COMP2620/6262 (Logic) Tutorial

Week 10

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.

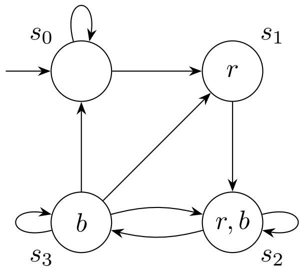
This week's quiz is on **tableaux** for LTL. Your tutor will hand out blank paper, on which you should clearly write your university ID and name. Your tutor will also hand out paper with all tableaux rules for LTL, including loop and simple repetition rules. They will then write two signed propositions on the whiteboard. You should then construct a **complete** tableaux for those signed propositions. In other words, if a branch terminates, you should cross if appropriate, indicate with [LOOP,n] or [REP,n] for some number n if applicable, then move on to complete all non-terminated branches. You will not need to extract a satisfying model. You will have **twenty minutes** to do this.

If your tableau becomes repetitive due to similar or identical propositions appearing in multiple nodes, you may omit some steps with a short English explanation of the reason. Do not skip any steps, including numbering and justifying your lines, the first time you deal with any propositions.

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.

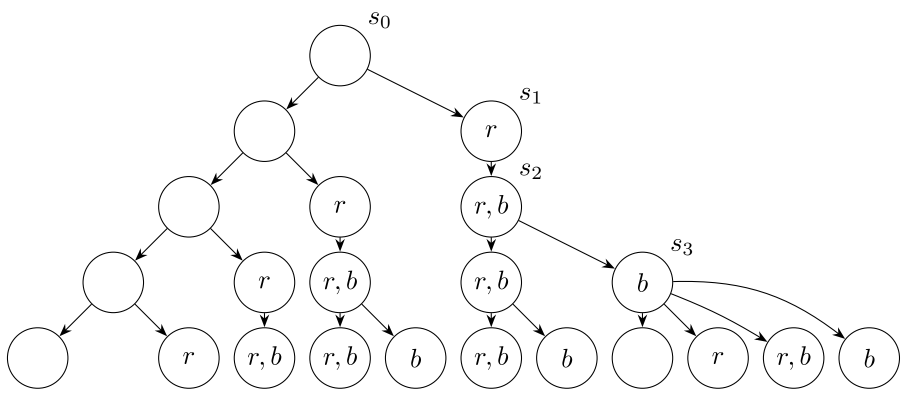
This Week's Exercises

1. Consider a transition system representing a machine that can receive one or more unstarted requests (r), and become busy fulfilling these requests (b). Here the notion of 'next state' will correspond to a real clock ticking, and therefore the machine might, for example, stay busy for an unpredictable number of steps, and might receive new requests while it is busy. It also has to become busy before it is able to label a request as started (which will negate r, unless new requests come in the meantime).



Draw the corresponding computation tree up to level 4, where the root counts as level 0.

Solution.



2. State the following CTL propositions in plain and intuitive English, and discuss informally whether or not they hold for the machine of the previous question.

   - EGb

     Solution. It might be that the machine is busy at all times. This is not correct because the machine starts at s_0, which is not busy. We can however affirm $EG\neg b$, because we might never leave state s_0.

   - $AG(r \to AXb)$

     Solution. If the machine (ever) has an unstarted request, it will be busy at the next moment. This can be confirmed by seeing that the only states labelled with r (s_1 and s_2) can only transition to states labelled with b (s_2 and s_3)

   - $EF(b \wedge \neg r \wedge EX(\neg b \wedge r))$

     Solution. It is possible that, at some point in the future, we will be busy with no unstarted requests, then at the next moment not be busy but have an unstarted request (presumably, the new request came in at the same moment that all previous work was completed). This holds because of the transition from s_3 to s_1.

   - $AGAF(r \to b)$

     Solution. At all possible future states, the machine will eventually reach a state at which r implies b. Recall that $r \to b$ is equivalent to $\neg r \vee b$, so s_1 is the only state that does not satisfy this. So we could rephrase this to say that at all possible future states, the machine will eventually reach a state that is not s_1. This is true because if it ever does reach s_1, it next transitions to s_2.

   - $E[\neg r\, U\, b]$

     Solution. It is possible that the machine eventually becomes busy, and there are no unstarted requests before then. This fails because there is no way for the machine to become busy without an unstarted request coming in strictly earlier.

   - $E[\neg r\, U\, AXAXb]$

     Solution. It is possible that the machine eventually reaches a state where it must become busy at the next but one moment, and there was no request before then. This holds because after some period in s_0, we might transition to s_1, and all states reachable within the next two moments (s_2 and s_3) affirm b.

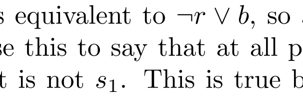
3. Are the following equivalences of CTL? If so, argue why, using the equivalences learned in class. If not, define a transition system that affirms one and negates the other.

   - $\neg(AFp \vee EG\neg q)$ and $EG\neg p \wedge AFq$

     Solution. $\neg(AFp \vee EG\neg q) \equiv \neg AFp \wedge \neg EG\neg q$ by one of the first dualities we learned - de Morgan's laws. This is equivalent to $EG\neg p \wedge AF\neg\neg q$, which finally equals $EG\neg p \wedge AFq$ by double negation elimination.

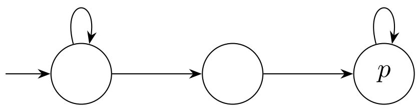
   - $AXEXp$ and $EXAXp$.

     Solution.



All next steps from the start (of which there is only one possible!) can possibly transition to the state with p, so $AXEXp$. But there exists a step from the start (again, only one possibility) which does not satisfy that all transitions from it satisfy p. Therefore $EXAXp$ fails.

The following system satisfies $EXAXp$ but not $AXEXp$. Can you see why?



- $\neg AGEGp$ and $AFEF\neg p$

  Solution.

  This question was asked in error: these propositions are in fact equivalent (both sides are equivalent to $EF\neg p$), but it is not possible to show this, as asked, using equivalences learned in class (one would instead need to argue via the semantics).

- $\neg A[p\,U\,q]$ and $AFq \to E[\neg p\,U\,\neg(p \vee q)]$

  Solution. $\neg A[p\,U\,q] \equiv \neg\neg(E[\neg p\,U\,\neg p \wedge \neg p] \vee EG\neg q)$ by the complex equivalence seen in lectures. This is equivalent to $E[\neg p\,U\,\neg p \wedge \neg q] \vee EG\neg q$ by double negation; $E[\neg p\,U\,\neg(p \vee q)] \vee \neg AFq$ by duality; $\neg AFq \vee E[\neg p\,U\,\neg(p \vee q)]$ by commutativity of disjunction; and finally by the fact that propositional logic, $\neg p \vee q \equiv p \to q$, we have the right hand side.

4. The LTL tableaux rule for $\mathbf{T} : F\varphi$ branches into two possibilities: $\mathbf{T} : \varphi$ and $\mathbf{T} : XF\varphi$. Justify this with a mathematical agument (not a tableaux proof!) using the LTL semantics, that $F\varphi$ and $\varphi \vee XF\varphi$ are equivalent.

   Could a similar splitting into a disjunction be possible for AF and/or EF in CTL? Again, use the semantics as justification.

   Solution. For any transition system $\mathcal{M}$ and path σ, we have $\vDash_{\mathcal{M},\sigma} \varphi \vee XF\varphi$ iff
   $\vDash_{\mathcal{M},\sigma} \varphi$ or $\vDash_{\mathcal{M},\sigma} XF\varphi$, iff
   $\vDash_{\mathcal{M},\sigma} \varphi$ or $\vDash_{\mathcal{M},\sigma_{\geq 1}} F\varphi$, iff
   $\vDash_{\mathcal{M},\sigma} \varphi$ or there exists $i \geq 1$ such that $\vDash_{\mathcal{M},\sigma_{\geq i}} \varphi$.
   But this is the same as saying that φ holds for the path starting at 0 or some i greater than 1, i.e. any natural number, so this holds iff $\vDash_{\mathcal{M},\sigma} F\varphi$.

   The answer is to the question about CTL is yes: $AF\varphi$ is equivalent to $\varphi \vee AXAF\varphi$, and $EF\varphi$ is equivalent to $\varphi \vee EXEF\varphi$. I will show the case for A; the case for E is similar.

   For any transition system $\mathcal{M}$ and state s, we have $\vDash_{\mathcal{M},s} \varphi \vee AXAF\varphi$ iff
   $\vDash_{\mathcal{M},s} \varphi$ or $\vDash_{\mathcal{M},s} AXAF\varphi$, iff
   for all states s' such that $s \to s'$ in $\mathcal{M}$, $\vDash_{\mathcal{M},s} \varphi$ or $\vDash_{\mathcal{M},s'} AF\varphi$, iff
   for all states s' such that $s \to s'$ in $\mathcal{M}$, and all paths σ starting at s', there is exists a state t in σ such that $\vDash_{\mathcal{M},s} \varphi$ or $\vDash_{\mathcal{M},t} AF\varphi$.
   Now $\vDash_{\mathcal{M},s} AF\varphi$ iff for all paths starting at s, there exists a state t' in that path such that $\vDash_{\mathcal{M},t'} \varphi$. If that state t is s, then $\vDash_{\mathcal{M},s} \varphi$. Otherwise it is a state in some path starting at s', for any $s \to s'$, and this matches the second case of the disjunction above. So either way the two sides of the equivalence coincide.

5. **The test at the start of the next tutorial will resemble this question, by asking you to model English language sentences into CTL*.**

   As an aspiring logician, you are often approached by people demanding that you formalise their opinions and ravings. Today's customer is a wizard who wants you to record their observations and predictions about the world of magic and the supernatural. Luckily, although you are not exactly an expert in this domain, you can see that CTL* can express their statements. Define some appropriate propositional variables, then formalise the following:

   - "It is necessary that eventually the phoenix shall burn"

     Solution. Writing p as 'the phoenix burns', $A[Fp]$.

   - "It may be that the fairies shall always fail to fly"

     Solution. Writing f for 'the fairies fly', $E[G\neg f]$.

- "It must be either that at some time the phoenix shall burn, or that tomorrow the goblins shall prosper"
  Solution. Writing g for 'the goblins prosper', $A[Fp \vee Xg]$.
- "Perhaps it is true both that goblins will prosper until the sirens sing, and that if the fairies are flying then the sirens shall sing forever"
  Solution. Writing s for 'the sirens sing', $E[(g\,U\,s) \wedge (f \to Gs)]$.

Bad news! After the wizard occupied your time, a rugby fanatic has tackled you and demanded a logical translation of their outlandish claims about prospects for the current and future seasons of four Australian rugby union teams. These you must also translate into CTL*:

- "Possibly, the Waratahs will win forever"
  Solution. Writing w as 'the Waratahs win', $E[Gw]$.
- "We will definitely eventually have a season where both the Reds and Force win"
  Solution. Writing r for 'the Reds win', and f for 'the Force win', $A[F(r \wedge f)]$. Note that if we did not specify the Reds and Force winning in the same season, $A[Fp \wedge Fq]$ is more plausible.
- "Maybe, at some time, the Reds will not win in their next season"
  Solution. $E[F\neg Xr]$, or equivalently, $E[FX\neg r]$.
- "Certainly, either the Brumbies are winning, or the Brumbies will win until it becomes necessary that the Reds win forever"
  Solution. Writing b for 'the Brumbies win', $A[b \vee (b\,U\,A[Gr])]$.

6. (Tricky and Open Ended) How might 'past operators' work in LTL, CTL, or CTL*? By past operators we mean for example: X^{-1}, a unary connective meaning 'on the immediately previous state in the path considered'; F^{-1}, meaning 'there exists a previous state in the path considered', $(AF)^{-1}$, meaning 'for all paths that lead to the current state, there exists a previous state in such a path', and so on. Would adding these operators to our logics make them more expressive, or can we express them using the logics we already have?

   Solution. To answer this question for LTL it might help to start with the question: what exactly should the semantics of $X^{-1}p$ be? Fixing a model $\mathcal{M}$ and path σ, we have $\vDash_{\mathcal{M},\sigma} p$ if p holds at position 0 of σ. Does $\vDash_{\mathcal{M},\sigma} X^{-1}p$ then ask about position -1 of the path? This does not seem to make sense. We can make it make sense in a few ways: by requiring that paths have infinite pasts as well as infinite futures; by saying such a proposition is vacuously true; by saying it is false because it refers to a time that does not exist; or we could use a similar trick as for CTL* and, just as that logic stops you from writing a temporal operator that is not under an A or E, we could stop people writing down X^{-1} except under some other operator which moves us into the future.

   All of these approaches seem defensible but in fact the 'false because it refers to a time that does not exist' approach is most common. Semantics are defined with respect to a transition system, a path, *and* a natural number for the current position of interest in the path, so e.g.

   - $\vDash_{\mathcal{M},\sigma,i} X\varphi$ if $\vDash_{\mathcal{M},\sigma,i+1} X\varphi$
   - $\vDash_{\mathcal{M},\sigma,i} X^{-1}\varphi$ if $i > 0$ and $\vDash_{\mathcal{M},\sigma,i-1} \varphi$

   It turns out that every proposition of this 'LTL with past operators' logic can be translated into an LTL proposition which is true for exactly the same set of models. In other words, adding past operators to LTL might be convenient, but it does not add the ability to express anything new to the logic (just like adding an 'exclusive or' to our propositional logic). The proof of this is outside the scope of this course, but the translation would for example map $X^{-1}p$ to the LTL formula $\bot$, and $XXX^{-1}p$ to Xp.

   The situation for CTL is different: adding past operators does allow one to express different things, by allowing one to talk about 'alternative presents'. The formula $AX(AX)^{-1}p$ is not equivalent to p because, even if we are at a state that satisfies p, after going back a step with $(AX)^{-1}$ we then go forward with AX to consider every possible next state, some of which might not satisfy p! Less intuitively, it turns out that CTL* is so powerful that adding past operators does *not* increase its expressive power. For a full discussion, including proofs, of these facts see the 1995 paper A hierarchy of temporal logics with past.