COMP2620/6262 (Logic) Tutorial

Week 11

Semester 1, 2025

Tutorial Quiz

In each tutorial, apart from week 2, there is a short quiz on skills practised in the previous tutorial. Your top 7 quiz attempts, out of the 9 available, will collectively count for 50% of your final mark.

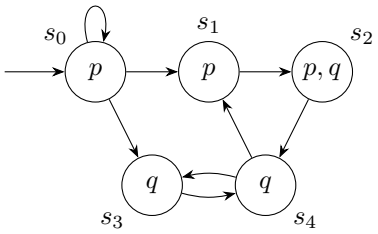
This week's quiz is on **modelling English into CTL***. Your tutor will hand out blank paper, on which you should clearly write your university ID and name. Your tutor will also hand out paper with an English language description of current and future states of a situation. You will translate each line of the description into a CTL* proposition. You must use CTL* syntax - not LTL or CTL, even if you feel that an LTL or CTL proposition could be used. Where you believe the English language description is ambiguous, you will be asked to present what you think is the most plausible translation, and then briefly discuss what other translations might be reasonable. You will have **eight minutes** to do this.

You are not permitted to have any other resource on the table during this quiz, including any electronic device. If you finish your quiz before time elapses you may put your hand up and your tutor will collect your sheet. Once you have done this, you may get a device out and start work silently on this week's questions. If you are still working when time elapses you must stop writing immediately and let your tutor collect your paper.

There will be no tutorial next week, and so no quiz.

This Week's Exercises

1. Model check the following CTL propositions against the transition system below. You should label all states.



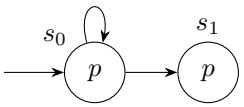
   - $EX(p \wedge EXq)$
     Solution. $\mathbf{s_0}$: $p, EXq, p \wedge EXq, EX(p \wedge EXq)$; $\mathbf{s_1}$: $p, EXq, p \wedge EXq, EX(p \wedge EXq)$; $\mathbf{s_2}$: $p, q, EXq, p \wedge EXq$; $\mathbf{s_3}$: q, EXq; $\mathbf{s_4}$: $q, EXq, EX(p \wedge EXq)$.
   - $E[pU \neg p \wedge EXp]$
     Solution. $\mathbf{s_0}$: $p, EXp, E[pU \neg p \wedge EXp]$; $\mathbf{s_1}$: $p, EXp, E[pU \neg p \wedge EXp]$; $\mathbf{s_2}$: $p, q, E[pU \neg p \wedge EXp]$; $\mathbf{s_3}$: $q, \neg p$; $\mathbf{s_4}$: $q, \neg p, EXp, \neg p \wedge EXp, E[pU \neg p \wedge EXp]$.
   - $EG\neg EG(p \wedge \neg q)$
     Solution. $\mathbf{s_0}$: $p, \neg q, p \wedge \neg q, EG(p \wedge \neg q)$; $\mathbf{s_1}$: $p, \neg q, p \wedge \neg q, \neg EG(p \wedge \neg q), EG\neg EG(p \wedge \neg q)$; $\mathbf{s_2}$: $p, q, \neg EG(p \wedge \neg q), EG\neg EG(p \wedge \neg q)$; $\mathbf{s_3}$: $q, \neg EG(p \wedge \neg q), EG\neg EG(p \wedge \neg q)$; $\mathbf{s_4}$: $q, \neg EG(p \wedge \neg q), EG\neg EG(p \wedge \neg q)$.

2. Identify the Strongly Connected Components (SCCs) in the transition system above. Are any of them trivial?

If you did not do so already, solve the final part of question 1 using the more efficient SCC method (if you did use that method, you could instead try out the simpler deletion-based method).

Solution. s_0 on its own is one SCC; all the other states together form the only other SCC. Neither is trivial: although s_0 forms an SCC on its own, it has a self loop.

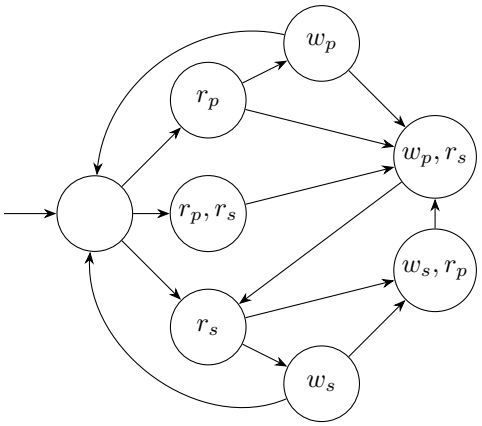
For the SCC-based model check for $EG(p \wedge \neg q)$, we first consider the subgraph of states that satisfy $p \wedge \neg q$:



s_0 and s_1 each form one element SCCs, but only the s_1 one is trivial. So we label s_0 with $EG(p \wedge \neg q)$ and do not label s_1 so, because it has no path to a non-trivial SCC.

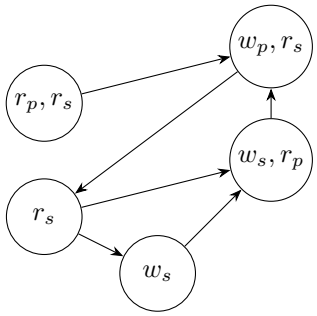
For the model check for $EG\neg EG(p \wedge \neg q)$ we disregard s_0. Then the whole graph is a single SCC so everything in that graph gets the label.

3. This system models a priority process and a secondary process which both need access to some non-shareable resource. Requests come in for these processes (r_p, r_s) and they can do work (w_p, w_s), but the priority process takes precedence, with requests for it even making the secondary process pause if it has started its work. We assume that after each process finishes work it must be inactive for one step to check whether it has any pending requests.



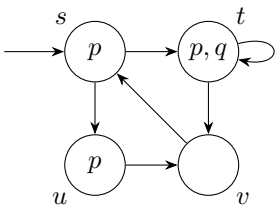
- State a CTL proposition that means that, if the secondary process starts work, it will eventually become inactive (neither working nor holding a pending request).
  Solution. $AG(w_s \rightarrow AF(\neg w_s \wedge \neg r_s))$
- Give a reason (informally) that this proposition fails.
  Solution. We can get stuck in a loop from w_s, r_p (the secondary process is working but a priority request has been recieved) to w_p, r_s (the secondary process is paused to do the priority work) to r_s (the priority work is finished) then back to w_s, r_p (the secondary process restarts, but another priority request is detected). We never reach a state satisfying $\neg w_s \wedge \neg r_s$. The situation is the same if we sometimes include w_s in our loop between r_s and w_s, r_p.
- Perhaps it is not plausible that the primary process would be always active, or just about to become active. Model check the CTL proposition from the first part of this question, replacing each A by A_C and E by E_C, where C is the single fairness constraint $\{\neg w_p \wedge \neg r_p \wedge \neg EX r_p\}$.
  Solution. We first need to translate (the fair version of) our proposition into the set of connectives that we learned to model check. By standard conversions this is equivalent to
  $\neg E_C F(w_s \wedge E_C G\neg(\neg w_s \wedge \neg r_s))$
  We can express EF via EU (recalling $\top$ is any theorem, e.g. $\neg\bot$) in the usual way:
  $\neg E_C[\top\, U\, w_s \wedge E_C G\neg(\neg w_s \wedge \neg r_s)]$
  We finally desugar our $E_C U$ to get:
  $\neg E[\top\, U\, w_s \wedge E_C G\neg(\neg w_s \wedge \neg r_s) \wedge E_C G\top]$
  Now we begin to label:

– The whole system is an SCC, but none of them satisfy the constaint, so we label $E_C G \top$ nowhere.
– $\neg(\neg w_s \wedge \neg r_s)$ is written anywhere with either w_s or r_s.
– For $E_C G \neg(\neg w_s \wedge \neg r_s)$ we look at the subgraph



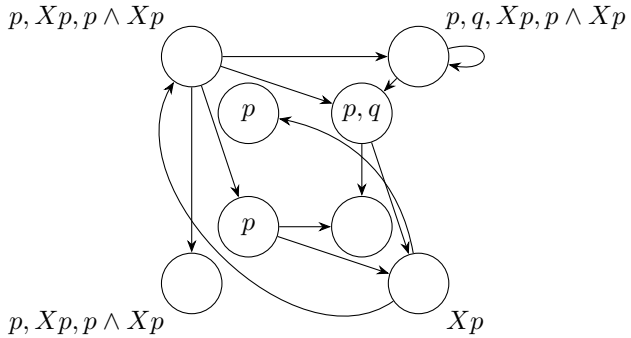
We have two SCCs: a trivial one with r_p, r_s and one with the other states. But $\{\neg w_p \wedge \neg r_p \wedge \neg EX r_p\}$ fails throughout the non-trivial SCC, as for example both the lower states are labelled $EX r_p$. So there is no fair SCC, so we label $E_C G \neg(\neg w_s \wedge \neg r_s)$ nowhere.
– We therefore write $E_C G \neg(\neg w_s \wedge \neg r_s) \wedge E_C G \top$ nowhere, so write $E[\top \, U \, w_s \wedge E_C G \neg(\neg w_s \wedge \neg r_s) \wedge E_C G \top]$ nowhere, so its negation everywhere, so our model check succeeds.

4. Model check the following LTL propositions against the transition system below. You should label all states in your expanded transition system. You will need to convert som of the propositions to use only the set of connectives that we learned to model check: $\bot, \neg, \wedge, X, U$.

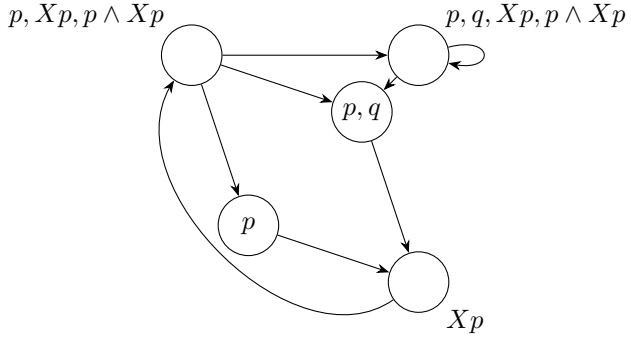


- $p \wedge Xp$

Solution. Initial graph with all possible states:



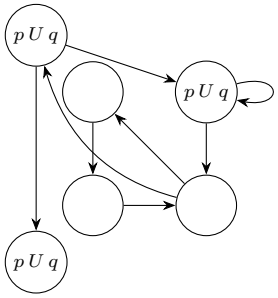
Then erase states without outbound transitions:



Only one copy of the start state remains, and it satisfies Xp, so the model check succeeds.

- $p\,U\,q$

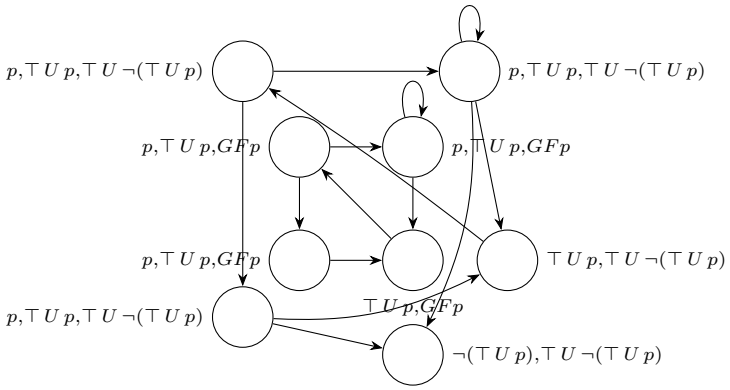
  Solution. Initial graph (omitting for space reasons the p and q labels, which are as in the original system):



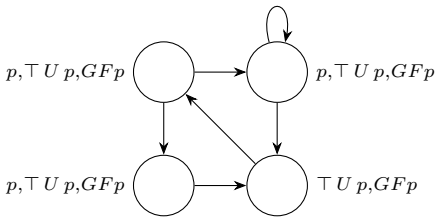
  Only the bottom state gets deleted for lacking an outbound arrow. There remain two copies of the start state, and one is not labelled $p\,U\,q$, so the model check fails.

- GFp

  Solution. $GFp \equiv \neg F \neg Fp \equiv \neg(\top\,U\,\neg(\top\,U\,p))$. Initial graph, omitting $\top$ which is labelled everywhere:



  We delete the bottom state because it has no outbound transitions. But then the entire outside perimeter becomes useless, as all states claim $\top\,U\,\neg(\top\,U\,p)$ but no state in it delivers on $\neg(\top\,U\,p)$. So we delete those four states also to end up with:



  Only one copy of the start state remains, and it satisfies GFp, so the model check succeeds.