\!]_\alpha = [\![\mu x.\neg\varphi[x/\neg x]]\!]_\alpha$, one essential equality which is needed to ensure soundness of the transformation to PNF.

    Hint: Expand $[\![.]\!]$ as much as possible and perform induction over the number of fixpoint-iterations, i.e., over the n in $\tau^n(\dots)$. A graphical interpretation of the fixpoint-iteration might also be helpful (using set diagrams).

21. Perform μ-calculus model checking for the following example.

$$\varphi = \mu x.\langle a\rangle\neg\mu y.((\neg x \vee y) \wedge [b]y)$$



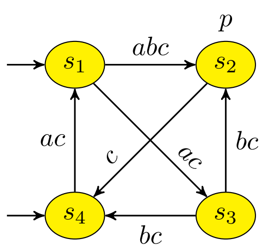
    (a) Transform φ into an equivalent formula ψ in PNF. What is the alternation depth of ψ? (Note that φ only contains μ-operators)

    (b) Determine $TS \models \varphi$ using the naive model checking algorithm.

(c) Determine $TS \models \varphi$ by checking $TS \models \psi$ with the algorithm of Emerson and Lei.

22. Perform model-checking for the following transition system and the formula $\nu x.\varphi_x$ using the algorithm of Emerson and Lei where

$$\begin{aligned}\varphi_x &= (\nu y.\varphi_y) \wedge \mu z.\varphi_z \\ \varphi_y &= [a](x \wedge y) \\ \varphi_z &= (\nu w.\varphi_w) \vee \langle b \rangle (z \wedge x) \\ \varphi_w &= \langle c \rangle w \wedge \neg p\end{aligned}$$



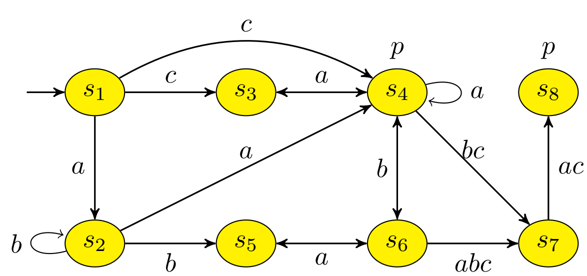
23. Currently the algorithm of Emerson and Lei only accepts formulas in PNF. If it would accept arbitrary closed formulas, it is straight-forward to define $\mathsf{sem}(\neg\varphi) = \textbf{return } S \setminus \mathsf{sem}(\varphi)$.

    Figure out why this algorithm is unsound. (Exercise 21 may be helpful.)

24. Prove the variant of the theorem of Knaster & Tarski.

25. Perform model-checking using the bottom-up and the top-down coloring algorithms for the following transition system and formula.

$$\varphi = \mu x.\Big(\big(\big([a]\nu y.(\neg p \wedge [b]y)\big) \wedge [b]x \big) \vee \big([c]\nu z.p \wedge \langle a \rangle z\big) \Big)$$



26. Explain how one can extract the positional winning strategies of ∃loise and ∀belard from the bottom-up coloring algorithm. (You do not have to prove that your extracted strategy really is a winning strategy.)