ANU
Logic Summer School
Lectures for December 2024
Verification Via Temporal Logic

Guest lecturer: Mark Reynolds, The University of Western Australia

December 2024

Description

These 5x 1 hour lectures introduce temporal logic as a means of formal verification of systems.

Outline of 5 Lectures, 6 Topics

- The Logics: LTL, CTL, CTL*
- A tableau for LTL-Sat
- Soundness
- Completeness
- CTL Satisfiability
- (if we have time) CTL* model checking

Lecture 2, LTL satisfiability

We start a detailed look at an algorithm to decide the satisfiability of LTL formulas.

We want to invent an algorithm which solves the LTL-SAT problem. Input should be a formula from LTL. Output is "yes" or "no" depending on whether the formula is satisfiable or not.

(REMINDER) LTL Syntax:

If $p \in \mathcal{L}$ then p is a wff.

If α and β are wff then so are $\neg\alpha$, $\alpha \wedge \beta$, $X\alpha$, and $\alpha U \beta$.

LTL Semantics:

Write $M, \sigma \models \alpha$ iff the formula α is true of the fullpath σ in the structure $M = (S, R, g)$ defined recursively by:

$M, \sigma \models p$ iff $p \in g(\sigma_0)$, for $p \in \mathcal{L}$

$M, \sigma \models \neg\alpha$ iff $M, \sigma \not\models \alpha$

$M, \sigma \models \alpha \wedge \beta$ iff $M, \sigma \models \alpha$ and $M, \sigma \models \beta$

$M, \sigma \models X\alpha$ iff $M, \sigma_{\geq 1} \models \alpha$

$M, \sigma \models \alpha U \beta$ iff there is some $i \geq 0$ such that $M, \sigma_{\geq i} \models \beta$ and for each j, if $0 \leq j < i$ then $M, \sigma_{\geq j} \models \alpha$

Satisfiability:

A formula α is satisfiable iff there is some structure (S, R, g) with some fullpath σ through it such that $(S, R, g), \sigma \models \alpha$.

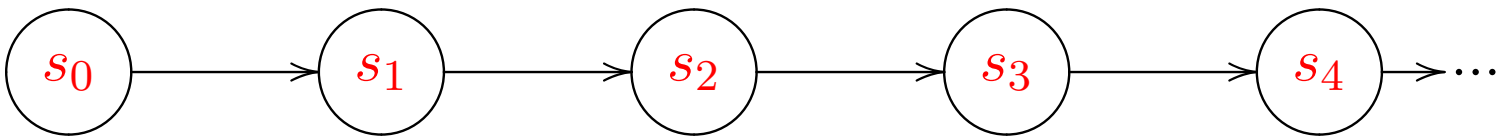
Eg, $\top$, p, Fp, $p \wedge Xp \wedge F\neg p$, Gp are each satisfiable.

Eg, $\bot$, $p \wedge \neg p$, $Fp \wedge G\neg p$, $p \wedge G(p \rightarrow Xp) \wedge F\neg p$ are each not satisfiable.

Can we invent an algorithm for deciding whether an input formula (of LTL) is satisfiable or not?

Build a model:

To test satisfiability of a formula, what about trying to build a model of it?
Eg, suppose that we ask about the satisfiability of
$\neg p \wedge X\neg p \wedge (qUp)$

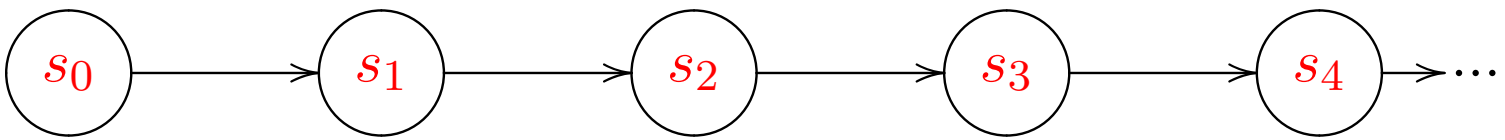


Let's make $\neg p \wedge X\neg p \wedge (qUp)$ true at s_0.

By propositional reasoning we need to make $\neg p$, $X\neg p$ and qUp true at s_0 as well.

Build a model of $\neg p \wedge X\neg p \wedge (qUp)$:

So $\{\neg p \wedge X\neg p \wedge (qUp), \neg p, X\neg p, qUp\}$ are to hold at s_0.



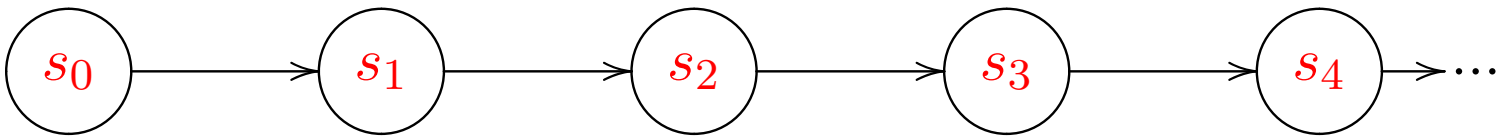
Easy to make $\neg p$ hold there.
Easy to see that we should also make $\neg p$ true at s_1.
But what about qUp?

qUp:

There are two alternative ways to make $\alpha U \beta$ true at a state s.
You can make β true there.
OR
You can make α true there and make $\alpha U \beta$ true at a next state.

In our case, with qUp we can not do the former in s_0 so we need to make q true at s_0 and postpone qUp until s_1.
And again at s_1 we have to make q true there and postpone qUp until s_2.
At s_2 we can use the first case.
Thus we show the formula is satisfiable and we have built a model.

From tableau labels to labelling:



$\{\neg p \wedge X\neg p \wedge (qUp), \neg p, X\neg p, qUp, q\}$ are to hold at s_0.
$\{\neg p, qUp, q\}$ are to hold at s_1.
$\{qUp, p\}$ are to hold at s_2.

Answer: $\{q\}, \{q\}, \{p\}, \{\}, \{\}, ...$

Can this be generalised?:

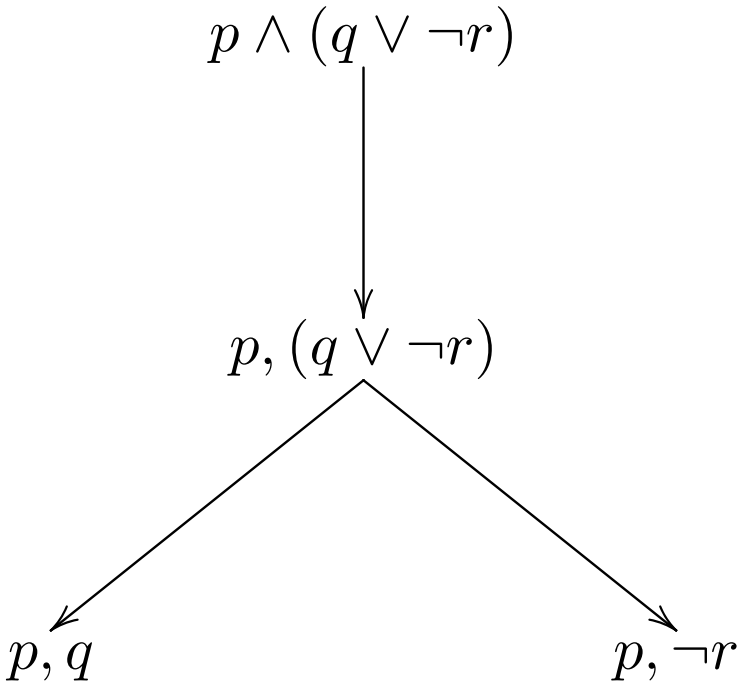
Labelling nodes with formulas is good. Starting from s_0 and working forwards in time is good.

However, some problems:
How to deal with choices (that are not immediately obvious).
What if we need to go on forever building the model?
What if we go on forever making something that is not going to be a model?

The following is my newish tree-shaped tableau for LTL. It builds on work by Sistla and Clarke (1985) and is influenced by LTL tableaux by Wolper (1982) and Schwendimann (1998). There's an Arciv paper from 2016 and the idea is also described with an implementation in an IJCAI 2016 paper (Brunello etal 2016).

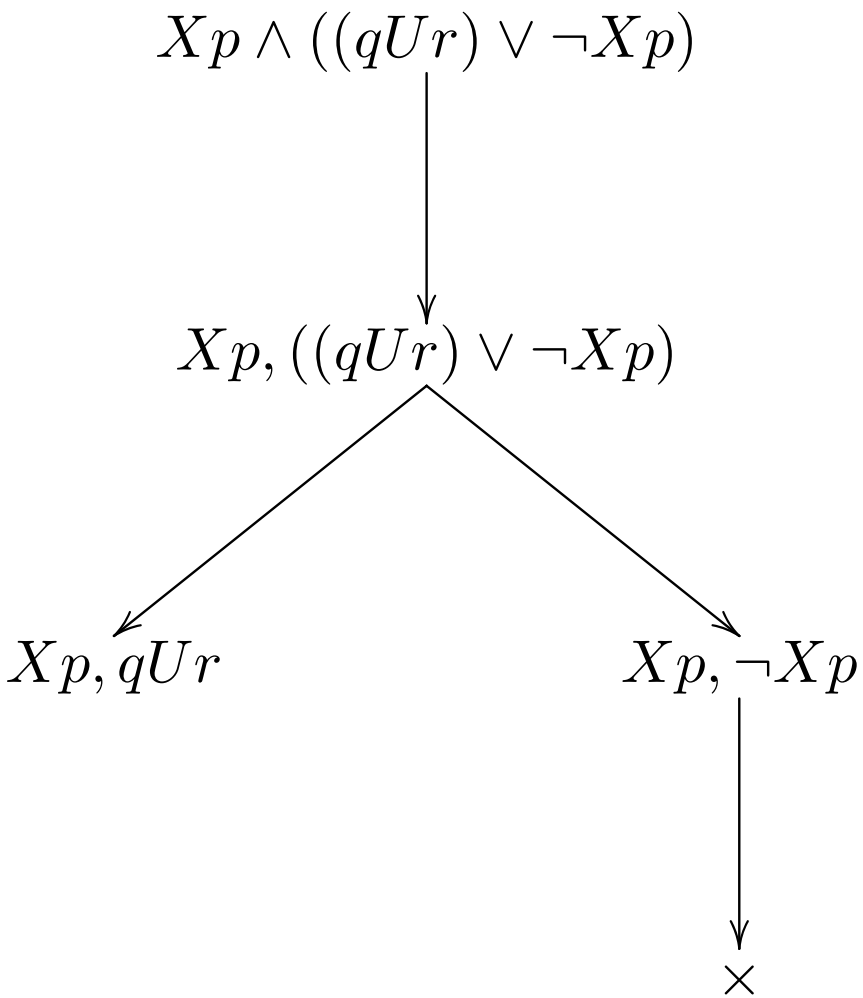
Reminder of Tableau for classical propositional logic:

Possibilities branch into a tree as we work down the page ...



Same rules for LTL:

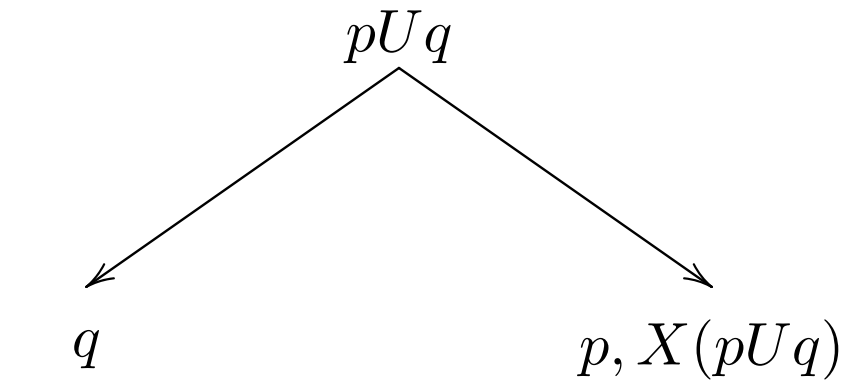
Same things can happen for LTL within a state ...

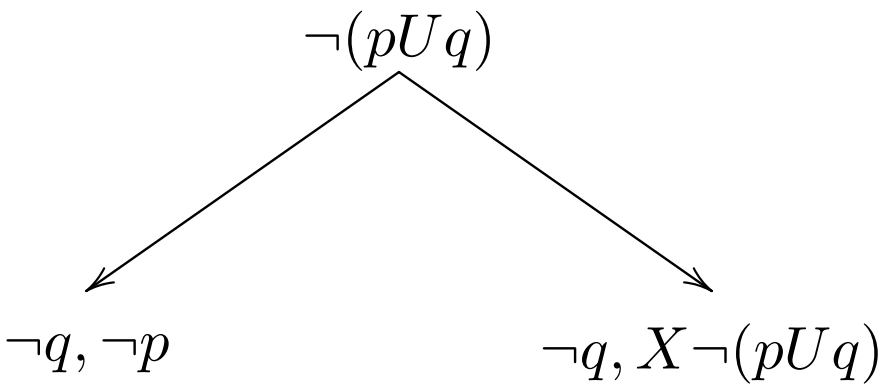


Rules:

[EMP]: If a node is labelled $\{\}$ then this node can be *ticked*.
[X]: If a node is labelled Γ with some α and $\neg\alpha$ in Γ then this node can be *crossed*.
[DNEG]: If a node is labelled $\Gamma \cup \{\neg\neg\alpha\}$ then this node can have one child labelled $\Gamma \cup \{\alpha\}$).
[CON]: If a node is labelled $\Gamma \cup \{\alpha \wedge \beta\}$ in Γ then this node can have one child labelled $\Gamma \cup \{\alpha, \beta\}$.
[DIS]: If a node is labelled $\Gamma \cup \{\neg(\alpha \wedge \beta)\}$ in Γ then this node can have two children labelled $\Gamma \cup \{\neg\alpha\}$ and $\Gamma \cup \{\neg\beta\}$ respectively.
(plus similar for other abbreviations and their negations)

Until also gives us choices:





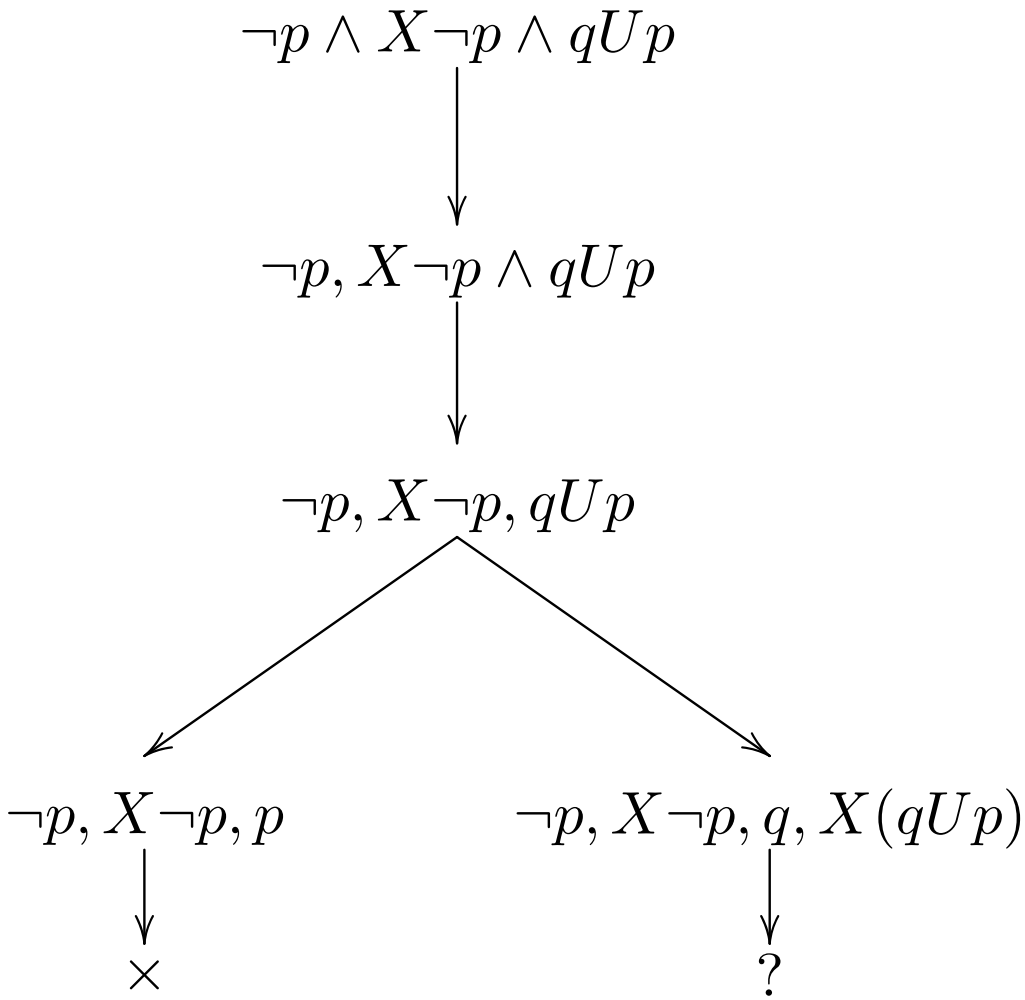
Rules for Until:

[UNT]: If a node is labelled $\Gamma \uplus \{\alpha U \beta\}$ in Γ then this node can have two children labelled $\Gamma \cup \{\alpha, X(\alpha U \beta)\}$ and $\Gamma \cup \{\beta\}$.

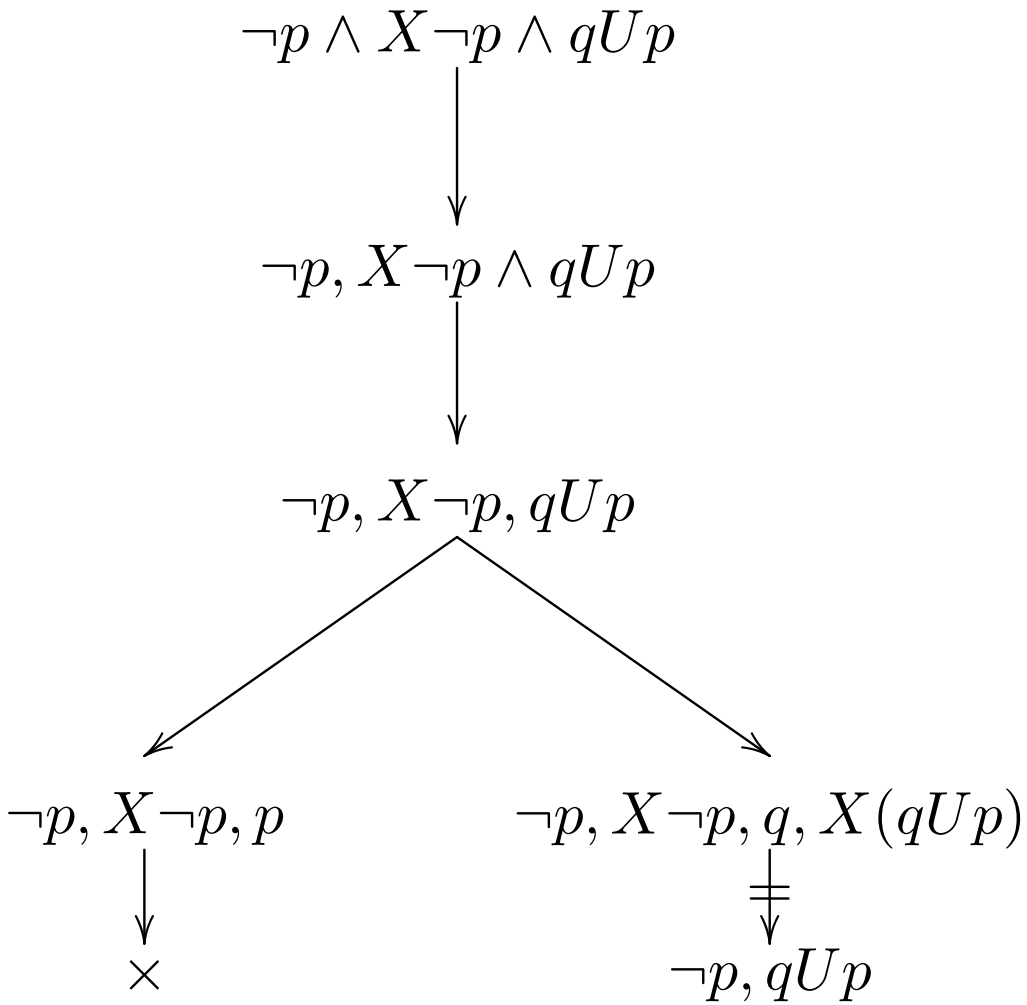
[NUN]: If a node is labelled $\Gamma \uplus \{\neg(\alpha U \beta)\}$ in Γ then this node can have two children labelled $\Gamma \cup \{\neg\beta, X\neg(\alpha U \beta)\}$ and $\Gamma \cup \{\neg\alpha, \neg\beta\}$.

(plus similar for F and G.)

But what to do when we want to move forwards in time?:



Introduce a new type of step:



Step Children:

If none of the above (static) rules are applicable to a node then we say that the node is *propositionally complete* and then, and only then, is the following rule applicable.

[TRANSITION]: The node, labelled by propositionally complete Γ say, can have one child, called a *step-child*, whose label Δ is defined as follows.
$\Delta = \{\alpha | X\alpha \in \Gamma\} \cup \{\neg\alpha | \neg X\alpha \in \Gamma\}$.

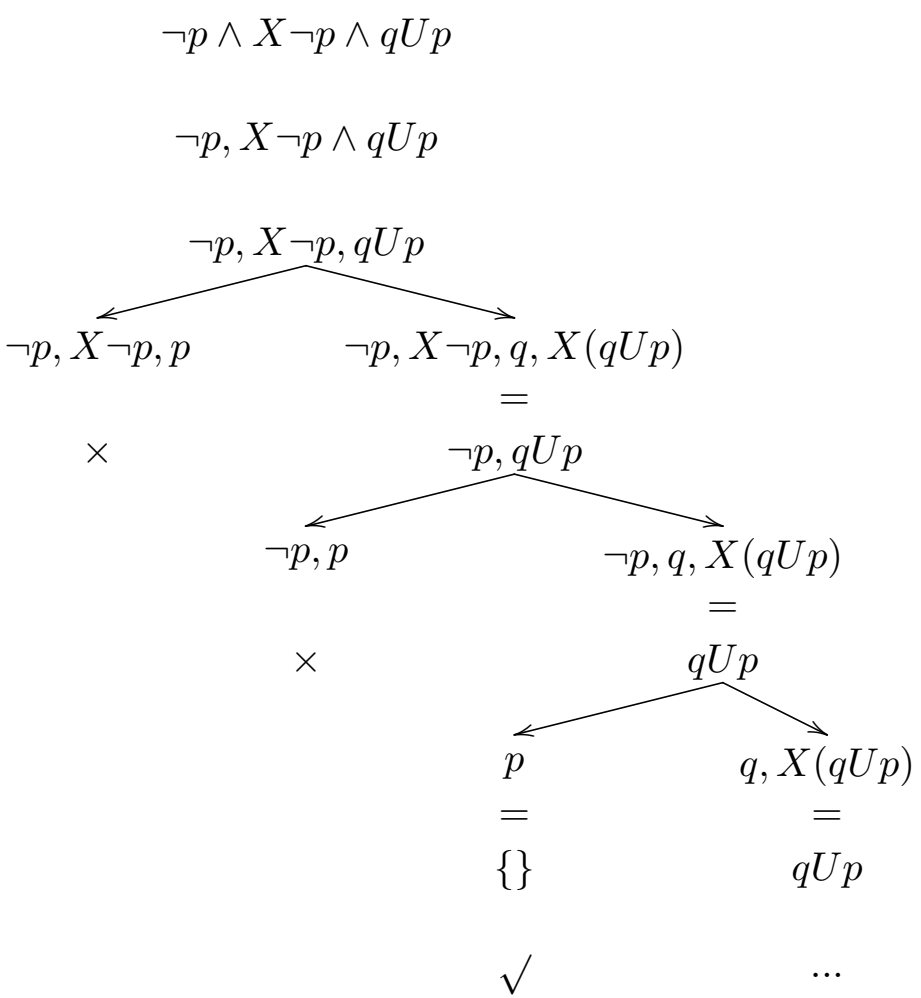
Note that Δ may be empty. After a step rule the try to use the static rules again.

Example:

Can now do the whole $\neg p \wedge X \neg p \wedge (q U p)$ example.

Homework.

Homework: $\neg p \wedge X\neg p \wedge (qUp)$ example.



Infinite behaviour:

Still some work to do.

We will try these examples in the next few slides ...

Gp

$G(p \wedge q) \wedge F\neg p$

$p \wedge G(p \leftrightarrow X\neg p) \wedge G(q \rightarrow \neg p) \wedge GF\neg q \wedge GF\neg p$

$p \wedge G(p \rightarrow Xp) \wedge F\neg p$

Example: Gp

Gp gives rise to a very repetitive infinite tableau.

$$Gp$$

$$p, XGp$$
$$=$$
$$Gp$$

$$p, XGp$$
$$=$$
$$Gp$$

$$p, XGp$$

$$\ldots$$

Notice that the infinite fullpath that it suggests is a model for Gp as would a fullpath just consisting of the one state with a self-loop (a transition from itself to itself).

Example: $G(p \wedge q) \wedge F\neg p$

But $G(p \wedge q) \wedge F\neg p$ shows that we can not just accept infinite loops as demonstrating satisfiability....

$$G(p \wedge q) \wedge F\neg p$$

$$G(p \wedge q), F\neg p$$

$$p \wedge q, XG(p \wedge q), F\neg p$$

$$p, q, XG(p \wedge q), F\neg p$$

$$p, q, XG(p \wedge q), \neg p \qquad p, q, XG(p \wedge q), XF\neg p$$

$$\times \qquad =$$

$$G(p \wedge q), F\neg p$$

$$\dots$$

$$XG(p \wedge q), XF\neg p$$

$$=$$

$$G(p \wedge q), F\neg p$$

$$\dots$$

$$XG(p \wedge q), XF\neg p$$

$G(p \wedge q) \wedge F\neg p$ continued

Notice that the infinite fullpath that the tableau suggests is this time not a model for $G(p \wedge q) \wedge F\neg p$.

Constant repeating of p, q being made true does not satisfy the conjunct $F\neg p$.

We have postponed the *eventuality* forever.

This is not acceptable.

Eventualities:

An *eventuality* is just a formula of the form $\alpha U \beta$.

(This includes $F\gamma \equiv \top U \gamma$).

If $\alpha U \beta$ appears in the tableau label of a node u then we want β to appear in the label of some later (or equal node) v. In that case we say that the eventuality is *satisfied* by v.

Eventualities are eventually satisfied in any (actual) model of a formula: by the semantics of until.

Loops:

If a label is repeated along a branch and all eventualities are satisfied in between then we can build a model by looping states. In fact, the ancestor can have a superset and it will work.

[LOOP]: If a node n has a proper ancestor (i.e. not itself) m such that $\Gamma(m) \supseteq \Gamma(n)$, m has a STEP-child, and all eventualities in $\Gamma(m)$ are satisfied by labels between m and n (including m itself) then n can be ticked.

Nice example to try:

$$p \wedge G(p \leftrightarrow X\neg p) \wedge G(q \rightarrow \neg p) \wedge G(r \rightarrow \neg p) \wedge G(q \rightarrow \neg r) \wedge GFq \wedge GFr$$

Are we done yet?

No.

Examples like $G(p \wedge q) \wedge F\neg p$ may have branches that go on forever without a tick. We need to stop and fail branches so that we can answer "no" correctly and terminate and so that we do not get distracted when another branch may be successful. In fact, no infinite branches should be allowed.

Try also: $p \wedge G(p \rightarrow Xp) \wedge F\neg p$

Can't we see that these infinite branches are just getting repetitive without making a model?

Closure set:

As we construct the tableau model we only need to record, in the labels, which formulas we want to be true at that time from a finite set of interesting formulas.
The *closure set* for a formula ϕ is as follows:

$$\{\psi, \neg\psi | \psi \leq \phi\} \cup \{X(\alpha U \beta), \neg X(\alpha U \beta) | \alpha U \beta \leq \phi\}$$

(Where $\psi \leq \phi$ means that ψ is a subformula of ϕ.)

Size of closure set is $\leq 4n$ where n is the length of the initial formula.

This shows that only formulas from a finite set will appear in labels.

...and only $\leq 2^{4n}$ possible labels.

Don't go on further than you need to:

This gives us the idea of *useless* intervals on branches in the tableau.

If a node at the end of a branch (of a partially complete tableau) has a label which has appeared already twice above, and between the second and third appearance there are no new eventualities satisfied then that whole interval of states has been useless!

The **REP**etition rule:

[REP]:
Suppose that $u = u_0, u_1, ..., u_{j-1}, u_j = v, u_{j+1}, ..., u_k = w$ is a sequence of consecutive descendants in order.
Suppose that $\Gamma(u) = \Gamma(v) = \Gamma(w)$, u and v have STEP-children and w is propositionally complete.
Suppose also that for all eventualities $\alpha U \beta \in \Gamma(u)$, if β is satisfied between v and w then β is satisfied between u and v anyway.
Then w can be crossed.

(We assume that there are some unsatisfied eventualities from $\Gamma(u)$ left. Otherwise use the LOOP rule to tick the branch.)

Examples:

Now try $G(p \wedge q) \wedge F\neg p$ and $p \wedge G(p \rightarrow Xp) \wedge F\neg p$.

LTL Tableau Summary:

Is a finite tree of nodes labelled by subsets of the closure set of ϕ such that:

- the root is labelled with $\{\phi\}$
- the labels of children of each node are according to one of the tableau rules

Successful if some leaf is ticked.

Failed if all leaves are crossed.

(Not really a proper description of an algorithm but we will see that the further details of which formula to consider at each step in building the tableau are unimportant).

Proof of Correctness:

This will consist of three parts.

Proof of soundness. If a formula has a successful tableau then it has a model.

Proof of completeness: If a formula has a model then building a tableau will be successful.

Proof of termination. Show that the tableau building algorithm will always terminate.

First, the Proof of Termination:

(Sketch only)

Any reasonable tableau search algorithm will always terminate because there can be no infinitely long branches.

We know this because the REP rule will cross any that go on too long.

Thus there will either be at least one tick or all crosses.

Termination is also why we require that other rules are not used repeatedly in between STEP rules.


Rajeev Alur and Thomas A. Henzinger.
Real-time logics: Complexity and expressiveness.
Inf. Comput., 104(1):35–77, 1993.


Burgess, J. P. Axioms for Tense Logic I: "Since" and "Until",
Notre Dame J. Formal Logic, 23(2):367–374, 1982.


J. P. Burgess and Y. Gurevich.
The decision problem for linear temporal logic.
Notre Dame J. Formal Logic, 26(2):115–128, 1985.


E. Emerson and E. C. Clarke.
Using branching time temporal logic to synthesise synchronisation skeletons.
Sci. of Computer Programming, 2, 1982.


E. Emerson and A. Sistla.
Deciding branching time logic.

In *Proc. 16th ACM Symposium on Theory of Computing*, 1984.


M. Fischer and R. Ladner.
Propositional dynamic logic of regular programs.
J. Computer and System Sciences, 18:194–211, 1979.


Oliver Friedmann, Markus Latte, and Martin Lange.
A decision procedure for CTL* based on tableaux and automata.
In *IJCAR'10*, pages 331–345, 2010.


R. Goré.
Tableau methods for modal and temporal logics.
In M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors, *Handbook of Tableau Methods*, pages 297–396. Kluwer Academic Publishers, 1999.


H. Kamp.
Tense logic and the theory of linear order.

PhD thesis, University of California, Los Angeles, 1968.


Y. Kesten, Z. Manna, and A. Pnueli.
Temporal verification of simulation and refinement.
In *A decade of concurrency: reflections and perspectives: REX school/symposium, Noordwijkerhout, the Netherlands, June 1–4, 1993*, pages 273–346. Springer–Verlag, 1994.


A. Pnueli.
The temporal logic of programs.
In *Proceedings of the Eighteenth Symposium on Foundations of Computer Science*, pages 46–57, 1977.
Providence, RI.


V. R. Pratt.
Models of program logics.
In *Proc. 20th IEEE. Symposium on Foundations of Computer Science, San Juan*, pages 115–122, 1979.


Mark Reynolds.

A tableau-based decision procedure for CTL*.
Journal of Formal Aspects of Computing, pages 1–41, August 2011.


A. Sistla and E. Clarke.
Complexity of propositional linear temporal logics.
J. ACM, 32:733–749, 1985.


Valentin Goranko, Angelo Kyrilov, and Dmitry Shkatov.
Tableau tool for testing satisfiability in ltl: Implementation and experimental analysis.
Electronic Notes in Theoretical Computer Science, 262(0):113 – 125, 2010.
Proceedings of the 6th Workshop on Methods for Modalities (M4M-6 2009).


A. Sistla and E. Clarke.
Complexity of propositional linear temporal logics.
J. ACM, 32:733–749, 1985.

S. Schwendimann.

A new one-pass tableau calculus for PLTL.

In Harrie C. M. de Swart, editor, *Proceedings of International Conference, TABLEAUX 1998, Oisterwijk*, LNAI 1397, pages 277–291. Springer, 1998.

P. Schmitt and J. Goubault-Larrecq.

A tableau system for linear-time temporal logic.

In *TACAS 1997*, pages 130–144, 1997.

M. Vardi and L. Stockmeyer.

Improved upper and lower bounds for modal logics of programs.

In *17th ACM Symp. on Theory of Computing, Proceedings*, pages 240–251. ACM, 1985.

P. Wolper.

The tableau method for temporal logic: an overview.

Logique et Analyse, 28:110–111, June–Sept 1985.