LTL Tableaux

neXt Rules, Other Rules, Infinite Branches

Ranald Clouston

April 24, 2025

Australian
National
University

LTL Tableaux

With first order (and propositional) logic we saw how the tableaux method can help us to prove validity, satisfiability, and to extract satisfying models.

First order models are sometimes infinite, which is a mismatch with finite proof techniques.

- The branching quantifier rules only sometimes allow us to avoid an infinite development and find a finite satisfying model.

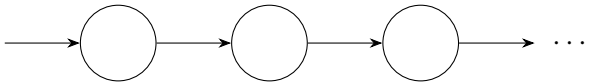
LTL seems worse, because *all* its semantics are infinite: even if the transition system is finite, the paths which are the actual subject of our semantics are infinite.

Yet we will present a tableaux method that is not only sound and complete, but a decision procedure for satisfiability!

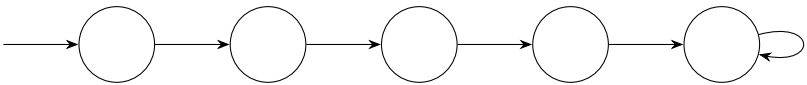
- We will not prove these metalogical properties.

Finite Models from LTL Tableaux

The transition systems we derive from our tableaux will be deterministic: each state can transition to exactly one state:

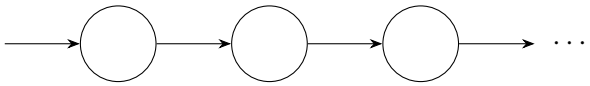


One way to keep things finite is to discover that after a certain number of steps we no longer care what our model does, so we can happily loop forever in the final state, e.g.:

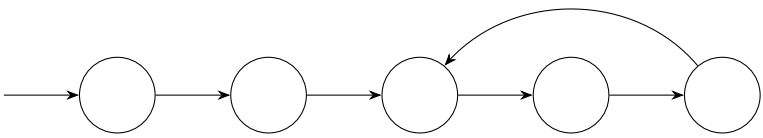


Finite Models from LTL Tableaux

The transition systems we derive from our tableaux will be deterministic: each state can transition to exactly one state:



The more general way to keep things finite is to build our system in 'lasso' shape, repeating ourselves after a certain point, e.g.:



neXt Rules

Stepping between Nodes

First order logic tableaux never discard formulas: even if we will never again apply a rule to them, we keep them in case their contradiction appears.

With temporal logic, when we move to the new state, we throw most of our facts away!

- We will use a special symbol $\fatslash$ in our tableaux to indicate these steps.
- The only propositions that trigger these steps are those with X as main connective.
- We call the parts of our tableaux between $\fatslash$ symbols (or between the start and $\fatslash$, or $\fatslash$ and the end) nodes.
- We sometimes give out nodes names like l, m, n. We write $m < n$ if m and n are on the same branch and there is least one $\fatslash$ after m and before n; $m \leq n$ if $m = n$ or $m < n$.
- Contradictions only close branches when they appear in the same node - there is no contradiction between $\mathbf{T} : \varphi$ and $\mathbf{F} : \varphi$ at different nodes.

Tableaux Rule for neXt

A node is poised if it is terminated open, except for X propositions.

This rule may *only* be used when the node is poised.

$$\frac{\mathbf{T} : X\varphi_1 \quad \cdots \quad \mathbf{T} : X\varphi_m \quad \mathbf{F} : X\psi_1 \quad \cdots \quad \mathbf{F} : X\psi_n}{\begin{array}{c} \Downarrow \\ \mathbf{T} : \varphi_1 \\ \vdots \\ \mathbf{T} : \varphi_m \\ \mathbf{F} : \psi_1 \\ \vdots \\ \mathbf{F} : \psi_n \end{array}}$$

Tableaux Rule for neXt

A node is poised if it is terminated open, except for X propositions.

This rule may *only* be used when the node is poised.

More compact notation:

$$\frac{\mathbf{T} : X\varphi_1 \quad \cdots \quad \mathbf{T} : X\varphi_m \quad \mathbf{F} : X\psi_1 \quad \cdots \quad \mathbf{F} : X\psi_n}{\Downarrow}$$
$$\mathbf{T} : \varphi_1, \cdots, \mathbf{T} : \varphi_m, \mathbf{F} : \psi_1, \cdots, \mathbf{F} : \psi_n$$

So all X propositions fire at the same time, and all other propositions are forgotten!

Whiteboard Examples

$$\frac{\mathbf{T} : X\varphi_1 \quad \cdots \quad \mathbf{T} : X\varphi_m \quad \mathbf{F} : X\psi_1 \quad \cdots \quad \mathbf{F} : X\psi_n}{\Downarrow}$$
$$\mathbf{T} : \varphi_1, \cdots, \mathbf{T} : \varphi_m, \mathbf{F} : \psi_1, \cdots, \mathbf{F} : \psi_n$$

- $\mathbf{T} : p, \mathbf{T} : X\neg p$
- $\neg Xp \vdash X\neg p$
- $X(p \to q), \neg Xq \vdash \neg Xp$

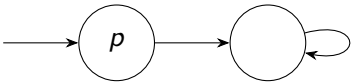
Extracting a Transition System

If we add only the X rule to propositional logic, our tableaux still terminate.

- All our propositions get smaller as we move down through our tableaux.

If we terminate open, define a transition system with the same number of states as there are nodes in the branch, transitioning from one to the next, then looping on the final state. The labelling function is given by the signed propositional variables.

e.g. $\mathbf{T} : p$, $\mathbf{T} : X\neg p$:



Rules for G, F, and U

Tableaux Rule for True Globally

$$\frac{\mathbf{T} : G\varphi}{\begin{array}{c}\mathbf{T} : \varphi \\ \mathbf{T} : XG\varphi\end{array}}$$

If φ is true globally, it is true at the present, and globally true from the next moment.

This rule does not imperil termination within a node: $XG\varphi$ is stuck until we step, and φ is smaller than $G\varphi$. But it is obviously concerning for termination of the tableau!

- Whiteboard example: $Gp \vdash XXp$

Tableaux Rules for False Globally

$$\frac{\mathbf{F} : G\varphi}{\mathbf{F} : \varphi \qquad \mathbf{F} : XG\varphi}$$

If φ is not always true, either we are already at the node where it fails, or it fails at some time on or after the next state.

Tableaux Rules for some Future

Because of the duality between G and F, the **T** rule for F branches and the **F** rule does not.

$$\frac{\mathbf{T} : F\varphi}{\mathbf{T} : \varphi \qquad \mathbf{T} : XF\varphi} \qquad \frac{\mathbf{F} : F\varphi}{\begin{array}{c}\mathbf{F} : \varphi \\ \mathbf{F} : XF\varphi\end{array}}$$

- Whiteboard example: show the satisfiability of $\mathbf{T} : Fp$, $\mathbf{F} : Gp$

Tableaux Rules for Until

$$\frac{\mathbf{T} : \varphi \, U \, \psi}{\begin{array}{c|c} \mathbf{T} : \psi & \begin{array}{c} \mathbf{T} : \varphi \\ \mathbf{T} : X(\varphi \, U \, \psi) \end{array} \end{array}} \qquad \frac{\mathbf{F} : \varphi \, U \, \psi}{\begin{array}{c} \mathbf{F} : \psi \\ \hline \mathbf{F} : \varphi \quad \mathbf{F} : X(\varphi \, U \, \psi) \end{array}}$$

- Whiteboard example: $\mathbf{F} : p$, $\mathbf{F} : Xp$, $\mathbf{T} : q \, U \, p$

Termination of Infinite Branches

Infinite Branches

We have many rules which append X to a proposition. Then the X-rule duplicates that proposition into the next node.

This creates the obvious risk of infinite branches, repeating propositions forever.

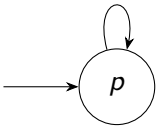
- in fact (not proved) all LTL tableaux with infinite branches will eventually start repeating themselves;
- Not necessarily a 'self loop' repetition: $\mathbf{T} : G(p \to X\neg p), \mathbf{T} : G(\neg p \to Xp)$ repeats itself in cycles of length 2.

If our tableaux procedure is to be complete and terminating, we need a way to recognise that we have entered an infinite branch, and to decide whether that branch is satisfiable or not.

Infinite Branches, Good and Bad

Some infinite branches are satisfiable

- e.g. $\mathbf{T} : Gp$ can clearly be satisfied by the transition system



Others appear not to be

- e.g. $\mathbf{T} : Gp, \mathbf{T} : F\neg p$

Eventualities

The signed propositions $\mathbf{T} : F\varphi$, $\mathbf{T} : \varphi \, U \, \psi$, and $\mathbf{F} : G\varphi$ are called eventualities.

- Each guarantee that something will come true, or false, eventually.
- terminology: the X-eventualities are $\mathbf{T} : XF\varphi$, $\mathbf{T} : X(\varphi \, U \, \psi)$, and $\mathbf{F} : XG\varphi$

The unsatisfiable infinite branches (which we cross) are *those with eventualities that are postponed forever*.

For example, the tableau created by $\mathbf{T} : Gp, \mathbf{T} : F\neg p$ never fulfils the eventuality $\mathbf{T} : F\neg p$.

- Easy to observe in this case, but how do we identify these situations in general?

The Loop and Repetition Rules

We will introduce three rules: Loop, Simple Repetition, and Repetition.

- à la rules 'stop and cross a branch with a contradiction in it' and 'stop and leave open a branch with no rules applicable to it'.

These rules each do two jobs: identify that we reached an unproductively repetitive situation, and decide whether to cross, or leave open.

- The Loop rule leaves branches open, and Repetition rules cross them.
- They are a bit technical to state, so hard to understand at first.
- Not obvious, nor proven in this course: every otherwise infinite branch will eventually satisfy one of these rules.

The Loop Rule

If we call our current node n, and it is poised, and

- there is a node $l < n$ such that all base case and X-formulas of n were already in l;
- and for every X-eventuality - respectively $\mathbf{T} : XF\varphi$, $\mathbf{T} : X(\varphi \, U \, \psi)$, or $\mathbf{F} : XG\varphi$ - in l we have, respectively, $\mathbf{T} : \varphi$, $\mathbf{T} : \psi$, or $\mathbf{F} : \varphi$, in some node m such that $l < m \leq n$;

then the current branch terminates open (satisfiable).

Notation: write LOOP under your branch, with a number indicating how many steps back you have to go to the earlier node l, e.g. '[LOOP,2]' if it is the current node's grandparent.

Whiteboard examples:

- $\mathbf{T} : Gp$
- Observe informally that $\mathbf{T} : Gp, \mathbf{T} : F\neg p$ will never fulfil the Loop rule.
- Find a model that shows that $GFp \vdash FGp$ is not valid.

Extracting a Satisfying Model from The Loop Rule

Our satisfying model has a state for each node in our branch, *except* the last one.

This is because the last one was a repetition of a previous one, so we put in our 'lasso' transition from the state corresponding to the second-to-the-last node, to whichever node our last one repeated.

e.g. $\mathbf{T} : Gp$ and $\mathbf{T} : GFp$, $\mathbf{F} : FGp$:

The Simple Repetition Rule

How do we recognise that we have reached an infinite branch that cannot be turned into a satisfying model?

The general rule for this is quite complicated, but there is a simpler version that is good enough for almost all jobs.

Intutition:

- The loop rule requires that we deal with all X-eventualities between our repeating nodes, so vacuously includes the case that there were no X-eventualities.
- The simple repetitition rule requires that there are one or more X-eventualities, and we fail to deal with any of them. Because we are in a repetitive situation, we will *never* deal with any of them!

The Simple Repetition Rule

If we call our current node n, and it is poised, and

- there is a node $l < n$ with the same base case and X-formulas as n, and this includes at least one X-eventuality;
- and there is no X-eventuality - respectively $\mathbf{T} : XF\varphi$, $\mathbf{T} : X(\varphi\, U\, \psi)$, or $\mathbf{F} : XG\varphi$ - such that, respectively, $\mathbf{T} : \varphi$, $\mathbf{T} : \psi$, or $\mathbf{F} : \varphi$, is in a node m such that $l < m \leq n$;

then the current branch should be closed (unsatisfiable) with a cross.

Notation: write REP under your branch, with a number indicating how many steps back you have to go to the earlier node l, e.g. '[REP,2]' if it is the current node's grandparent.

Whiteboard examples:

- $\mathbf{T} : Gp$, $\mathbf{T} : F\neg p$
- $Gp \vdash GFp$

Towards a General Repetition Rule

We covered the situation were all X-eventualities are paid off between repetitive nodes (loop), and where none are (simple repetition). How about when some, but not all, are?

A satisfiable example of such partial progress:

- $\mathbf{T} : GXFp$, $\mathbf{T} : GXF\neg p$, $\mathbf{T} : Fp$, $\mathbf{T} : F\neg p$, if we repeatedly take the branch with $\mathbf{T} : p$.

An unsatisfiable example:

- $\mathbf{T} : GXFp$, $\mathbf{T} : GXF\bot$, $\mathbf{T} : Fp$, $\mathbf{T} : F\bot$.

The (slightly unwieldy) solution: wait until we have *three* repeating nodes, and cross if we are not making any more progress between the second and third, than we did between the first and second.

The Repetition Rule

If we call our current node n, and it is poised, and

- the Loop rule is not applicable, and;
- there are nodes $l < m < n$ with the exact same base case and X-formulas as n, and;
- The following holds for every X-eventuality in the node:
  - for $\mathbf{T} : XF\varphi$, if $\mathbf{T} : \varphi$ at a node j such that $m < j \leq n$, then $\mathbf{T} : \varphi$ at a node k such that $l < k \leq m$;
  - for $\mathbf{T} : X(\varphi\, U \psi)$, if $\mathbf{T} : \psi$ at a node j such that $m < j \leq n$, then $\mathbf{T} : \psi$ at a node k such that $l < k \leq m$;
  - for $\mathbf{F} : XG\varphi$, if $\mathbf{F} : \varphi$ at a node j such that $m < j \leq n$, then $\mathbf{T} : \varphi$ at a node k such that $l < k \leq m$;

then the current branch terminates closed (unsatisfiable), and can be crossed.

Notation: write REP under your branch, with two numbers indicating how many steps back $l$and m are, e.g. '[REP,2,4]'.

A Pragmatic Note on the Repetition Rules

The simple repetition rule is not necessary; any case that is crossed by it would be crossed by the general rule.

- But the simpler rule is simpler, and faster - cross at the first repetition.

Examples that require the general rule to cross mostly only 'naturally' arise with examples that are too big to do by hand anyway.

For this reason, in this course we will only require practice of the simple repetition rule.

- We might still ask conceptual questions about the general rule.