TABLEAU METHODS FOR LINEAR-TIME TEMPORAL LOGICS

Luca Geatti
Free University of Bozen-Bolzano, Italy

Nicola Gigante
Free University of Bozen-Bolzano, Italy

Angelo Montanari
University of Udine, Italy

IJCAI-ECAI 2022 Tutorial
July 23, 2022

Who are we?

Luca Geatti

Post-doc

Formal verification,
automated synthesis,
temporal logics.

Nicola Gigante

Researcher

Formal verification,
temporal planning,
temporal logics.

Angelo Montanari

Full Professor

Formal verification,
(interval) temporal logics,
data science.

Linear-time temporal logics

Temporal logics are **everywhere** in formal verification and artificial intelligence.

- specification
- verification
- temporal reasoning (*e.g.*, planning)

Tableau methods

Tableau methods are an important class of **reasoning techniques** for logics.

- born as theoretical tools for the proof theory of classical logics
- adapted to virtually any existing logic
- useful **theoretical** and **practical** tools for temporal logics

What this tutorial is about

We will give an introduction to **tableau methods** for **linear-time temporal logics**.

- **classical** techniques
- recent **advancements**
- **theory** and **practice**

Timeline of the tutorial

1. Introduction – Angelo
2. Classical tableaux – Angelo

☕ Break

3. Tree-shaped tableaux – Nicola
4. Tableaux for timed logics – Luca
5. SAT encodings and efficient implementation – Luca and Nicola
6. Conclusions – Nicola

INTRODUCTION

Temporal logic

Temporal logic, in its many incarnations, is the de-facto standard language for specifying properties of systems in **formal verification** and **artificial intelligence**.

- born in the '50s as a tool for philosophical argumentation about time [Pri57]
- the idea of its use in formal verification can be traced back to the '70s [Pnu77]
- since then, it found several applications in **artificial intelligence** as well

Temporal logic in AI

In **artificial intelligence**, when do we need to use **logic** to talk about **time**?

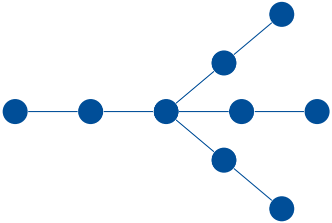
- automated **planning**
    - temporally extended goals [BK98]
    - temporal planning [FL03; May+07]
    - timeline-based planning [Del+17]
- automated **synthesis** [Jac+17]
    - system specification
- autonomy under **uncertainty** [BD19; BDP18]
    - specification of goals for planning over MDPs and POMDPs
- **reinforcement** learning [De +20; Ham+21]
    - specification of reward functions and safety conditions
- **knowledge** representation
    - temporal description logics [Art+14]
- **multi-agent** systems
    - temporal epistemic logics [Ben+09]
- temporal knowledge **mining** [BSS19]

Representing time

There are many choices to be made for the representation of **time**.

Linear

Branching



Representing time

There are many choices to be made for the representation of **time**.

Infinite

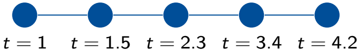
Finite

Representing time

There are many choices to be made for the representation of **time**.

Qualitative

Real-time



Representing time

There are many choices to be made for the representation of **time**.

Discrete

Dense

Representing time

There are many choices to be made for the representation of **time**.

We focus here on:

- **linear**-time logics
- **discrete**-time logics
- **infinite**-time logics
- **qualitative** and, in the second part, **real-time** logics

Linear Temporal Logic

Linear Temporal Logic (LTL) is the most used temporal logic

- introduced by Pnueli in the '70s [Pnu77]
- most of the focus of this tutorial.

Linear Temporal Logic

LTL is a **modal** logic.

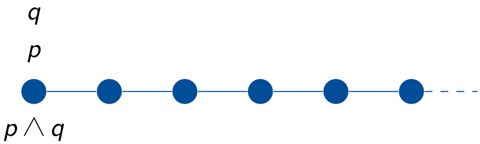
- interpreted over discrete, infinite **state sequences**
- it extends classical **propositional** logic
- temporal **operators** are used to talk about how propositions change over time

Linear Temporal Logic

Temporal operators

Linear Temporal Logic

Temporal operators



q

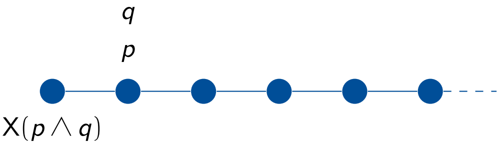
p

$p \wedge q$

p and q hold

Linear Temporal Logic

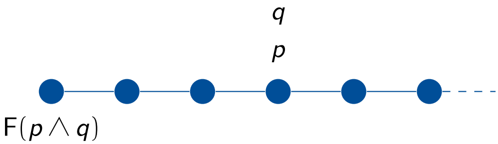
Temporal operators



tomorrow p and q hold

Linear Temporal Logic

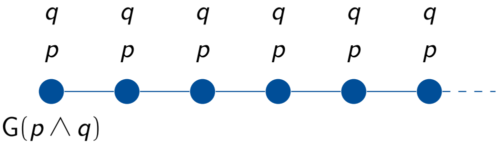
Temporal operators



eventually p and q hold

Linear Temporal Logic

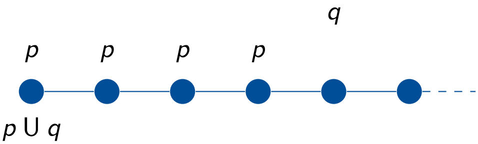
Temporal operators



p and q hold **always**

Linear Temporal Logic

Temporal operators



p holds **until** q holds

Linear Temporal Logic

Examples

$\mathsf{G}\neg(p \wedge q)$	p and q never hold together
$\mathsf{G}(p \rightarrow (\mathsf{X}q \vee \mathsf{XX}q \vee \mathsf{XXX}q))$	whenever p holds, q holds within three steps
$\mathsf{G}(p \rightarrow \mathsf{F}q)$	whenever p holds, q will eventually hold in the future
$\mathsf{G}(\mathsf{G}p \rightarrow \mathsf{F}q)$	whenever it happens that p always holds from there on, q will hold in the future
$\mathsf{GF}p$	p happens infinitely often
$\mathsf{FG}p$	p will become true and will remain true forever
$\mathsf{G}(p \rightarrow \mathsf{G}p)$	whenever p becomes true, it remains true forever
$\neg p \,\mathsf{U}\, q$	p remains false until the first time q holds.

Linear Temporal Logic

Derivable operators

Some syntactic elements are derivable from others:

$$\begin{aligned}
\top &\equiv p \vee \neg p && \text{truth} \\
\bot &\equiv \neg \top && \text{falsity} \\
p \rightarrow q &\equiv \neg p \vee q && \text{implication} \\
\mathsf{F}\phi &\equiv \top \ \mathsf{U}\ \phi && \\
\mathsf{G}\phi &\equiv \neg \mathsf{F} \neg \phi && \\
\phi \ \mathsf{R}\ \psi &\equiv \neg(\neg \phi \ \mathsf{U}\ \neg \psi) && \textbf{release} \text{ operator}
\end{aligned}$$

Linear Temporal Logic

Past operators

LTL with Past (LTL+P) supports **past operators**.

future		**past**	
$X\phi$	tomorrow	$Y\phi$	yesterday
		$Z\phi$	weak yesterday
$F\phi$	eventually	$O\phi$	once
$G\phi$	always	$H\phi$	historically
$\phi \mathbin{U} \psi$	until	$\phi \mathbin{S} \psi$	since
$\phi \mathbin{R} \psi$	release	$\phi \mathbin{T} \psi$	triggered

Linear Temporal Logic

Satisfiability

Satisfiability

The **satisfiability** checking problem asks whether there exists a **state sequence** satisfying a given LTL formula.

Linear Temporal Logic

Satisfiability

LTL satisfiability is an important problem.

- **entailment** is a satisfiability question:

$$\phi \supset \psi \quad \text{iff} \quad \phi \wedge \neg\psi \text{ is unsat.}$$

- **model-checking** can be reduced to satisfiability:

$$M \models \psi \quad \text{iff} \quad \phi_M \supset \psi$$

- **sanity checking** of specifications is a satisfiability question:
  - unsatisfiable specifications are buggy
  - valid specifications are useless
- **STRIPS planning** can be reduced to LTL satisfiability [May+07]

Linear Temporal Logic

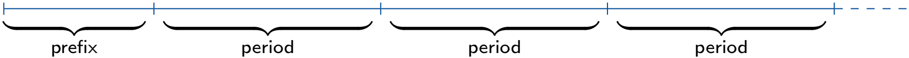
Satisfiability

Theorem ([SC85])

Satisfiability checking for LTL formulas is ***PSPACE-complete****.*

Useful facts:

- LTL+P can be **exponentially** more succinct than LTL, [Mar03]
  but satisfiability is still **PSPACE-complete**. [LP00]
- we can restrict *w.l.o.g.* to **periodic** models:



Linear Temporal Logic

Algorithms

How can we solve the satisfiability problem?

- Reduction to model-checking [Cav+14]
- Temporal resolution [HK03]
- Automata-based techniques [Li+14]
- **tableau methods**

Tableau methods

Tableau methods are a class of algorithms for the **satisfiability** of a variety of different logics.

- introduced in the ’50s for classical propositional logic [Bet59] and first-order logic [Smu95]
- adopted later for modal, temporal, nonmonotonic, many-valued, and substructural logics [DAg+99]

Tableau methods

> ***Now, what is a tableau?*** *[. . .] A tableau method is a formal proof procedure [. . .] with certain characteristics. First, it is a refutation procedure: to show a formula* X *is valid we begin with some syntactical expression intended to assert it is not. [. . .] Next, the expression asserting the invalidity of* X *is broken down syntactically, generally splitting things into several cases. [. . .] Finally there are rules for closing cases. [. . .] A closed tableau beginning with an expression asserting that* X *is not valid is a tableau proof of* X*.*

Melvin Fitting
Handbook of Tableau Methods
[DAg+99]

Tableau methods

> *Now, what is a tableau? [. . .] A tableau method is* ***a formal proof procedure*** *[. . .] with certain characteristics. First, it is a refutation procedure: to show a formula X is valid we begin with some syntactical expression intended to assert it is not. [. . .] Next, the expression asserting the invalidity of X is broken down syntactically, generally splitting things into several cases. [. . .] Finally there are rules for closing cases. [. . .] A closed tableau beginning with an expression asserting that X is not valid is a tableau proof of X.*

Melvin Fitting
Handbook of Tableau Methods
[DAg+99]

Tableau methods

> *Now, what is a tableau? [. . .] A tableau method is a formal proof procedure [. . .] with certain characteristics. First, it is* ***a refutation procedure****: to show a formula X is valid we begin with some syntactical expression intended to assert it is not. [. . .] Next, the expression asserting the invalidity of X is broken down syntactically, generally splitting things into several cases. [. . .] Finally there are rules for closing cases. [. . .] A closed tableau beginning with an expression asserting that X is not valid is a tableau proof of X.*

Melvin Fitting
Handbook of Tableau Methods
[DAg+99]

Tableau methods

> *Now, what is a tableau? [. . .] A tableau method is a formal proof procedure [. . .] with certain characteristics. First, it is a refutation procedure: to show a formula X is valid we begin with some syntactical expression intended to assert it is not. [. . .] Next, the expression asserting the invalidity of X is broken down* ***syntactically****, generally splitting things into several cases. [. . .] Finally there are rules for closing cases. [. . .] A closed tableau beginning with an expression asserting that X is not valid is a tableau proof of X.*

Melvin Fitting
Handbook of Tableau Methods
[DAg+99]

Tableau methods

> *Now, what is a tableau? [. . .] A tableau method is a formal proof procedure [. . .] with certain characteristics. First, it is a refutation procedure: to show a formula X is valid we begin with some syntactical expression intended to assert it is not. [. . .] Next, the expression asserting the invalidity of X is broken down syntactically, generally splitting things into* ***several cases****. [. . .] Finally there are rules for closing cases. [. . .] A closed tableau beginning with an expression asserting that X is not valid is a tableau proof of X.*

Melvin Fitting
Handbook of Tableau Methods
[DAg+99]

Tableau methods

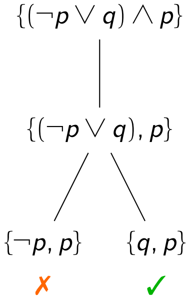
> *Now, what is a tableau? [. . .] A tableau method is a formal proof procedure [. . .] with certain characteristics. First, it is a refutation procedure: to show a formula X is valid we begin with some syntactical expression intended to assert it is not. [. . .] Next, the expression asserting the invalidity of X is broken down syntactically, generally splitting things into several cases. [. . .] Finally there are rules for* ***closing cases****. [. . .] A closed tableau beginning with an expression asserting that X is not valid is a tableau proof of X.*

Melvin Fitting
Handbook of Tableau Methods
[DAg+99]

Tableau methods

Example: propositional logic

sat. formula



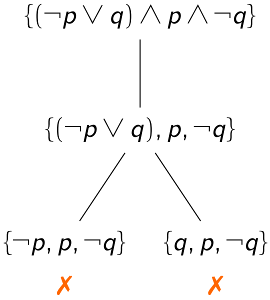
$\{(\neg p \vee q) \wedge p\}$

$\{(\neg p \vee q), p\}$

$\{\neg p, p\}$ ✗

$\{q, p\}$ ✓

unsat. formula



$\{(\neg p \vee q) \wedge p \wedge \neg q\}$

$\{(\neg p \vee q), p, \neg q\}$

$\{\neg p, p, \neg q\}$ ✗

$\{q, p, \neg q\}$ ✗

Tableaux for temporal logics

Tableau methods are among the first reasoning techniques investigated for temporal logics.

- linear temporal logic [Wol83; LP00]
- computation tree logic (CTL) and CTL* [Rey09]
- timed logics [AH93; Del+17]
- temporal logics on continuous time [Rey14]
- many variations of interval temporal logics

Tableaux for LTL

In introducing tableau methods for LTL, we will follow their historical development:

- classic, **graph-shaped** tableaux are simple to understand and useful theoretical tools
- recent, **tree-shaped** tableaux are amenable to efficient implementation

CLASSICAL TABLEAUX

Classical tableau methods for LTL

Here we introduce the classical tableau methods for LTL.

First introduced here:

P. Wolper. "The Tableau Method for Temporal Logic: An Overview." In: *Logique et Analyse* 28 (1985), pp. 119–136

Modern exposition with past operators can be found here:

O. Lichtenstein and A. Pnueli. "Propositional Temporal Logics: Decidability and Completeness." In: *Logic Journal of the IGPL* 8.1 (2000), pp. 55–85. DOI: `10.1093/jigpal/8.1.55`

Classical tableau methods for LTL

Classical methods are **graph-shaped**:

- a graph structure is built from the formula, representing all its possible models
- the graph is traversed to look for actual models of the formula

Expansion rules

All the tableau methods that we will see in this tutorial are based on some **expansion rules**.

rule	ϕ	$\Gamma_1(\phi)$	$\Gamma_2(\phi)$
negation	$\neg\neg\phi$	$\{\phi\}$	
conjunction	$\alpha \wedge \beta$	$\{\alpha, \beta\}$	
	$\neg(\alpha \wedge \beta)$	$\{\neg\alpha\}$	$\{\neg\beta\}$
disjunction	$\alpha \vee \beta$	$\{\alpha\}$	$\{\beta\}$
	$\neg(\alpha \vee \beta)$	$\{\neg\alpha, \neg\beta\}$	
until	$\alpha \, \mathsf{U} \, \beta$	$\{\beta\}$	$\{\alpha, \mathsf{X}(\alpha \, \mathsf{U} \, \beta)\}$
	$\neg(\alpha \, \mathsf{U} \, \beta)$	$\{\neg\alpha, \neg\beta\}$	$\{\neg\beta, \neg\mathsf{X}(\alpha \, \mathsf{U} \, \beta)\}$
eventually	$\mathsf{F}\beta$	$\{\beta\}$	$\{\mathsf{XF}\beta\}$
	$\neg\mathsf{F}\beta$	$\{\neg\beta, \neg\mathsf{XF}\beta\}$	
always	$\mathsf{G}\alpha$	$\{\alpha, \mathsf{XG}\alpha\}$	
	$\neg\mathsf{G}\alpha$	$\{\neg\alpha\}$	$\{\neg\mathsf{XG}\alpha\}$

Expansion rules

Note

Some rules are unary ($\Gamma_2(\phi)$ empty), others are binary (both sets not empty)

For unary rules:

$$\phi \equiv \bigwedge \Gamma_1(\phi)$$

For binary rules:

$$\phi \equiv \left(\bigwedge \Gamma_1(\phi) \vee \bigwedge \Gamma_2(\phi) \right)$$

Example

$$\alpha \, \mathsf{U} \, \beta \equiv \beta \vee (\alpha \wedge \mathsf{X}(\alpha \, \mathsf{U} \, \beta))$$

hence:

$$\Gamma_1(\alpha \, \mathsf{U} \, \beta) = \{\beta\}$$
$$\Gamma_2(\alpha \, \mathsf{U} \, \beta) = \{\alpha, \mathsf{X}(\alpha \, \mathsf{U} \, \beta)\}$$

Closure of a formula

When building the tableau for a formula ϕ, all the formulas and subformulas of ϕ that might be needed are collected into the **closure** of ϕ.

Definition (Closure of a formula)

The **closure** of an LTL formula ϕ is the smallest set of formulas $C(\phi)$ such that:

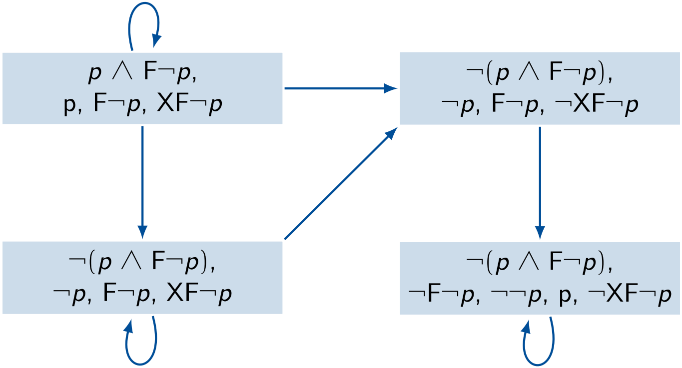
- $\phi \in C(\phi)$
- if $\phi \in C(\phi)$ is not a negation, then $\neg\phi \in C(\phi)$
- if $\psi \in C(\phi)$ has an expansion rule, then $\Gamma_1(\psi) \subseteq C(\psi)$ and $\Gamma_2(\psi) \subseteq C(\psi)$.

Example

The closure of the formula $\phi \equiv p \land \mathsf{F}\neg p$ is the following:

$$C(\phi) = \{p \land \mathsf{F}\neg p, \neg(p \land \mathsf{F}\neg p), \mathsf{F}\neg p, \neg\mathsf{F}\neg p, \mathsf{XF}\neg p, \neg\mathsf{XF}\neg p, p, \neg p\}$$

Tableau construction



$p \wedge \mathsf{F}\neg p$, p, $\mathsf{F}\neg p$, $\mathsf{XF}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$, $\neg p$, $\mathsf{F}\neg p$, $\neg\mathsf{XF}\neg p$

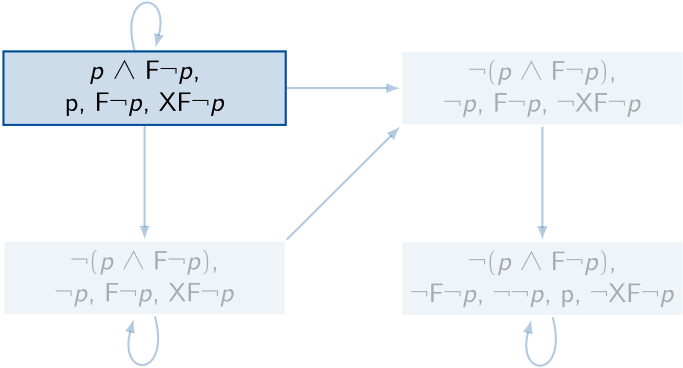
$\neg(p \wedge \mathsf{F}\neg p)$, $\neg p$, $\mathsf{F}\neg p$, $\mathsf{XF}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$, $\neg\mathsf{F}\neg p$, $\neg\neg p$, p, $\neg\mathsf{XF}\neg p$

This is the tableau for $p \wedge \mathsf{F}\neg p$.

The tableau is a **graph**.

Tableau construction



$p \land \mathsf{F}\neg p$,
p, $\mathsf{F}\neg p$, $\mathsf{XF}\neg p$

$\neg(p \land \mathsf{F}\neg p)$,
$\neg p$, $\mathsf{F}\neg p$, $\neg\mathsf{XF}\neg p$

$\neg(p \land \mathsf{F}\neg p)$,
$\neg p$, $\mathsf{F}\neg p$, $\mathsf{XF}\neg p$

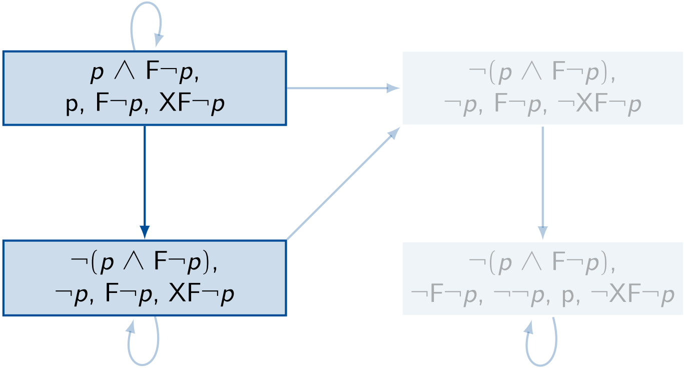
$\neg(p \land \mathsf{F}\neg p)$,
$\neg\mathsf{F}\neg p$, $\neg\neg p$, p, $\neg\mathsf{XF}\neg p$

Each node of the tableau is called an **atom**.

An atom Δ is a **maximally consistent** subset of $C(\phi)$.

- $\psi \in \Delta$ iff $\neg\psi \notin \Delta$
- if $\psi \in \Delta$ has a unary expansion rule, then $\Gamma_1(\psi) \subseteq \Delta$
- if $\psi \in \Delta$ has a binary expansion rule, then either $\Gamma_1(\psi) \subseteq \Delta$ or $\Gamma_2(\psi) \subseteq \Delta$

Tableau construction



$p \wedge \mathrm{F}\neg p$,
p, $\mathrm{F}\neg p$, $\mathrm{XF}\neg p$

$\neg(p \wedge \mathrm{F}\neg p)$,
$\neg p$, $\mathrm{F}\neg p$, $\neg\mathrm{XF}\neg p$

$\neg(p \wedge \mathrm{F}\neg p)$,
$\neg p$, $\mathrm{F}\neg p$, $\mathrm{XF}\neg p$

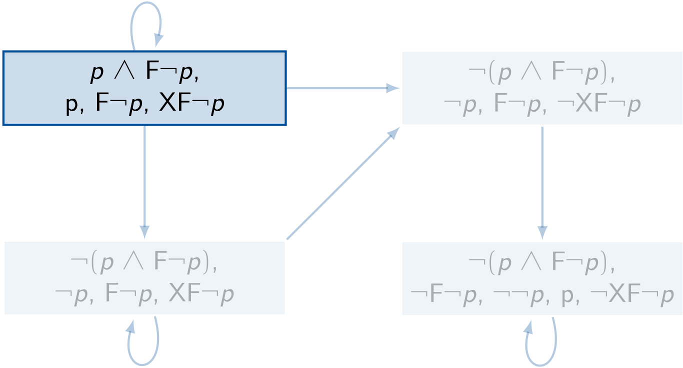
$\neg(p \wedge \mathrm{F}\neg p)$,
$\neg\mathrm{F}\neg p$, $\neg\neg p$, p, $\neg\mathrm{XF}\neg p$

Edges of the tableau represent **locally consistent temporal transitions**.

$\Delta_1 \rightarrow \Delta_2$ iff:

- for each $\mathrm{X}\psi \in \Delta_1$, we have $\psi \in \Delta_2$
- for each $\neg\mathrm{X}\psi \in \Delta_1$, we have $\neg\psi \in \Delta_2$

Tableau construction



$p \wedge F\neg p$,
p, $F\neg p$, $XF\neg p$

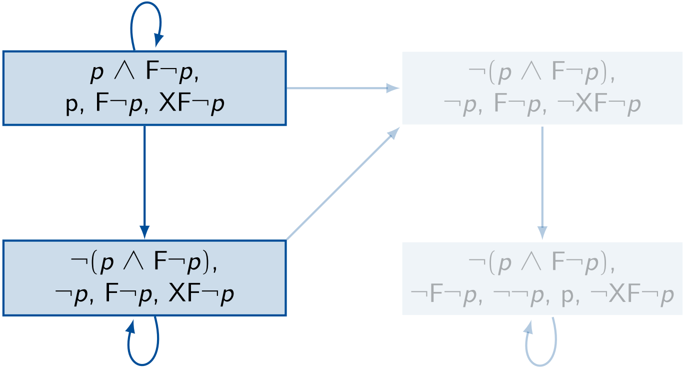
$\neg(p \wedge F\neg p)$,
$\neg p$, $F\neg p$, $\neg XF\neg p$

$\neg(p \wedge F\neg p)$,
$\neg p$, $F\neg p$, $XF\neg p$

$\neg(p \wedge F\neg p)$,
$\neg F\neg p$, $\neg\neg p$, p, $\neg XF\neg p$

An atom Δ is **initial** if $\phi \in \Delta$.

Tableau construction



$p \wedge \mathsf{F}\neg p$,
p, $\mathsf{F}\neg p$, $\mathsf{X}\mathsf{F}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg p$, $\mathsf{F}\neg p$, $\neg\mathsf{X}\mathsf{F}\neg p$

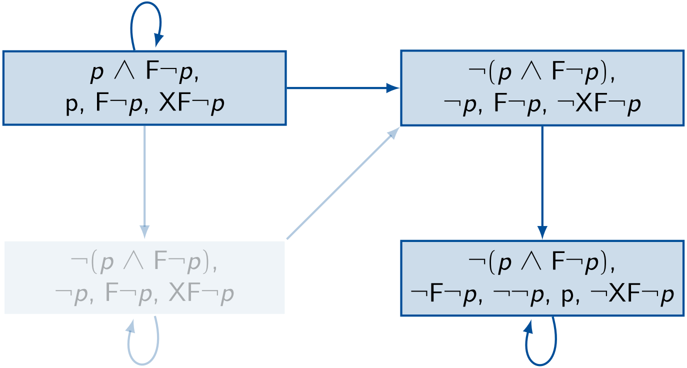
$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg p$, $\mathsf{F}\neg p$, $\mathsf{X}\mathsf{F}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg\mathsf{F}\neg p$, $\neg\neg p$, p, $\neg\mathsf{X}\mathsf{F}\neg p$

Any **infinite** path $\overline{\Delta} = \langle \Delta_0, \Delta_1, \ldots \rangle$ starting from an **initial** atom represents a possible **model** for ϕ.

But we still miss something.

Tableau construction



Any **infinite** path $\overline{\Delta} = \langle \Delta_0, \Delta_1, \ldots \rangle$ starting from an **initial** atom represents a possible **model** for ϕ.

But we still miss something.

Eventualities

Paths in the tableau have some features by construction:

- **locally consistent**, *i.e.*, the single atoms are logically consistent
- **universal** formulas are satisfied *i.e.*, $G\alpha$

However, they are not guaranteed *a priori* to be correct models.

Example

If $\alpha \mathrm{U} \beta \in \Delta$, then we have either $\beta \in \Delta$ or $\alpha, \mathrm{X}(\alpha \mathrm{U} \beta) \in \Delta$.

However, the former case is not guaranteed to ever happen in a path.

Eventualities

Definition (X-eventuality)

An **X-eventuality** is a formula of the form $\mathsf{X}(\alpha \mathbin{\mathsf{U}} \beta)$ or $\mathsf{XF}\beta$.

We must look for paths in the tableau where each X-eventuality is **satisfied**.

Definition (Fulfilling path)

A path $\overline{\Delta} = \langle \Delta_0, \Delta_1, \ldots \rangle$ in a tableau is **fulfilling** if for each $i \geqslant 0$,
if $\mathsf{X}(\alpha \mathbin{\mathsf{U}} \beta) \in \Delta_i$ or $\mathsf{XF}\beta \in \Delta_i$, then there is a $j > i$ such that $\beta \in \Delta_j$.

Finding fulfilling paths

How to find fulfilling paths?

Definition (Strongly connected component)

A **strongly connected component** (SCC) of a graph is a subgraph where each node is **reachable** from each other.

Definition (Fulfilling SCC)

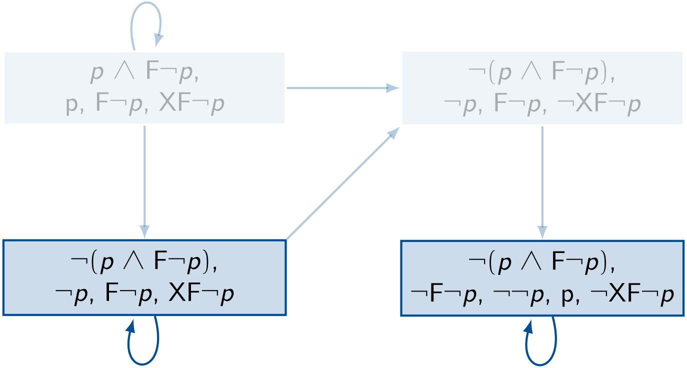
A **fulfilling SCC** of a tableau is an SCC S of the tableau such that if any X-eventuality $\mathsf{X}(\alpha \,\mathsf{U}\, \beta)$ (or $\mathsf{XF}\beta$) appear in a node of S, then β appears in a node of S.

Finding fulfilling paths

Theorem

An LTL formula ϕ is ***satisfiable*** *if and only if the tableau for ϕ contains a* ***fulfilling SCC*** *reachable from an initial atom.*

Tableau construction



$p \land \mathsf{F}\neg p$,
p, $\mathsf{F}\neg p$, $\mathsf{XF}\neg p$

$\neg(p \land \mathsf{F}\neg p)$,
$\neg p$, $\mathsf{F}\neg p$, $\neg\mathsf{XF}\neg p$

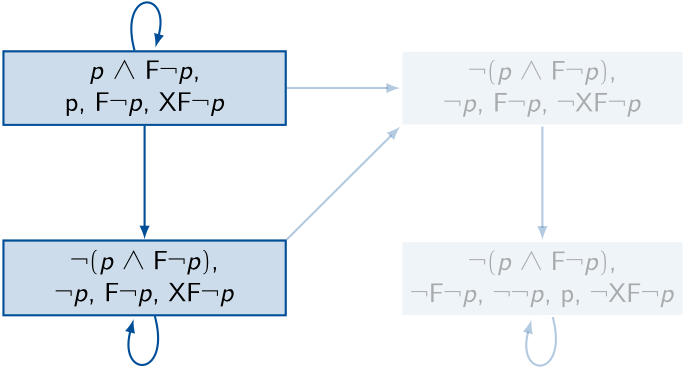
$\neg(p \land \mathsf{F}\neg p)$,
$\neg p$, $\mathsf{F}\neg p$, $\mathsf{XF}\neg p$

$\neg(p \land \mathsf{F}\neg p)$,
$\neg\mathsf{F}\neg p$, $\neg\neg p$, p, $\neg\mathsf{XF}\neg p$

In this case, we have two **fulfilling SCCs** reachable from the initial atom.

$p \land \mathsf{F}\neg p$ is **satisfiable**.

Tableau construction



$p \wedge \mathsf{F}\neg p$,
p, $\mathsf{F}\neg p$, $\mathsf{X}\mathsf{F}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg p$, $\mathsf{F}\neg p$, $\neg\mathsf{X}\mathsf{F}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg p$, $\mathsf{F}\neg p$, $\mathsf{X}\mathsf{F}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg\mathsf{F}\neg p$, $\neg\neg p$, p, $\neg\mathsf{X}\mathsf{F}\neg p$

These give us many **models**.

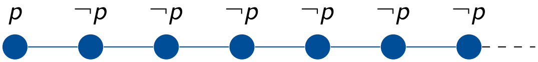
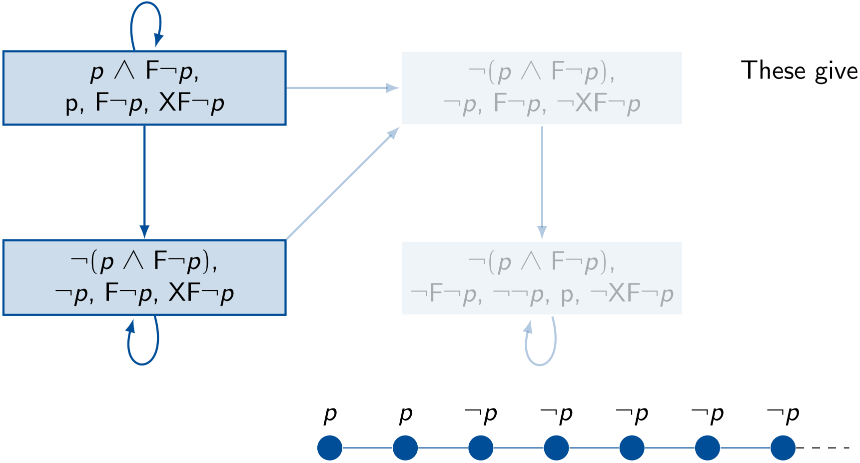


Tableau construction



$p \wedge \mathrm{F}\neg p$,
p, $\mathrm{F}\neg p$, $\mathrm{XF}\neg p$

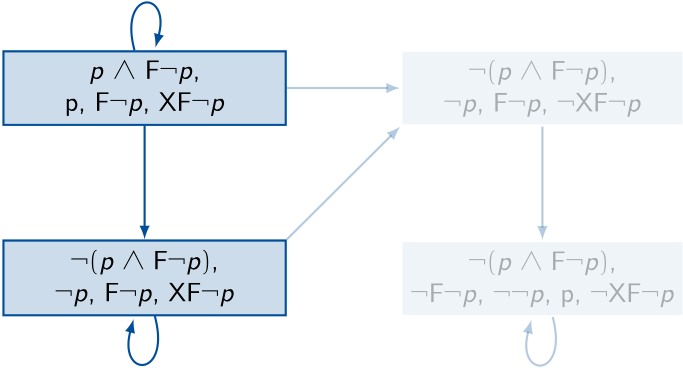
$\neg(p \wedge \mathrm{F}\neg p)$,
$\neg p$, $\mathrm{F}\neg p$, $\neg\mathrm{XF}\neg p$

$\neg(p \wedge \mathrm{F}\neg p)$,
$\neg p$, $\mathrm{F}\neg p$, $\mathrm{XF}\neg p$

$\neg(p \wedge \mathrm{F}\neg p)$,
$\neg\mathrm{F}\neg p$, $\neg\neg p$, p, $\neg\mathrm{XF}\neg p$

These give us many **models**.

Tableau construction



$p \land F\neg p$,
p, $F\neg p$, $XF\neg p$

$\neg(p \land F\neg p)$,
$\neg p$, $F\neg p$, $\neg XF\neg p$

$\neg(p \land F\neg p)$,
$\neg p$, $F\neg p$, $XF\neg p$

$\neg(p \land F\neg p)$,
$\neg F\neg p$, $\neg\neg p$, p, $\neg XF\neg p$

These give us many **models**.

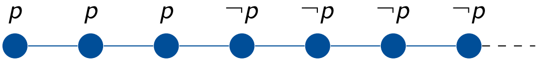
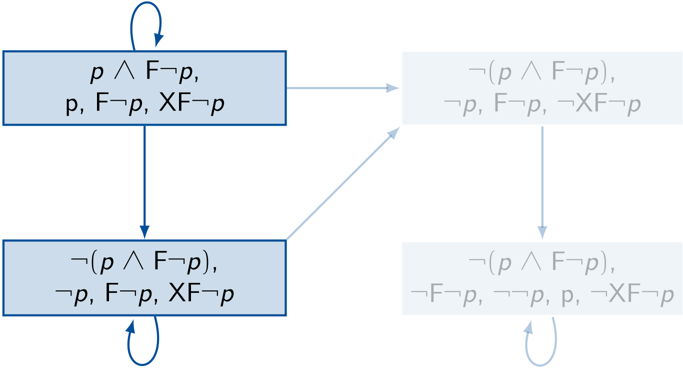


Tableau construction



$p \wedge \mathsf{F}\neg p$,
p, $\mathsf{F}\neg p$, $\mathsf{XF}\neg p$

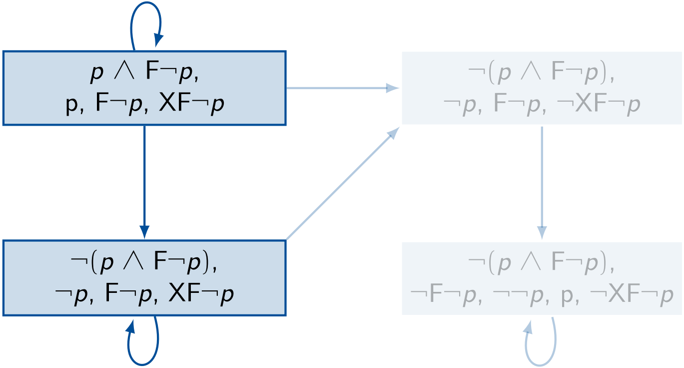
$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg p$, $\mathsf{F}\neg p$, $\neg\mathsf{XF}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg p$, $\mathsf{F}\neg p$, $\mathsf{XF}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg\mathsf{F}\neg p$, $\neg\neg p$, p, $\neg\mathsf{XF}\neg p$

These give us many **models**.

Tableau construction



$p \wedge \mathrm{F}\neg p$,
p, $\mathrm{F}\neg p$, $\mathrm{XF}\neg p$

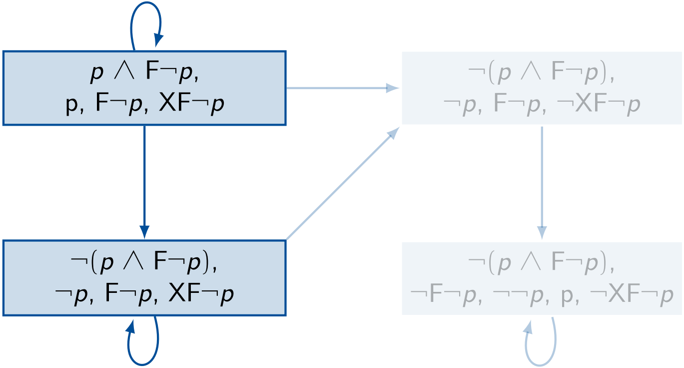
$\neg(p \wedge \mathrm{F}\neg p)$,
$\neg p$, $\mathrm{F}\neg p$, $\neg\mathrm{XF}\neg p$

$\neg(p \wedge \mathrm{F}\neg p)$,
$\neg p$, $\mathrm{F}\neg p$, $\mathrm{XF}\neg p$

$\neg(p \wedge \mathrm{F}\neg p)$,
$\neg\mathrm{F}\neg p$, $\neg\neg p$, p, $\neg\mathrm{XF}\neg p$

These give us many **models**.

Tableau construction



$p \wedge F\neg p$,
p, $F\neg p$, $XF\neg p$

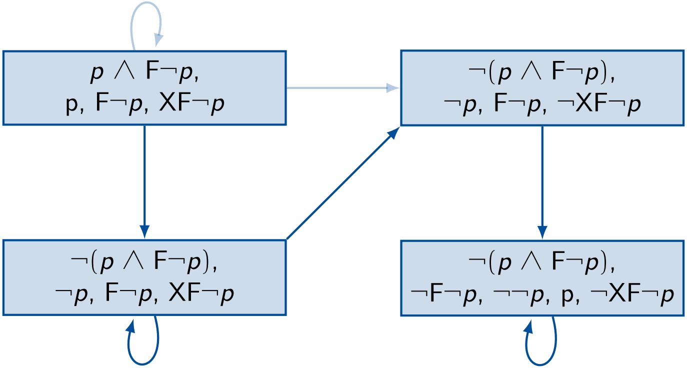
$\neg(p \wedge F\neg p)$,
$\neg p$, $F\neg p$, $\neg XF\neg p$

$\neg(p \wedge F\neg p)$,
$\neg p$, $F\neg p$, $XF\neg p$

$\neg(p \wedge F\neg p)$,
$\neg F\neg p$, $\neg\neg p$, p, $\neg XF\neg p$

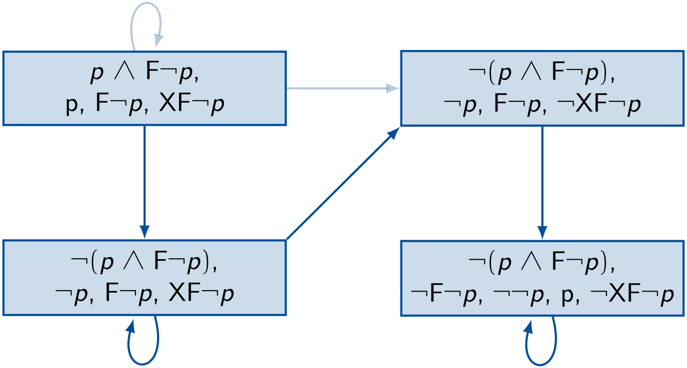
These give us many **models**.

Tableau construction



These give us many **models**.

Tableau construction



$p \wedge \mathrm{F}\neg p$,
p, $\mathrm{F}\neg p$, $\mathrm{XF}\neg p$

$\neg(p \wedge \mathrm{F}\neg p)$,
$\neg p$, $\mathrm{F}\neg p$, $\neg\mathrm{XF}\neg p$

$\neg(p \wedge \mathrm{F}\neg p)$,
$\neg p$, $\mathrm{F}\neg p$, $\mathrm{XF}\neg p$

$\neg(p \wedge \mathrm{F}\neg p)$,
$\neg\mathrm{F}\neg p$, $\neg\neg p$, p, $\neg\mathrm{XF}\neg p$

These give us many **models**.

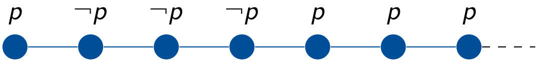
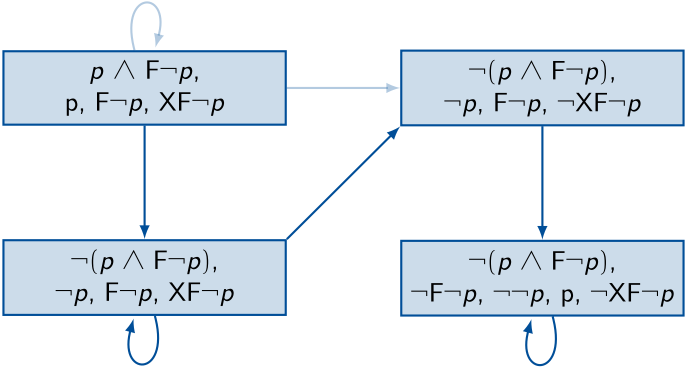


Tableau construction



$p \wedge \mathsf{F}\neg p$,
p, $\mathsf{F}\neg p$, $\mathsf{XF}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg p$, $\mathsf{F}\neg p$, $\neg\mathsf{XF}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg p$, $\mathsf{F}\neg p$, $\mathsf{XF}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg\mathsf{F}\neg p$, $\neg\neg p$, p, $\neg\mathsf{XF}\neg p$

These give us many **models**.

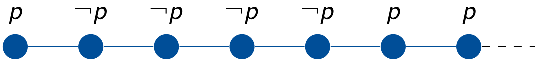
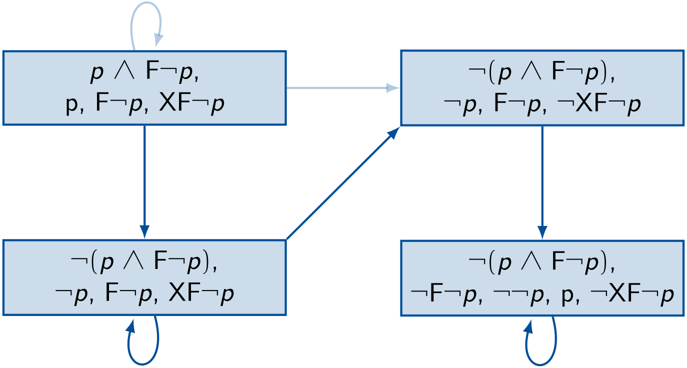


Tableau construction



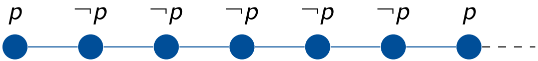
$p \wedge \mathsf{F}\neg p$,
p, $\mathsf{F}\neg p$, $\mathsf{XF}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg p$, $\mathsf{F}\neg p$, $\neg\mathsf{XF}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg p$, $\mathsf{F}\neg p$, $\mathsf{XF}\neg p$

$\neg(p \wedge \mathsf{F}\neg p)$,
$\neg\mathsf{F}\neg p$, $\neg\neg p$, p, $\neg\mathsf{XF}\neg p$

These give us many **models**.



Recap

We can test satisfiability of an LTL formula ϕ with a classic **graph-shaped** tableau.

- Build the graph structure made of all the possible **atoms** and their connecting edges
- find **strongly connected components** and look for fulfilling ones
- if at least a fulfilling SCC is reachable by an initial atom, ϕ is **satisfiable**

Drawbacks

Classical tableaux for LTL are easy to understand and useful theoretical tools.

However, in practice they have some drawbacks:

- the **two-phase** procedure requires to first build a (huge) graph structure
- the graph is then traversed to look for fulfilling SCCs
- the graph is of **exponential size**, requiring huge amounts of memory

Many authors tried to address these problems.

- **incremental** tableaux tried to build the graph nodes **on the fly** [Kes+93]
- a real solution came from **tree-shaped** tableaux [Sch98; Rey16] that we will introduce in a minute.

COFFEE BREAK

TREE-SHAPED TABLEAUX

Tree-shaped tableaux

Recent tableau systems for LTL are **graph-shaped**.

First tree-shaped system by Schwendimann:

S. Schwendimann. "A New One-Pass Tableau Calculus for PLTL." In: *Proc. of the 7th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*. Vol. 1397. LNCS. Springer, 1998, pp. 277–292. DOI: `10.1007/3-540-69778-0\_28`

Most recent one by Reynolds:

M. Reynolds. "A New Rule for LTL Tableaux." In: *Proc. of the 7th International Symposium on Games, Automata, Logics and Formal Verification*. Vol. 226. EPTCS. 2016, pp. 287–301. DOI: `10.4204/EPTCS.226.20`

Tree-shaped tableaux

Recent tableau systems for LTL are **graph-shaped**.

First tree-shaped system by Schwendimann:

S. Schwendimann. “A New One-Pass Tableau Calculus for PLTL.” In: *Proc. of the 7th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*. Vol. 1397. LNCS. Springer, 1998, pp. 277–292. DOI: `10.1007/3-540-69778-0\_28`

Most recent one by Reynolds:

M. Reynolds. “A New Rule for LTL Tableaux.” In: *Proc. of the 7th International Symposium on Games, Automata, Logics and Formal Verification*. Vol. 226. EPTCS. 2016, pp. 287–301. DOI: `10.4204/EPTCS.226.20`

Reynolds' tableau

We'll now introduce the tree-shaped tableau by Reynolds [Rey16]:

- a **tree** is built instead of a graph
- each **accepted** branch of the tree represents a **model**
- a **single pass** is sufficient to build a branch and decide to accept or reject it
- the construction of each branch is completely **independent** from the other branches:
  - easy parallelization
  - efficient SAT encodings are possible (see later)

An assumption

In what follows we will assume any formula ϕ to be in **negated normal form** (NNF).

- negations are only applied to **propositions**
- this is a trivial assumption since any formula can be put into NNF very easily
- example: $\neg \mathsf{F}(p \wedge q) \equiv \mathsf{G}(\neg p \vee \neg q)$

Expansion rules

The negated normal form greatly simplifies the expansion rules:

rule	ϕ	$\Gamma_1(\phi)$	$\Gamma_2(\phi)$
conjunction	$\alpha \wedge \beta$	$\{\alpha, \beta\}$	
disjunction	$\alpha \vee \beta$	$\{\alpha\}$	$\{\beta\}$
until	$\alpha \, \mathsf{U} \, \beta$	$\{\beta\}$	$\{\alpha, \mathsf{X}(\alpha \, \mathsf{U} \, \beta)\}$
release	$\alpha \, \mathsf{R} \, \beta$	$\{\alpha, \beta\}$	$\{\beta, \mathsf{X}(\alpha \, \mathsf{R} \, \beta)\}$
eventually	$\mathsf{F}\beta$	$\{\beta\}$	$\{\mathsf{XF}\beta\}$
always	$\mathsf{G}\alpha$	$\{\alpha, \mathsf{XG}\alpha\}$	

Closure

The definition of closure is slightly simpler as well:

Definition (Closure of a formula)

The **closure** of an LTL formula ϕ is the smallest set of formulas $C(\phi)$ such that:

- $\phi \in C(\phi)$
- if $\psi \in C(\phi)$ has an expansion rule, then $\Gamma_1(\psi) \subseteq C(\psi)$ and $\Gamma_2(\psi) \subseteq C(\psi)$.

Example

The closure of the formula $\phi \equiv p \wedge \mathsf{GF}\neg p$ is the following:

$$C(\phi) = \{p \wedge \mathsf{GF}\neg p, \mathsf{GF}\neg p, \mathsf{XGF}\neg p, \mathsf{F}\neg p, \mathsf{XF}\neg p, p, \neg p\}$$

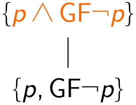
Tableau construction

$\{p \wedge \mathrm{GF}\neg p\}$

The tableau for a formula ϕ is a **tree**:

- nodes u are labelled by $\Gamma(u) \subseteq \mathrm{C}(\phi)$
- the label of the **root** is $\Gamma(u_0) = \{\phi\}$
- to build the tree, we **expand** the formulas in the nodes' labels following the expansion rules.

Tableau construction



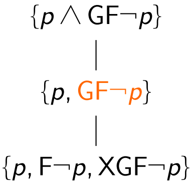
$\{p \wedge \mathsf{GF}\neg p\}$

$\{p, \mathsf{GF}\neg p\}$

The tableau for a formula ϕ is a **tree**:

- nodes u are labelled by $\Gamma(u) \subseteq \mathsf{C}(\phi)$
- the label of the **root** is $\Gamma(u_0) = \{\phi\}$
- to build the tree, we **expand** the formulas in the nodes' labels following the expansion rules.

Tableau construction



$\{p \wedge \mathsf{GF}\neg p\}$

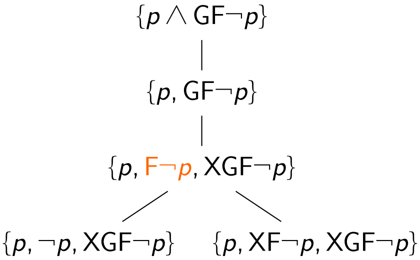
$\{p, \mathsf{GF}\neg p\}$

$\{p, \mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

The tableau for a formula ϕ is a **tree**:

- nodes u are labelled by $\Gamma(u) \subseteq C(\phi)$
- the label of the **root** is $\Gamma(u_0) = \{\phi\}$
- to build the tree, we **expand** the formulas in the nodes' labels following the expansion rules.

Tableau construction



$\{p \wedge \mathrm{GF}\neg p\}$

$\{p, \mathrm{GF}\neg p\}$

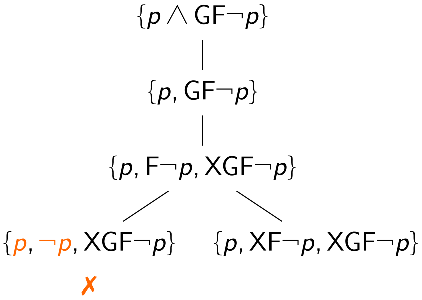
$\{p, \mathrm{F}\neg p, \mathrm{XGF}\neg p\}$

$\{p, \neg p, \mathrm{XGF}\neg p\}$ $\{p, \mathrm{XF}\neg p, \mathrm{XGF}\neg p\}$

The tableau for a formula ϕ is a **tree**:

- nodes u are labelled by $\Gamma(u) \subseteq C(\phi)$
- the label of the **root** is $\Gamma(u_0) = \{\phi\}$
- to build the tree, we **expand** the formulas in the nodes' labels following the expansion rules.

Tableau construction



$\{p \land \mathsf{GF}\neg p\}$

$\{p, \mathsf{GF}\neg p\}$

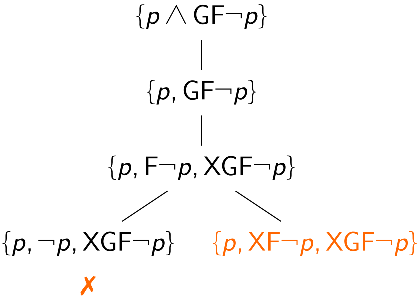
$\{p, \mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{p, \neg p, \mathsf{XGF}\neg p\}$ ✗

$\{p, \mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

When a **contradiction** is found, the branch is **rejected**.

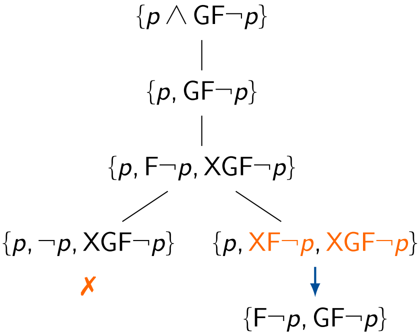
Tableau construction



When a node cannot be expanded further, it is called a **poised node**.

The label of the poised node describes the current **temporal state**.

Tableau construction



$\{p \wedge \mathsf{GF}\neg p\}$

$\{p, \mathsf{GF}\neg p\}$

$\{p, \mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

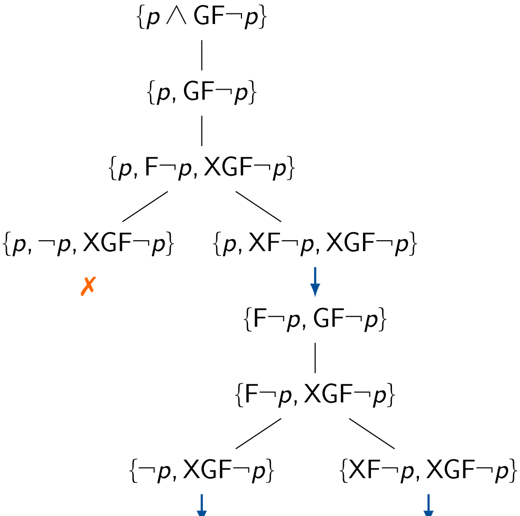
$\{p, \neg p, \mathsf{XGF}\neg p\}$ ✗

$\{p, \mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

The **STEP** rule is applied to poised nodes, advancing to the **next** temporal state.

Tableau construction



$\{p \wedge \mathsf{GF}\neg p\}$

$\{p, \mathsf{GF}\neg p\}$

$\{p, \mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{p, \neg p, \mathsf{XGF}\neg p\}$ $\{p, \mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

✗

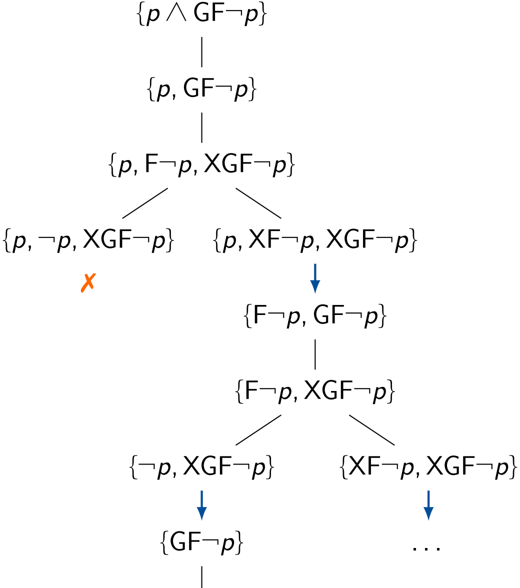
$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{\neg p, \mathsf{XGF}\neg p\}$ $\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

The construction continues in the same way, **expanding** nodes and applying the **STEP** rule.

Tableau construction



$\{p \wedge \mathsf{GF}\neg p\}$

$\{p, \mathsf{GF}\neg p\}$

$\{p, \mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{p, \neg p, \mathsf{XGF}\neg p\}$ $\{p, \mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

✗

$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

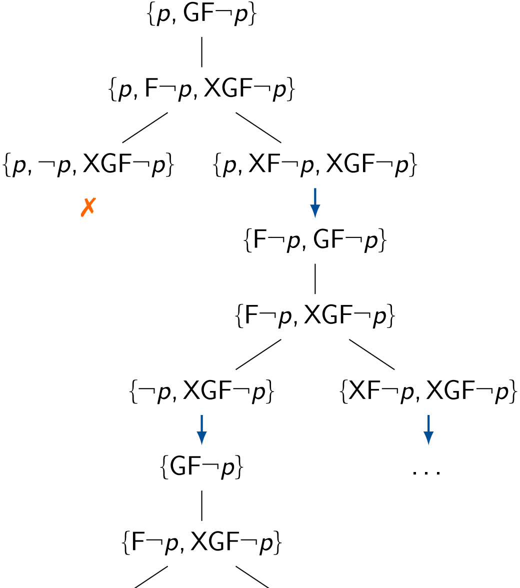
$\{\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{\neg p, \mathsf{XGF}\neg p\}$ $\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{GF}\neg p\}$ $\ldots$

The construction continues in the same way, **expanding** nodes and applying the **STEP** rule.

Tableau construction



$\{p, \mathsf{GF}\neg p\}$

$\{p, \mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{p, \neg p, \mathsf{XGF}\neg p\}$ ✗

$\{p, \mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

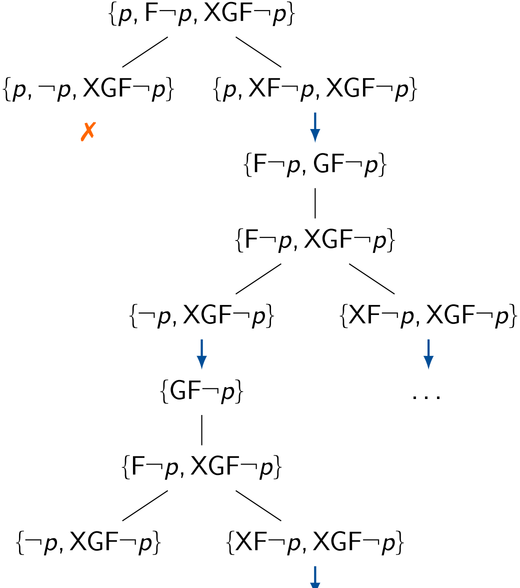
$\{\mathsf{GF}\neg p\}$

$\dots$

$\{\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

The construction continues in the same way, **expanding** nodes and applying the **STEP** rule.

Tableau construction



$\{p, F\neg p, XGF\neg p\}$

$\{p, \neg p, XGF\neg p\}$ ✗

$\{p, XF\neg p, XGF\neg p\}$

$\{F\neg p, GF\neg p\}$

$\{F\neg p, XGF\neg p\}$

$\{\neg p, XGF\neg p\}$

$\{XF\neg p, XGF\neg p\}$

$\{GF\neg p\}$

$\ldots$

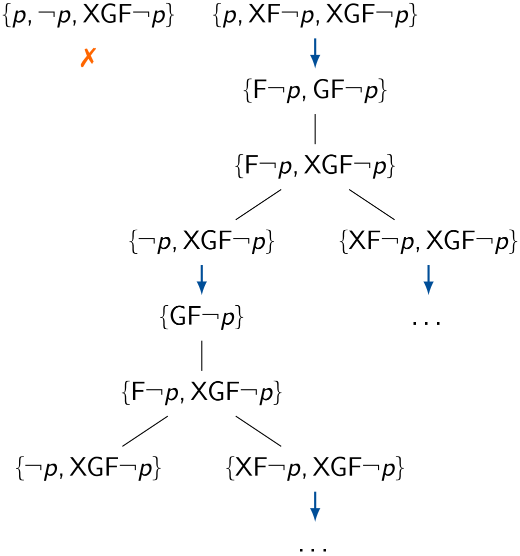
$\{F\neg p, XGF\neg p\}$

$\{\neg p, XGF\neg p\}$

$\{XF\neg p, XGF\neg p\}$

The construction continues in the same way, **expanding** nodes and applying the **STEP** rule.

Tableau construction



$\{p, \neg p, \mathsf{XGF}\neg p\}$ $\{p, \mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

✗

$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{\neg p, \mathsf{XGF}\neg p\}$ $\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{GF}\neg p\}$ ...

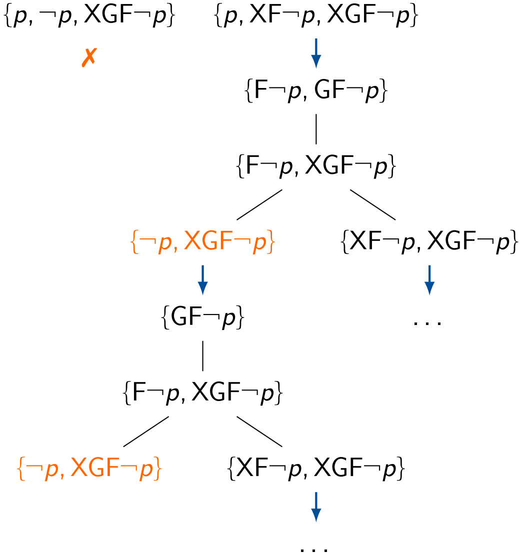
$\{\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{\neg p, \mathsf{XGF}\neg p\}$ $\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

...

The construction continues in the same way, **expanding** nodes and applying the **STEP** rule.

Tableau construction



$\{p, \neg p, \mathsf{XGF}\neg p\}$ ✗

$\{p, \mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{\neg p, \mathsf{XGF}\neg p\}$ $\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{GF}\neg p\}$ ...

$\{\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

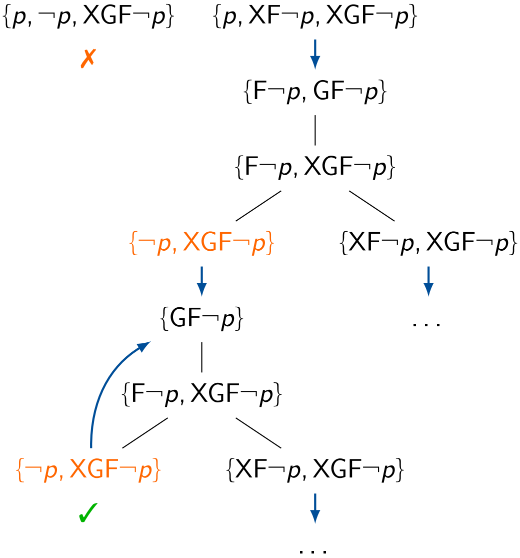
$\{\neg p, \mathsf{XGF}\neg p\}$ $\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

...

Now, we can notice this repetition:

- two poised nodes u and v with **the same label** appear in a branch
- all the X-**eventualities** requested in u (in this case none) have been fulfilled between u and v
- the model can infinitely **loop** through these states
- the **LOOP** rule accepts the branch

Tableau construction



$\{p, \neg p, \mathrm{XGF}\neg p\}$ ✗ $\quad \{p, \mathrm{XF}\neg p, \mathrm{XGF}\neg p\}$

$\{\mathrm{F}\neg p, \mathrm{GF}\neg p\}$

$\{\mathrm{F}\neg p, \mathrm{XGF}\neg p\}$

$\{\neg p, \mathrm{XGF}\neg p\}$ $\quad \{\mathrm{XF}\neg p, \mathrm{XGF}\neg p\}$

$\{\mathrm{GF}\neg p\}$ $\quad \dots$

$\{\mathrm{F}\neg p, \mathrm{XGF}\neg p\}$

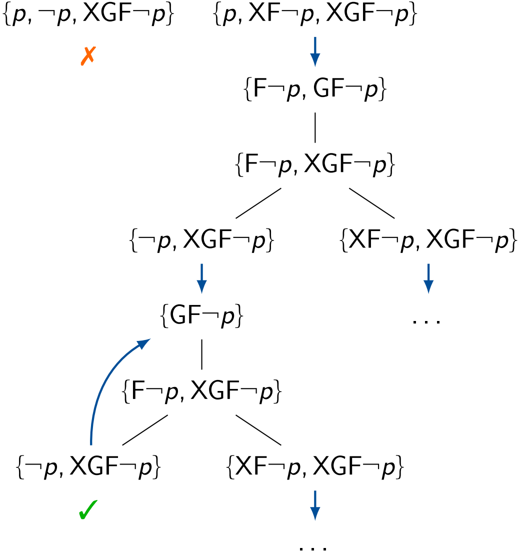
$\{\neg p, \mathrm{XGF}\neg p\}$ ✓ $\quad \{\mathrm{XF}\neg p, \mathrm{XGF}\neg p\}$

$\dots$

Now, we can notice this repetition:

- two poised nodes u and v with **the same label** appear in a branch
- all the X-**eventualities** requested in u (in this case none) have been fulfilled between u and v
- the model can infinitely **loop** through these states
- the **LOOP** rule accepts the branch

Tableau construction



$\{p, \neg p, \mathrm{XGF}\neg p\}$ $\{p, \mathrm{XF}\neg p, \mathrm{XGF}\neg p\}$

✗

$\{\mathrm{F}\neg p, \mathrm{GF}\neg p\}$

$\{\mathrm{F}\neg p, \mathrm{XGF}\neg p\}$

$\{\neg p, \mathrm{XGF}\neg p\}$ $\{\mathrm{XF}\neg p, \mathrm{XGF}\neg p\}$

$\{\mathrm{GF}\neg p\}$ . . .

$\{\mathrm{F}\neg p, \mathrm{XGF}\neg p\}$

$\{\neg p, \mathrm{XGF}\neg p\}$ $\{\mathrm{XF}\neg p, \mathrm{XGF}\neg p\}$

✓

. . .

A **model** for the formula can be extracted from the **poised nodes** of the branch.

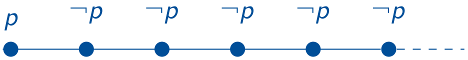
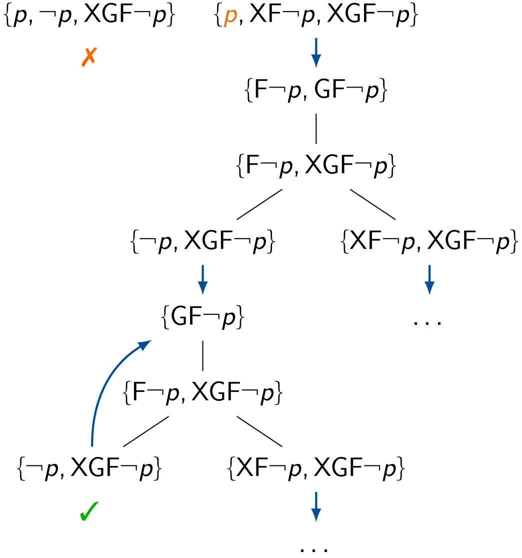


Tableau construction



$\{p, \neg p, \mathsf{XGF}\neg p\}$ ✗

$\{p, \mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{GF}\neg p\}$

. . .

$\{\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{\neg p, \mathsf{XGF}\neg p\}$ ✓

$\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

. . .

A **model** for the formula can be extracted from the **poised nodes** of the branch.

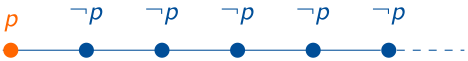
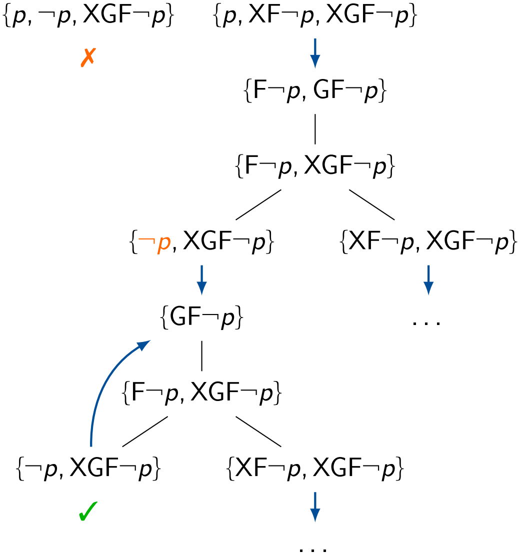


Tableau construction



$\{p, \neg p, \mathsf{XGF}\neg p\}$ ✗

$\{p, \mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{\neg p, \mathsf{XGF}\neg p\}$ $\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{GF}\neg p\}$ …

$\{\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{\neg p, \mathsf{XGF}\neg p\}$ ✓ $\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

…

A **model** for the formula can be extracted from the **poised nodes** of the branch.

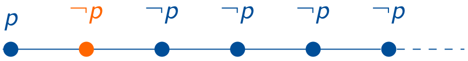
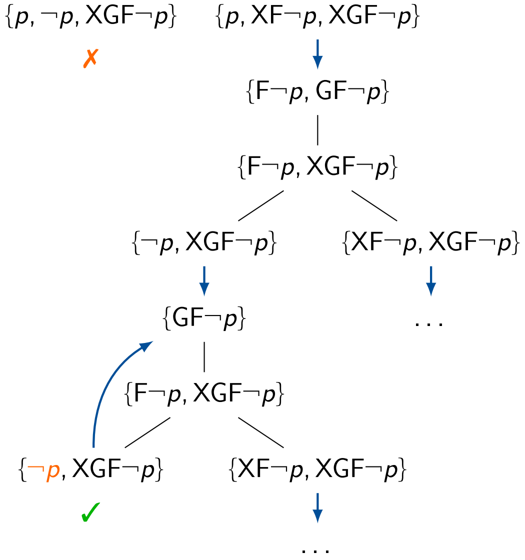


Tableau construction



A **model** for the formula can be extracted from the **poised nodes** of the branch.

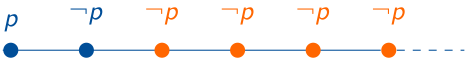
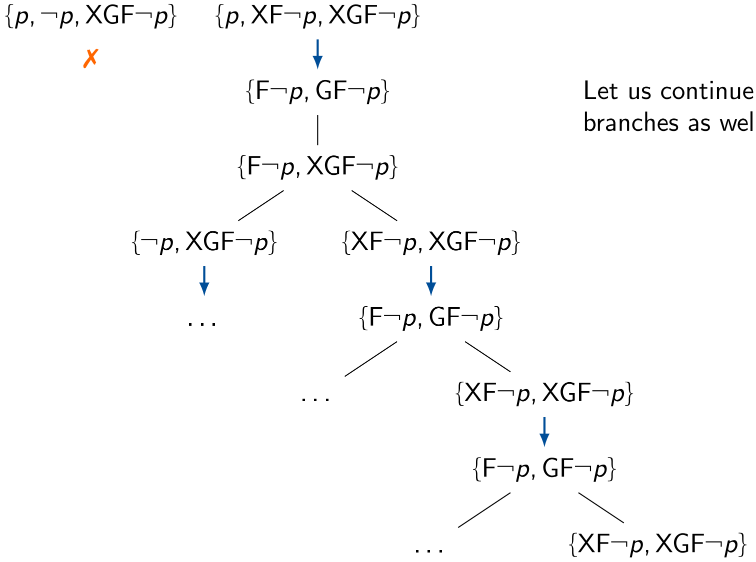


Tableau construction



$\{p, \neg p, \mathrm{XGF}\neg p\}$ $\{p, \mathrm{XF}\neg p, \mathrm{XGF}\neg p\}$

✗

$\{\mathrm{F}\neg p, \mathrm{GF}\neg p\}$

$\{\mathrm{F}\neg p, \mathrm{XGF}\neg p\}$

$\{\neg p, \mathrm{XGF}\neg p\}$ $\{\mathrm{XF}\neg p, \mathrm{XGF}\neg p\}$

. . . $\{\mathrm{F}\neg p, \mathrm{GF}\neg p\}$

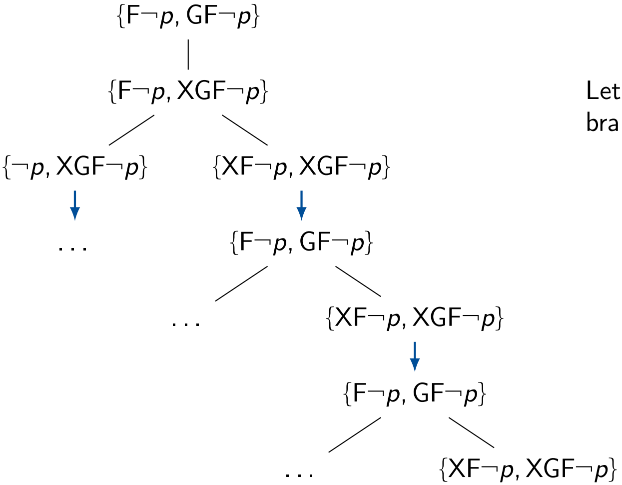
. . . $\{\mathrm{XF}\neg p, \mathrm{XGF}\neg p\}$

$\{\mathrm{F}\neg p, \mathrm{GF}\neg p\}$

. . . $\{\mathrm{XF}\neg p, \mathrm{XGF}\neg p\}$

Let us continue the construction of the other branches as well.

Tableau construction



$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\{\neg p, \mathsf{XGF}\neg p\}$ $\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\ldots$ $\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

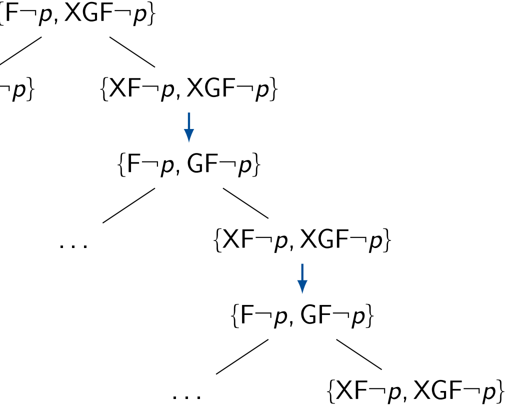
$\ldots$ $\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

$\ldots$ $\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

Let us continue the construction of the other branches as well.

Tableau construction



$\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\neg p\}$ $\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

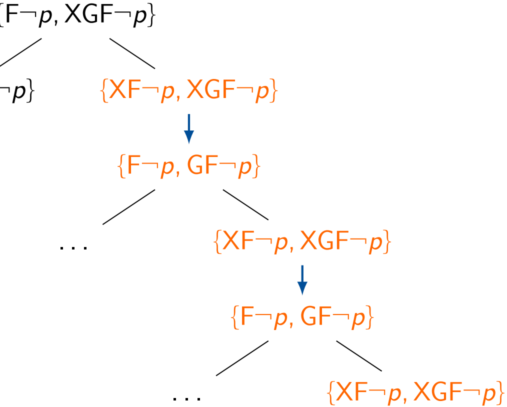
$\dots$ $\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

$\dots$ $\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

Let us continue the construction of the other branches as well.

Tableau construction



$\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\neg p\}$

$\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

$\ldots$

$\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

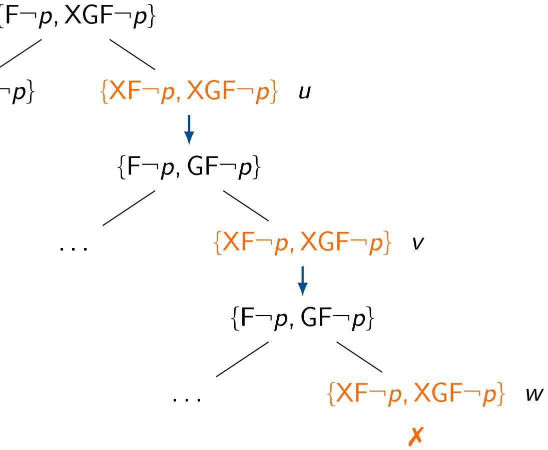
$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

$\ldots$

$\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$

The rightmost branch is expanding **indefinitedly**.

Tableau construction



$\mathsf{F}\neg p, \mathsf{XGF}\neg p\}$

$\neg p\}$

$\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$ u

$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

…

$\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$ v

$\{\mathsf{F}\neg p, \mathsf{GF}\neg p\}$

…

$\{\mathsf{XF}\neg p, \mathsf{XGF}\neg p\}$ w

✗

However, we can notice that:

- there are three poised nodes u, v and w with the **same label**
- all the X-**eventualities** requested in the nodes and fulfilled between v and w (none in this case) are also fulfilled between u and v.
- hence, the branch segment between v and w is doing **redundant work**

In this case, the **PRUNE** rule rejects the branch.

Tableau construction

In summary the tableau for ϕ is built as follows:

- starting from the root $\{\phi\}$ the **expansion rules** are applied until we find **poised nodes**
- the **termination rules** are checked against poised nodes:
  - the **CONTRADICTION** rule rejects propositional contradictions.
  - the **LOOP** rule detects infinite periodic models
  - the **EMPTY** rule detects finite prefixes that can continue arbitrarily
  - the **PRUNE** rule rejects unfulfilling branches
- if no termination rule triggers, the **STEP** rule advances to the next temporal state
- the formula is **satisfiable** iff there is an accepted branch

Tableau construction

In summary the tableau for ϕ is built as follows:

- starting from the root $\{\phi\}$ the **expansion rules** are applied until we find **poised nodes**
- the **termination rules** are checked against poised nodes:
  - the **CONTRADICTION** rule rejects propositional contradictions.
  - the **LOOP** rule detects infinite periodic models
  - the **EMPTY** rule detects finite prefixes that can continue arbitrarily
  - the **PRUNE** rule rejects unfulfilling branches
- if no termination rule triggers, the **STEP** rule advances to the next temporal state
- the formula is **satisfiable** iff there is an accepted branch

Definition (CONTRADICTION rule)

If u is a poised node with $\{p, \neg p\} \subseteq \Gamma(u)$ for some proposition p, the branch is **rejected**.

Tableau construction

In summary the tableau for ϕ is built as follows:

- starting from the root $\{\phi\}$ the **expansion rules** are applied until we find **poised nodes**
- the **termination rules** are checked against poised nodes:
  - the **CONTRADICTION** rule rejects propositional contradictions.
  - the **LOOP** rule detects infinite periodic models
  - the **EMPTY** rule detects finite prefixes that can continue arbitrarily
  - the **PRUNE** rule rejects unfulfilling branches
- if no termination rule triggers, the **STEP** rule advances to the next temporal state
- the formula is **satisfiable** iff there is an accepted branch

Definition (LOOP rule)

If there are two poised nodes u and v such that $\Gamma(u) = \Gamma(v)$ and all the X-**eventualities** requested in u are fulfilled between u and v, the branch is **accepted**.

Tableau construction

In summary the tableau for ϕ is built as follows:

- starting from the root $\{\phi\}$ the **expansion rules** are applied until we find **poised nodes**
- the **termination rules** are checked against poised nodes:
  - the **CONTRADICTION** rule rejects propositional contradictions.
  - the **LOOP** rule detects infinite periodic models
  - the **EMPTY** rule detects finite prefixes that can continue arbitrarily
  - the **PRUNE** rule rejects unfulfilling branches
- if no termination rule triggers, the **STEP** rule advances to the next temporal state
- the formula is **satisfiable** iff there is an accepted branch

Definition (EMPTY rule)

If there is a poised node u with $\Gamma(u) = \varnothing$, the branch is **accepted**.

Tableau construction

In summary the tableau for ϕ is built as follows:

- starting from the root $\{\phi\}$ the **expansion rules** are applied until we find **poised nodes**
- the **termination rules** are checked against poised nodes:
  - the **CONTRADICTION** rule rejects propositional contradictions.
  - the **LOOP** rule detects infinite periodic models
  - the **EMPTY** rule detects finite prefixes that can continue arbitrarily
  - the **PRUNE** rule rejects unfulfilling branches
- if no termination rule triggers, the **STEP** rule advances to the next temporal state
- the formula is **satisfiable** iff there is an accepted branch

Definition (PRUNE rule)

If there are three poised nodes u, v and w such that $\Gamma(u) = \Gamma(v) = \Gamma(w)$ and all the **X-eventualities** requested in the nodes and fulfilled between v and w are fulfilled between u and v as well, the branch is **rejected**.

Tableau construction

In summary the tableau for ϕ is built as follows:

- starting from the root $\{\phi\}$ the **expansion rules** are applied until we find **poised nodes**
- the **termination rules** are checked against poised nodes:
    - the **CONTRADICTION** rule rejects propositional contradictions.
    - the **LOOP** rule detects infinite periodic models
    - the **EMPTY** rule detects finite prefixes that can continue arbitrarily
    - the **PRUNE** rule rejects unfulfilling branches
- if no termination rule triggers, the **STEP** rule advances to the next temporal state
- the formula is **satisfiable** iff there is an accepted branch

Definition (STEP rule)

If u is a poised node, a child u' is created with $\Gamma(u') = \{\alpha \mid \mathsf{X}\alpha \in \Gamma(u)\}$

A closer look at the PRUNE rule

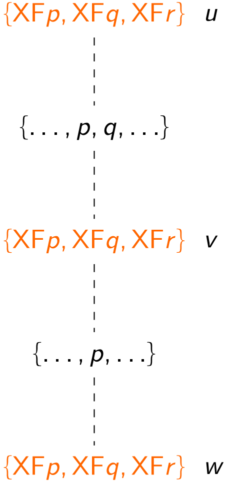
Definition (PRUNE rule)

If there are three poised nodes u, v and w such that $\Gamma(u) = \Gamma(v) = \Gamma(w)$ and all the X-**eventualities** requested in the nodes and fulfilled between v and w are fulfilled between u and v as well, the branch is **rejected**.

The **PRUNE** rule was the major novelty of Reynolds' tableau.

- it rejects branches that would develop infinitely because of unfulfillable **temporal requests**
- it is nevertheless quite difficult to get at first:
  - why wait for **three** occurrences of a label?
  - how can we be sure not to reject **useful** branches?
  - which kind of **models** are excluded?
- let's take a look

Why three occurrences?



$\{\mathsf{XF}p, \mathsf{XF}q, \mathsf{XF}r\}$ u

$\{\ldots, p, q, \ldots\}$

$\{\mathsf{XF}p, \mathsf{XF}q, \mathsf{XF}r\}$ v

$\{\ldots, p, \ldots\}$

$\{\mathsf{XF}p, \mathsf{XF}q, \mathsf{XF}r\}$ w

Let us view it from a different perspective:

- we are not looking for **three** nodes, but for **two** segments of the branch
- the segment between v and w is doing **redundant work** with regards to the segment between u and v.
- since the labels are the same, **any useful segment** that can be developed after w can also be developed after v.

Why three occurrences?

$\{\mathsf{XF}p, \mathsf{XF}q, \mathsf{XF}r\}$ u

$\{\dots, p, q, \dots\}$

$\{\mathsf{XF}p, \mathsf{XF}q, \mathsf{XF}r\}$ v

$\{\dots, p, \dots\}$

$\{\mathsf{XF}p, \mathsf{XF}q, \mathsf{XF}r\}$ w

Let us view it from a different perspective:

- we are not looking for **three** nodes, but for **two** segments of the branch
- the segment between v and w is doing **redundant work** with regards to the segment between u and v.
- since the labels are the same, **any useful segment** that can be developed after w can also be developed after v.

Why three occurrences?

But what if we do not wait for the third occurrence? Consider:

$$\underbrace{p \wedge \mathsf{G}(p \leftrightarrow \mathsf{X}\neg p)}_{1} \wedge \underbrace{\mathsf{GF}q_1 \wedge \mathsf{GF}q_2}_{2} \wedge \underbrace{\mathsf{G}\neg(q_1 \wedge q_2)}_{3} \wedge \underbrace{\mathsf{G}(q_1 \rightarrow \neg p) \wedge \mathsf{G}(q_2 \rightarrow \neg p)}_{4}$$

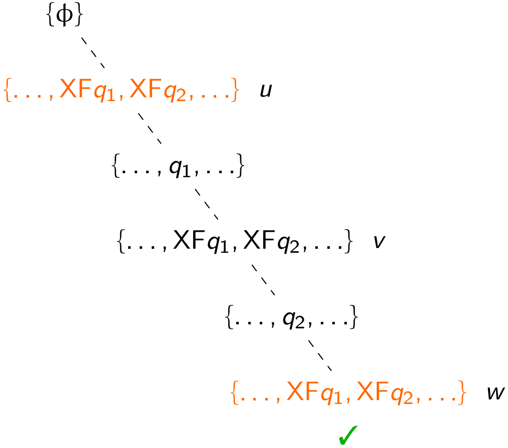
This formula says:

1. p holds at first, and p and $\neg p$ **alternate**
2. q_1 and q_2 holds **infinitely often**
3. q_1 and q_2 cannot hold **together**
4. q_1 and q_2 only hold when $\neg p$ holds

Why three occurrences?

Let us look at the tableau for that formula.

- since q_1 and q_2 cannot hold together, we can only fulfill **one request at a time**
- there is no way to fulfill **both** between a single repetition of the label
- rejecting the branch at v would be **wrong**
- instead, by waiting we give time to the **LOOP** rule to trigger



$\{\phi\}$

$\{\ldots, XFq_1, XFq_2, \ldots\}$ u

$\{\ldots, q_1, \ldots\}$

$\{\ldots, XFq_1, XFq_2, \ldots\}$ v

$\{\ldots, q_2, \ldots\}$

$\{\ldots, XFq_1, XFq_2, \ldots\}$ w

✓

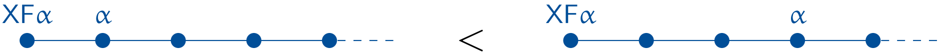
The PRUNE rule, a model-theoretic view

Since each branch of the tableau represents a model of the formula,
the **PRUNE** rule excludes many possible models.

- Which kind of models are excluded?
- Are we sure we do not exclude too many of them?

The PRUNE rule, a model-theoretic view

Consider these two different models:



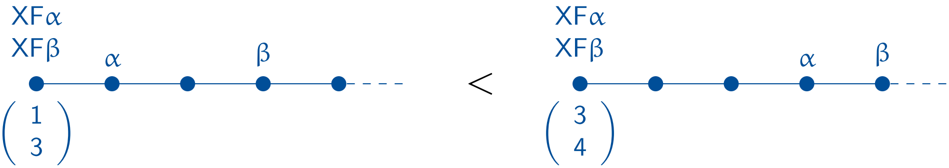
In the first, X Fα is fulfilled earlier.

We consider the first model **less than** the second in a suitably defined ordering relation.

The PRUNE rule, a model-theoretic view

More precisely, is ψ is an X-eventuality:

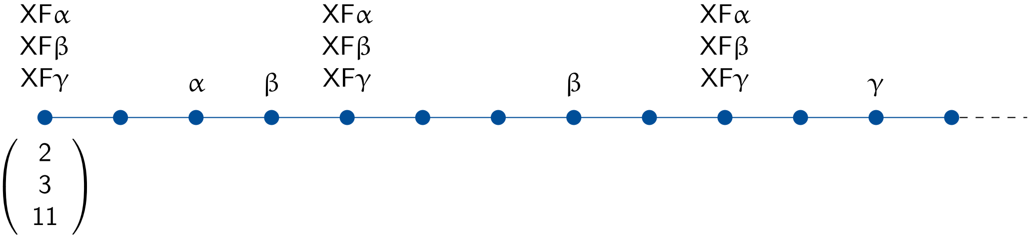
- if ψ holds at step i, $d_i(\psi)$ is the number of states **elapsed** before ψ is **fulfilled**
  - zero otherwise
- $d_i < d_i'$ is defined **component-wise**
- $\sigma < \sigma'$ if $\langle d_0, d_1, \ldots \rangle < \langle d_0', d_1', \ldots \rangle$ **lexicographically**



Minimal models in this pre order are called **greedy**.

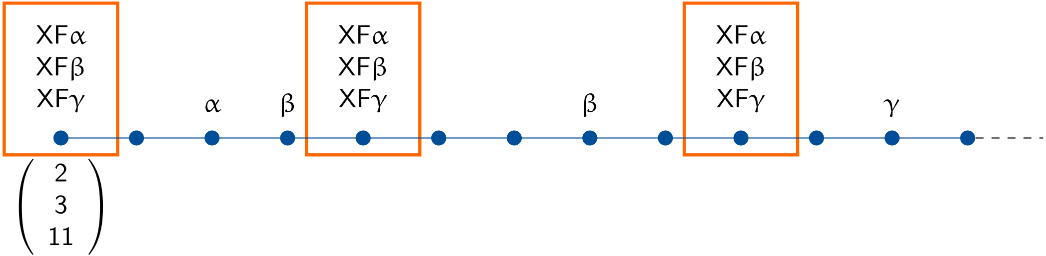
The PRUNE rule, a model-theoretic view

Consider this model:



The PRUNE rule, a model-theoretic view

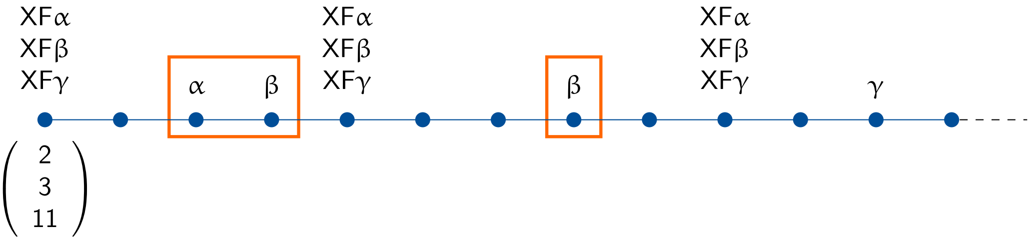
Consider this model:



Suppose that **the same formulas** hold at the highlighted states.

The PRUNE rule, a model-theoretic view

Consider this model:

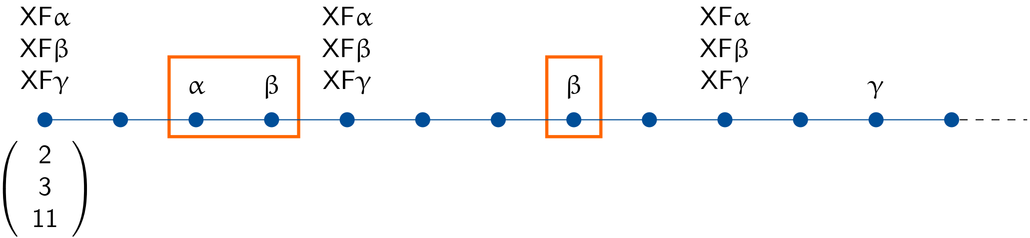


Suppose that **the same formulas** hold at the highlighted states:

- note that the X-eventualities fulfilled between the second and third state are fulfilled between the first and second ones as well

The PRUNE rule, a model-theoretic view

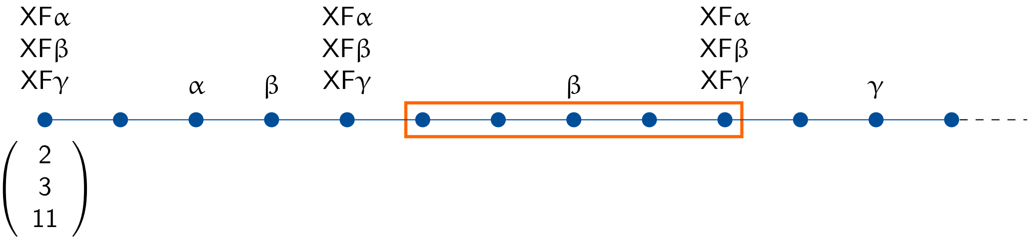
Consider this model:



That is the triggering condition of the **PRUNE** rule!

The PRUNE rule, a model-theoretic view

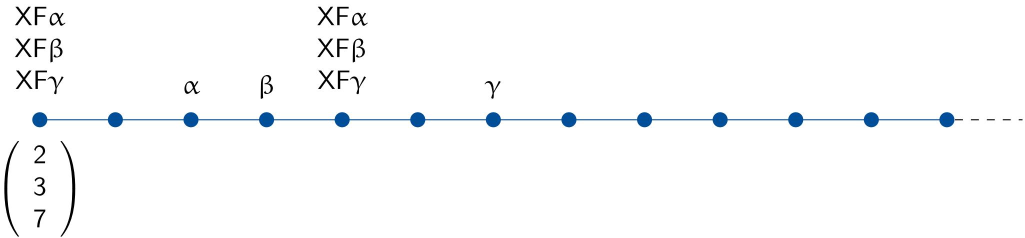
Consider this model:



Note that we can **remove** the middle segment, still obtaining a correct model.

The PRUNE rule, a model-theoretic view

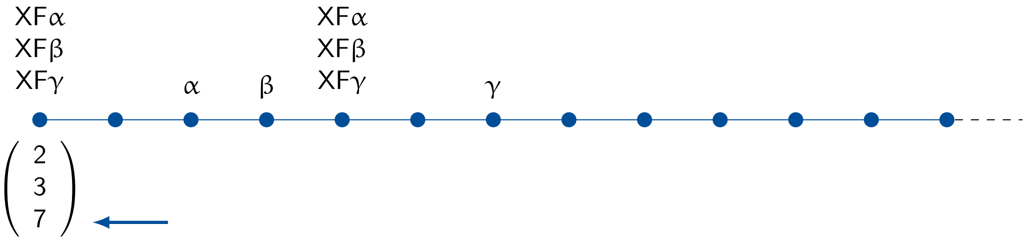
Consider this model:



Note that we can **remove** the middle segment, still obtaining a correct model.

The PRUNE rule, a model-theoretic view

Consider this model:



But then, the initial distance vector decreases: the new model is **smaller** than the old one!

The PRUNE rule, a model-theoretic view

In an accepted branch of the tableau, the **PRUNE** rule is never triggered.

- then in the model $\overline{\sigma}$ extracted from the branch it cannot happen what shown before
- hence $\overline{\sigma}$ is **greedy**

Theorem

Let $\overline{\sigma}$ a model extracted from the tableau for ϕ. Then $\overline{\sigma}$ is greedy.

The PRUNE rule, a model-theoretic view

But then, are we loosing important models? No.

Theorem

Let ϕ be a ***satisfiable*** *LTL formula. Then, there is a* ***greedy*** *model satisfying ϕ.*

Reynolds' tableau, final remarks

Reynolds' tableau is the state of the art of tableau methods for linear-time temporal logics.

- each branch can be developed **independently** from the others
  - embarassingly **parallel** implementation achieves great scalability [MR17]
  - very efficient **SAT encodings** possible (see later) [GGM19]
- the modular structure and clear model-theoretic interpretation makes it very **extensible**:
  - LTL with **past operators** [GMR17]
  - LTL over **finite traces** (just drop the **LOOP** rule)
  - **timed** logics (see later) [Gea+21a]
  - first-order LTL **modulo theories**
    - see our talk at IJCAI, July 28th, 15:30, *Knowledge Representation and Reasoning* session
  - let us now see an example of these extensions: a Reynolds' style tableau for **TPTL**

TABLEAUX FOR TIMED LOGICS

Timed Temporal Logics

A **real-time system** is commonly described as a system that

> *"controls an environment by receiving data, processing them, and returning the results sufficiently quickly to affect the environment at that time"*.

Timed Temporal Logics

A **real-time system** is commonly described as a system that

> *"controls an environment by receiving data, processing them, and returning the results sufficiently quickly to affect the environment at that time"*.

- its correctness does not depend only on their logical correctness, but also on their **response time**;

Timed Temporal Logics

A **real-time system** is commonly described as a system that

> “*controls an environment by receiving data, processing them, and returning the results sufficiently quickly to affect the environment at that time*”.

- its correctness does not depend only on their logical correctness, but also on their **response time**;
- most of the *mission* or *safety critical* systems are real-time: their formal correctness is an aspect that cannot be overlooked.

Timed Temporal Logics

In classical LTL, we can express the request-response property:

$$\varphi \coloneqq \mathsf{G}(\mathit{request} \rightarrow \mathsf{F}\mathit{response})$$

Timed Temporal Logics

In classical LTL, we can express the request-response property:

$$\varphi \coloneqq \mathsf{G}(\mathit{request} \rightarrow \mathsf{F}\mathit{response})$$

We do **not** know the exact times at which the request and the response actually take place: the only thing we know is the **temporal ordering** between these two events.

LTL ⇒ **qualitative** time requirements only,
not suitable for real-time properties.

Timed temporal logics have the goal of solving these limitations.

Timed State Sequence

Definition (Timed state sequence)

A time sequence $\tau = \tau_0\tau_1\tau_2\ldots$ is an infinite sequence of times $\tau_i \in \mathbb{N}$, for all $i \geqslant 0$, that satisfies the following conditions:

Timed State Sequence

Definition (Timed state sequence)

A time sequence $\tau = \tau_0\tau_1\tau_2\ldots$ is an infinite sequence of times $\tau_i \in \mathbb{N}$, for all $i \geqslant 0$, that satisfies the following conditions:

1. *monotonicity*: $\tau_i \leqslant \tau_{i+1}$, for all $i \geqslant 0$;
2. *progress*: for all $t \in \mathbb{N}$, there exists $i \geqslant 0$ such that $\tau_i > t$.

Timed State Sequence

Definition (Timed state sequence)

A time sequence $\tau = \tau_0 \tau_1 \tau_2 \dots$ is an infinite sequence of times $\tau_i \in \mathbb{N}$, for all $i \geqslant 0$, that satisfies the following conditions:

1. *monotonicity*: $\tau_i \leqslant \tau_{i+1}$, for all $i \geqslant 0$;
2. *progress*: for all $t \in \mathbb{N}$, there exists $i \geqslant 0$ such that $\tau_i > t$.

Let $\sigma = \sigma_0 \sigma_1 \sigma_2 \dots$ be an infinite state sequence. A **timed state sequence** $\rho = (\sigma, \tau)$ is a pair consisting of a state sequence σ and a time sequence τ.

Timed Propositional Temporal Logic

Timed Propositional Temporal Logic (**TPTL** [AH94]) allows for **quantitative** time requirements.

- Syntax:

$$
\begin{array}{rcl}
(\textit{terms})\ \pi & \coloneqq & x + c \mid c \\
(\textit{formulae})\ \phi & \coloneqq & p \mid \pi_1 \leqslant \pi_2 \mid \pi_1 \equiv_d \pi_2 \mid \\
 & & \neg\phi_1 \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \\
 & & \mathsf{X}\phi_1 \mid \phi_1 \,\mathsf{U}\, \phi_2 \mid \phi_1 \,\mathsf{R}\, \phi_2 \mid \\
 & & x.\phi_1
\end{array}
$$

  where x is a variable, $c,d \in \mathbb{N}$ and p is a proposition letter.
- 'x.' is a **freeze quantifier** : 'x.' freezes the variable x to the time of the local temporal context.

TPTL - Semantics

Let $\mathcal{E} : \mathcal{V} \to \mathbb{N}$ be an interpretation for the variables, that we call **environment**.
We inductively define $\rho, i \models_{\mathcal{E}} \phi$, as follows:

- $\rho, i \models_{\mathcal{E}} p$ iff $p \in \sigma_i$
- $\rho, i \models_{\mathcal{E}} \pi_1 \leqslant \pi_2$ iff $\mathcal{E}(\pi_1) \leqslant \mathcal{E}(\pi_2)$
- $\rho, i \models_{\mathcal{E}} \pi_1 \equiv_d \pi_2$ iff $\mathcal{E}(\pi_1) \equiv_d \mathcal{E}(\pi_2)$
- $\rho, i \models_{\mathcal{E}} x.\phi$ iff $\rho, i \models_{\mathcal{E}[x:=\tau_i]} \phi$

The other operators are interpreted in the same way as in LTL.

TPTL - Semantics

Let $\mathcal{E} : \mathcal{V} \rightarrow \mathbb{N}$ be an interpretation for the variables, that we call **environment**.
We inductively define $\rho, i \models_{\mathcal{E}} \phi$, as follows:

- $\rho, i \models_{\mathcal{E}} p$ iff $p \in \sigma_i$
- $\rho, i \models_{\mathcal{E}} \pi_1 \leqslant \pi_2$ iff $\mathcal{E}(\pi_1) \leqslant \mathcal{E}(\pi_2)$
- $\rho, i \models_{\mathcal{E}} \pi_1 \equiv_d \pi_2$ iff $\mathcal{E}(\pi_1) \equiv_d \mathcal{E}(\pi_2)$
- $\rho, i \models_{\mathcal{E}} x.\phi$ iff $\rho, i \models_{\mathcal{E}[x:=\tau_i]} \phi$

The other operators are interpreted in the same way as in LTL.
Example (classical time bounded request-response property):

$$\phi_{BR} \quad \coloneqq \quad \mathrm{G}x.(\mathit{request} \rightarrow \mathrm{F}y.(\mathit{response} \wedge y \leqslant x + 10))$$

Tableau Systems for TPTL

- The satisfiability problem for TPTL is EXPSPACE-complete.
- Note: a variant of TPTL with past operators captures timeline-based planning problems. Satisfiability ↔ planning
- First algorithm: two-pass and graph-shaped tableau by Alur and Henziger [AH94].
- Here we describe a more recent one-pass and tree-shaped tableau for TPTL.

Tableau for TPTL - Structure

- The tableau is a tree where each node is labeled by a set of subformulae and a time $\tau \in \mathbb{N}$;

Tableau for TPTL - Structure

- The tableau is a tree where each node is labeled by a set of subformulae and a time $\tau \in \mathbb{N}$;
- The initial tableau for $z.\phi$ (in Negated Normal Form) is a tree consisting of the following single node (the root):

$$\{z.\phi\}^{TIME=0}$$

Tableau for TPTL - Structure

- The tableau is built recursively and **top-down** starting from the root, by applying a set of rules to the leaves of the tree (in this order):

Tableau for TPTL - Structure

- The tableau is built recursively and **top-down** starting from the root, by applying a set of rules to the leaves of the tree (in this order):
  1. **expansion rules**: add one or two children to a leaf of the tree;

Tableau for TPTL - Structure

- The tableau is built recursively and **top-down** starting from the root, by applying a set of rules to the leaves of the tree (in this order):
  1. **expansion rules**: add one or two children to a leaf of the tree;
  2. **termination rules**: close a branch either by *ticking* a leaf, and thus accepting the branch (✓), or by *crossing* a leaf, and thus rejecting the branch (✗);

Tableau for TPTL - Structure

- The tableau is built recursively and **top-down** starting from the root, by applying a set of rules to the leaves of the tree (in this order):
  1. **expansion rules**: add one or two children to a leaf of the tree;
  2. **termination rules**: close a branch either by *ticking* a leaf, and thus accepting the branch (✓), or by *crossing* a leaf, and thus rejecting the branch (✗);
  3. **step rule**: force an advancement in time of the model.

Tableau for TPTL - Structure

- The tableau is built recursively and **top-down** starting from the root, by applying a set of rules to the leaves of the tree (in this order):
    1. **expansion rules**: add one or two children to a leaf of the tree;
    2. **termination rules**: close a branch either by *ticking* a leaf, and thus accepting the branch (✓), or by *crossing* a leaf, and thus rejecting the branch (✗);
    3. **step rule**: force an advancement in time of the model.
- If all the branches of the tableau are closed (that is, either ticked or crossed), we say that the tableau is *complete*.
- Given a complete tableau T_ϕ, the input formula ϕ is satisfiable if and only if there is in T_ϕ at least one accepted branch.

Expansion rules

rule	ϕ	$\Gamma_1(\phi)$	$\Gamma_2(\phi)$
conjunction	$x.(\alpha \wedge \beta)$	$\{x.\alpha, x.\beta\}$	
disjunction	$x.(\alpha \vee \beta)$	$\{x.\alpha\}$	$\{x.\beta\}$
until	$x.(\alpha \mathsf{U} \beta)$	$\{x.\beta\}$	$\{x.\alpha, x.\mathsf{X}(\alpha \mathsf{U} \beta)\}$
release	$x.(\alpha \mathsf{R} \beta)$	$\{x.\alpha, x.\beta\}$	$\{x.\beta, x.\mathsf{X}(\alpha \mathsf{R} \beta)\}$
eventually	$x.\mathsf{F}\beta$	$\{x.\beta\}$	$\{x.\mathsf{XF}\beta\}$
always	$x.\mathsf{G}\alpha$	$\{x.\alpha, x.\mathsf{XG}\alpha\}$	
freeze	$x.y.\alpha$	$\{x.\alpha[y \mapsto x]\}$	

Step rule

$(\cdot)^{\delta}$ is called a **temporal shift**. For instance:

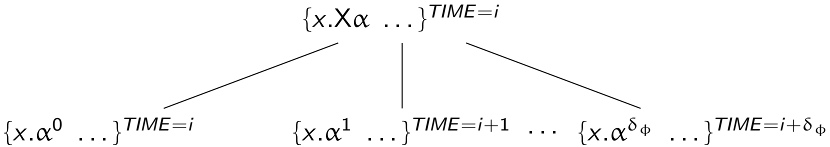
- $x.\mathsf{XG}y.(p \to y \leqslant x+1)^1 = x.\mathsf{XG}y.(p \to y \leqslant x)$
- $x.\mathsf{XG}y.(p \to y \leqslant x+1)^2 = x.\mathsf{XG}y.(p \to \bot)$

Step rule

$(\cdot)^{\delta}$ is called a **temporal shift**. For instance:

- $x.\mathsf{XG}y.(p \rightarrow y \leqslant x+1)^1 = x.\mathsf{XG}y.(p \rightarrow y \leqslant x)$
- $x.\mathsf{XG}y.(p \rightarrow y \leqslant x+1)^2 = x.\mathsf{XG}y.(p \rightarrow \bot)$

Once we reach a poised node, we can apply the **STEP** rule and advance in a state of the model.



$\{x.\mathsf{X}\alpha \ \dots\}^{TIME=i}$

$\{x.\alpha^0 \ \dots\}^{TIME=i}$ $\quad$ $\{x.\alpha^1 \ \dots\}^{TIME=i+1}$ $\quad \dots \quad$ $\{x.\alpha^{\delta_\phi} \ \dots\}^{TIME=i+\delta_\phi}$

where δ_ϕ is a value that we can pre-compute from the initial formula ϕ and that does not affect satisfiability.

Termination rules

Termination rules decide if

- the current branch has to be accepted (✓) (in this case we have found a **model**);
- the current branch has to be rejected (✗);
- or the current branch must be further explored (*i.e.*, STEP rule).

EMPTY rule and CONTRADICTION rule

EMPTY rule:

$$\{\ldots, x.p, x.q, \neg x.r, \ldots\}$$

$$|$$

$$\{\}$$

$$\checkmark$$

EMPTY rule and CONTRADICTION rule

EMPTY rule:

$$\{\ldots, x.p, x.q, \neg x.r, \ldots\}$$

$$|$$

$$\{\}$$

✓

CONTRADICTION rule:

$$\{\ldots, x.p, \neg x.p, \ldots\}$$

✗

SYNC rule

SYNC rule:

$$\{\ldots, x.(x \leqslant x+1), \ldots\}$$

✓

Remark: thanks to the expansion rule $x.y.\alpha \rightarrow \{x.\alpha[x/y]\}$ and the temporal shift, all the timing constraints that can appear in a label are of the form $x.(x \sim x + c)$, for some operator $\sim$ and some constant $c \in \mathbb{N}$.

LOOP rule

Among the models of this formula there are models featuring **infinitely many** requests, and consequently **infinitely many** responses.

$$\mathsf{G}x.(\mathit{request} \to \mathsf{F}y.(\mathit{response} \wedge y \leqslant x + 10))$$

LOOP rule

Among the models of this formula there are models featuring **infinitely many** requests, and consequently **infinitely many** responses.

$$\mathrm{G}x.(\mathit{request} \to \mathrm{F}y.(\mathit{response} \wedge y \leqslant x + 10))$$

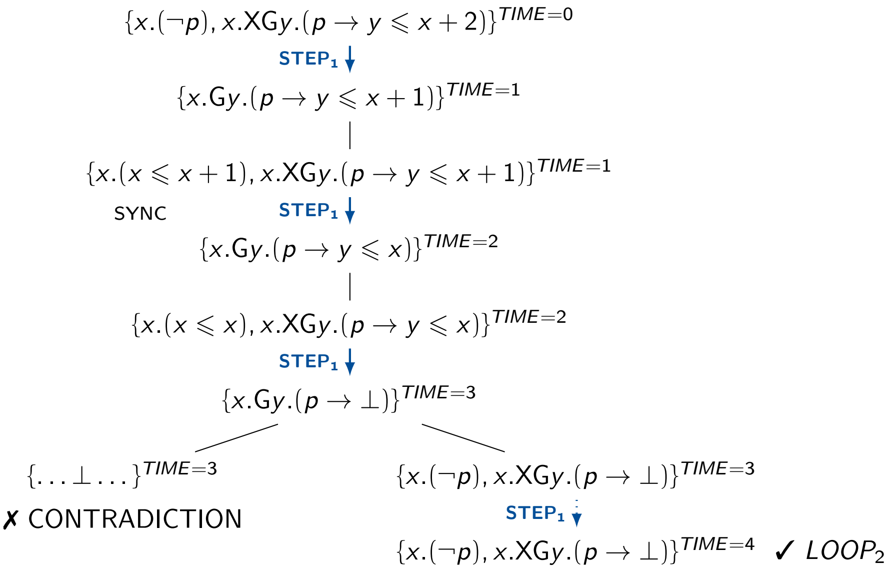
LOOP rule

Let v be a poised leaf, and let $u < v$ be a *poised node*, which is a proper ancestor of v, such that $\Gamma(u) = \Gamma(v)$ and all the eventualities (*i.e.*, $z.\mathrm{X}(\alpha\,\mathrm{U}\,\beta)$ or $z.\mathrm{XF}\beta$) requested in u are fulfilled between u and v (included). Then,

- if $\mathrm{time}(u) = \mathrm{time}(v)$, then v is crossed and the branch rejected;
- if $\mathrm{time}(u) < \mathrm{time}(v)$, then v is ticked and the branch accepted.

The PRUNE rule is the same as in the tableau for LTL.

Tableau for TPTL- Example



$\{x.(\neg p), x.\mathsf{XG}y.(p \to y \leqslant x + 2)\}^{TIME=0}$

STEP₁ ↓

$\{x.\mathsf{G}y.(p \to y \leqslant x + 1)\}^{TIME=1}$

$\{x.(x \leqslant x + 1), x.\mathsf{XG}y.(p \to y \leqslant x + 1)\}^{TIME=1}$

SYNC **STEP₁** ↓

$\{x.\mathsf{G}y.(p \to y \leqslant x)\}^{TIME=2}$

$\{x.(x \leqslant x), x.\mathsf{XG}y.(p \to y \leqslant x)\}^{TIME=2}$

STEP₁ ↓

$\{x.\mathsf{G}y.(p \to \bot)\}^{TIME=3}$

$\{\ldots \bot \ldots\}^{TIME=3}$

✗ CONTRADICTION

$\{x.(\neg p), x.\mathsf{XG}y.(p \to \bot)\}^{TIME=3}$

STEP₁ ↓

$\{x.(\neg p), x.\mathsf{XG}y.(p \to \bot)\}^{TIME=4}$ ✓ $LOOP_2$

SAT ENCODINGS AND EFFICIENT IMPLEMENTATION

Why SAT

Explicit construction of the tableau for a formula can be costly:

- existing implementations of temporal tableaux [Ber+16; AGW09] suffer on temporal formulas that require heavy **propositional reasoning**, *e.g.*:

  $$G(\phi) \wedge F(\psi)$$

  where ϕ and ψ are hard propositional formulas.
- without reinventing the wheel, why not leverage the decades of research on **SAT solvers**?
- in other words, instead of building the tableau **explicitly**, can we build it **symbolically**?

SAT encodings of Reynolds' tableau

We will see how to encode Reynolds' tableau in SAT to build and traverse it **symbolically**.

- a propositional formula **encodes the branches** of the tableau up to a certain depth k
    - one propositional model $\leftrightarrow$ one branch
    - note that the **independence** of each branch from the others is **essential** here
- the encoding is tested for satisfiability for **increasing values of** k
- a suitable encoding of the **PRUNE** rule tells us when to stop

Next Normal Form

The encoding is based on the **next normal form** (XNF) of a formula:

- a normal form where any temporal operator is nested inside a **tomorrow** operator

$$\begin{aligned}
\mathsf{xnf}(\ell) &\equiv \ell \quad \text{for } \ell \equiv p \text{ or } \ell \equiv \neg p \\
\mathsf{xnf}(\mathsf{F}\phi) &\equiv \mathsf{xnf}(\phi) \lor \mathsf{XF}\phi \\
\mathsf{xnf}(\mathsf{G}\phi) &\equiv \mathsf{xnf}(\phi) \land \mathsf{XG}\phi \\
\mathsf{xnf}(\phi \,\mathsf{U}\, \psi) &\equiv \mathsf{xnf}(\psi) \lor (\mathsf{xnf}(\phi) \land \mathsf{X}(\phi \,\mathsf{U}\, \psi)) \\
\mathsf{xnf}(\phi \,\mathsf{R}\, \psi) &\equiv \mathsf{xnf}(\psi) \land (\mathsf{xnf}(\phi) \lor \mathsf{X}(\phi \,\mathsf{R}\, \psi))
\end{aligned}$$

Note that $\mathsf{xnf}(\psi)$ encodes the **expansion rule** of ψ.

Stepping and grounding

Fix a depth k.

- the truth of each proposition p at each time step $0 \leqslant i \leqslant k$ is represented by a **stepped** proposition p^i
- **tomorrow** formulas $\mathsf{X}\alpha$ are **grounded** to propositions called $(\mathsf{X}\alpha)^i_{\mathsf{G}}$
- the stepped and grounded formula $\mathrm{xnf}(\psi)^i_G$ is a **propositional** formula

Unraveling

The branches **up to depth k** are encoded by a formula $[\![\phi]\!]^k$ called the **k-unraveling** of ϕ.

$$[\![\phi]\!]^0 \equiv \mathsf{xnf}(\phi)^0_G$$
$$[\![\phi]\!]^{k+1} \equiv [\![\phi]\!]^k \wedge \bigwedge_{\mathsf{X}\alpha \in \mathsf{C}(\phi)} \left((\mathsf{X}\alpha)^k_G \leftrightarrow \mathsf{xnf}(\alpha)^{k+1}_G \right)$$

Theorem

$[\![\phi]\!]^k$ is satisfiable iff the tableau for ϕ contains a branch with at least k poised nodes.

Unraveling

The branches **up to depth** k are encoded by a formula $[\![\phi]\!]^k$ called the **k-unraveling** of ϕ.

$$
\begin{aligned}
[\![\phi]\!]^0 &\equiv \mathsf{xnf}(\phi)^0_G \\
[\![\phi]\!]^{k+1} &\equiv [\![\phi]\!]^k \wedge \bigwedge_{\mathsf{X}\alpha \in \mathsf{C}(\phi)} \left((\mathsf{X}\alpha)^k_G \leftrightarrow \mathsf{xnf}(\alpha)^{k+1}_G \right)
\end{aligned}
$$

Note:

- here we are encoding the **expansion rules** and the **STEP** rule
- implicitly, also the **CONTRADICTION** rule, since only correct models are considered

Accepting branches

Now that we have encoded the branches, we have to select the **accepted** ones.

- we need to encode the **EMPTY** rule:

$$\mathsf{E}_k \equiv \bigwedge_{\mathsf{X}\alpha \in \mathsf{C}(\phi)} \neg(\mathsf{X}\alpha)^k_G$$

Accepting branches

Now that we have encoded the branches, we have to select the **accepted** ones.

- and the **LOOP** rule:

${}_{\ell}R_k \equiv \bigwedge_{\mathsf{X}\alpha \in C(\phi)} \left((\mathsf{X}\alpha)_G^{\ell} \leftrightarrow (\mathsf{X}\alpha)_G^{k} \right)$	nodes at position l and k have the same **X-requests**
${}_{\ell}F_k \equiv \bigwedge_{\psi \equiv \mathsf{X}(\psi_1 \mathsf{U} \psi_2)} \left(\psi_G^{k} \to \bigvee_{i=\ell+1}^{k} \mathsf{xnf}(\psi_2)_G^{i} \right)$	each requested **X-eventuality** is fulfilled between ℓ and k
$L_k \equiv \bigvee_{l=0}^{k-1} ({}_{\ell}R_k \wedge {}_{\ell}F_k)$	this happens for at least one ℓ

Accepting branches

Theorem

If the formula $[\![\phi]\!]^k \wedge (L_k \vee E_k)$ *is* ***satisfiable****, then the tableau for* ϕ *contains an* ***accepted*** *branch of at most* $k+1$ *poised nodes.*

Encoding the PRUNE rule

The encoding of the **PRUNE** rule follows a similar pattern.

- all the X-**eventualities** requested at position k and fulfilled between j and k are also fulfilled between ℓ and j:

$$ {}_{\ell}P_j^k \equiv \bigwedge_{\psi \equiv \mathsf{X}(\psi_1 \mathsf{U} \psi_2)} \left((\psi_G^k \wedge \bigvee_{i=j+1}^{k} \mathrm{xnf}(\psi_2)_G^i) \to \bigvee_{i=\ell+1}^{j} \mathrm{xnf}(\psi_2)_G^i \right) $$

- this happens for at least some ℓ and j such that ℓ, j and k share the same **X-requests**:

$$ P^k \equiv \bigvee_{\ell=0}^{k-2} \bigvee_{j=\ell+1}^{k-1} ({}_{\ell}R_j \wedge {}_{j}R_k \wedge {}_{\ell}P_j^k) $$

Encoding the PRUNE rule

Theorem

If $[\![\phi]\!]^k \wedge \bigwedge_{i=0}^{k} \neg P^i$ is ***unsatisfiable****, then the tableau for ϕ has only* ***rejected*** *branches.*

Putting everything together

The complete algorithm is shown here.

- each iteration of the loop **increases** k
- note that the calls to the SAT solver at lines 7 and 10 are **incremental**:
    - in each case a conjunct is added to $[\![\phi]\!]^k$
    - modern SAT solvers can **reuse** the work done in the previous call
- the check at **line 4** is **optional** but speeds up some cases
- the check at **line 10** can be **removed** if termination for unsatisfiable instances is not needed, and can **speed up** computation for satisfiable ones.

```
 1: procedure BLACK($\phi$)
 2:     $k \leftarrow 0$
 3:     while True do
 4:         if $[\![\phi]\!]^k$ is unsat. then
 5:             return UNSAT
 6:         end if
 7:         if $[\![\phi]\!]^k \wedge (L_k \vee E_k)$ is sat. then
 8:             return SAT
 9:         end if
10:         if $[\![\phi]\!]^k \wedge \bigwedge_{i=0}^{k} \neg P^i$ is unsat. then
11:             return UNSAT
12:         end if
13:         $k \leftarrow k + 1$
14:     end while
15: end procedure
```

BLACK, the Bounded LTL sAtisfiability ChecKer

We implemented the SAT-based algorithm shown before in a software tool called **BLACK**.

Let us talk a bit about it.

BLACK

BLACK is not just a prototype:

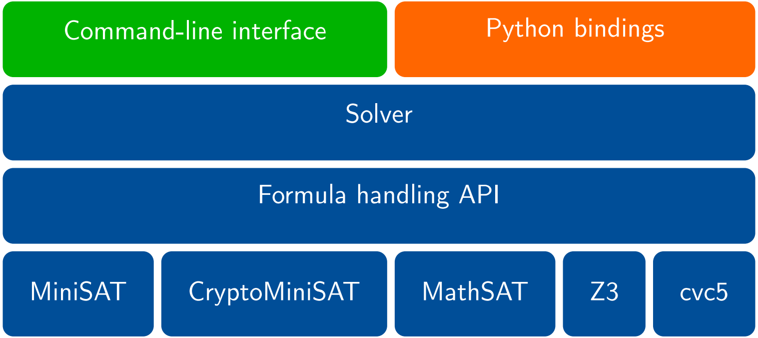
- **multiplatform** (Linux, macOS, Windows)
- **easy** to install and to use
  - binary **packages** provided for all the supported platforms
  - ergonomic command-line **interface**
  - comprehensive **documentation** (`https://www.black-sat.org`)
- designed as a **library**
  - well-defined C++20 **API** that can be **embedded** into any client application
  - Python bindings for easy prototyping (under development)

BLACK

BLACK is not just a prototype:

- flexible
    - multiple **backends** (MiniSAT, CryptoMiniSAT, Z3, MathSAT, cvc5)
    - multiple **logics** (LTL, LTLf, LTL with past, LTLf modulo theories)
- robust and **trustable**
    - comprehensive test suite with over 4000 test formulas and 100% code coverage
- **open source** (MIT license)
    - `https://github.com/black-sat/black`

Architecture



Command-line interface

Python bindings

Solver

Formula handling API

MiniSAT

CryptoMiniSAT

MathSAT

Z3

cvc5

Performance

BLACK's performance are competitive with other state-of-the-art tools

- nuXmv [Cav+14], Aalta [Li+14], *etc.*
- incremental SAT-based algorithm works better on formulas with **short** models
- suffers on formulas with **very long** models
- generally faster when **past operators** are involved [Gea+21b]
- the most robust when handling formulas with a huge number of **variables**
- see the papers for the plots [GGM19; Gea+21b]

DEMO

CONCLUSIONS

Wrap up

We made an overview of **tableau methods** for linear-time temporal logics.

- classical **graph-shaped** tableaux
- recent **tree-shaped** methods
- in-depth overview of **Reynolds'** tableau
- Reynolds-style tableau for the real-time **TPTL** logic
- efficient SAT encodings and the **BLACK** tool

Open directions

Many research directions are open:

- SAT or SMT encoding of the **TPTL** tree-shaped tableau
- **first-order** extensions (see our IJCAI talk)
- more efficient encodings:
    - the **LOOP** and **PRUNE** rule have an $\mathcal{O}(n^3)$ encoding. Can we do better?
- can we extract **unsat certificates** from the SAT encoding?
- the explicit implementations can be very easily **parallelised** [MR17]
    - can we parallelize the SAT-based algorithm somehow?

THANK YOU

REFERENCES

References

[AGW09] P. Abate, R. Goré, and F. Widmann. "An On-the-fly Tableau-based Decision Procedure for PDL-satisfiability." In: *Electronic Notes in Theoretical Computer Science* 231 (2009), pp. 191–209. doi: `10.1016/j.entcs.2009.02.036`.

[AH93] R. Alur and T. A. Henzinger. "Real-Time Logics: Complexity and Expressiveness." In: *Information and Computation* 104.1 (1993), pp. 35–77. doi: `10.1006/inco.1993.1025`.

[AH94] R. Alur and T. A. Henzinger. "A Really Temporal Logic." In: *Journal of the ACM* 41.1 (1994), pp. 181–204.

[Art+14] Alessandro Artale, Roman Kontchakov, Vladislav Ryzhikov, and Michael Zakharyaschev. "A Cookbook for Temporal Conceptual Data Modelling with Description Logics." In: *ACM Trans. Comput. Log.* 15.3 (2014), 25:1–25:50. doi: `10.1145/2629565`.

[BD19] Ronen I. Brafman and Giuseppe De Giacomo. "Planning for LTLf /LDLf Goals in Non-Markovian Fully Observable Nondeterministic Domains." In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. Ed. by Sarit Kraus. ijcai.org, 2019, pp. 1602–1608. doi: `10.24963/ijcai.2019/222`.

References

[BDP18] Ronen I. Brafman, Giuseppe De Giacomo, and Fabio Patrizi. "LTLf/LDLf Non-Markovian Rewards." In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 1771–1778.

[Ben+09] Johan van Benthem, Jelle Gerbrandy, Tomohiro Hoshi, and Eric Pacuit. "Merging Frameworks for Interaction." In: *J. Philos. Log.* 38.5 (2009), pp. 491–526. doi: `10.1007/s10992-008-9099-x`.

[Ber+16] M. Bertello, N. Gigante, A. Montanari, and M. Reynolds. "Leviathan: A New LTL Satisfiability Checking Tool Based on a One-Pass Tree-Shaped Tableau." In: *Proc. of the 25th International Joint Conference on Artificial Intelligence*. IJCAI/AAAI Press, 2016, pp. 950–956.

[Bet59] E. W. Beth. "Semantic Entailment and Formal Derivability." In: *Sapientia* 14.54 (1959), p. 311.

[BK98] Fahiem Bacchus and Froduald Kabanza. "Planning for Temporally Extended Goals." In: *Annals of Mathematics in Artificial Intelligence* 22.1-2 (1998), pp. 5–27.

References

[BSS19] Andrea Brunello, Guido Sciavicco, and Ionel Eduard Stan. "Interval temporal logic decision tree learning." In: *European Conference on Logics in Artificial Intelligence*. Springer. 2019, pp. 778–793.

[Cav+14] Roberto Cavada, Alessandro Cimatti, Michele Dorigatti, Alberto Griggio, Alessandro Mariotti, Andrea Micheli, Sergio Mover, Marco Roveri, and Stefano Tonetta. "The nuXmv symbolic model checker." In: *Proc. of the 26th International Conference on Computer Aided Verification*. Springer. 2014, pp. 334–342.

[DAg+99] Marcello D'Agostino, Dov M. Gabbay, Reiner Hähnle, and Joachim Posegga, eds. *Handbook of Tableau Methods*. Springer, 1999. isbn: 978-0-7923-5627-1.

[De +20] Giuseppe De Giacomo, Marco Favorito, Luca Iocchi, and Fabio Patrizi. "Imitation Learning over Heterogeneous Agents with Restraining Bolts." In: *Proceedings of the 13th International Conference on Automated Planning and Scheduling*. AAAI Press, 2020, pp. 517–521.

References

[Del+17] D. Della Monica, N. Gigante, A. Montanari, P. Sala, and G. Sciavicco. "Bounded Timed Propositional Temporal Logic with Past Captures Timeline-based Planning with Bounded Constraints." In: *Proc. of the 26th International Joint Conference on Artificial Intelligence*. 2017, pp. 1008–1014. doi: `10.24963/ijcai.2017/140`.

[FL03] Maria Fox and Derek Long. "PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains." In: *J. Artif. Intell. Res.* 20 (2003), pp. 61–124. doi: `10.1613/jair.1129`.

[Gea+21a] Luca Geatti, Nicola Gigante, Angelo Montanari, and Mark Reynolds. "One-pass and tree-shaped tableau systems for TPTL and TPTL_b+Past." In: *Inf. Comput.* 278 (2021), p. 104599. doi: `10.1016/j.ic.2020.104599`. url: `https://doi.org/10.1016/j.ic.2020.104599`.

[Gea+21b] Luca Geatti, Nicola Gigante, Angelo Montanari, and Gabriele Venturato. "Past Matters: Supporting LTL+Past in the BLACK Satisfiability Checker." In: *Proceedings of the 28th International Symposium on Temporal Representation and Reasoning*. 2021.

References

[GGM19] Luca Geatti, Nicola Gigante, and Angelo Montanari. “A SAT-Based Encoding of the One-Pass and Tree-Shaped Tableau System for LTL.” In: *Proceedings of the 28th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*. Ed. by Serenella Cerrito and Andrei Popescu. Vol. 11714. Lecture Notes in Computer Science. Springer, 2019, pp. 3–20. doi: `10.1007/978-3-030-29026-9\_1`.

[GMR17] N. Gigante, A. Montanari, and M. Reynolds. “A One-Pass Tree-Shaped Tableau for LTL+Past.” In: *Proc. of 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning*. Vol. 46. EPiC Series in Computing. 2017, pp. 456–473.

[Ham+21] Lewis Hammond, Alessandro Abate, Julian Gutierrez, and Michael J. Wooldridge. “Multi-Agent Reinforcement Learning with Temporal Logic Specifications.” In: *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems*. ACM, 2021, pp. 583–592. doi: `10.5555/3463952.3464024`.

References

[HK03] Ullrich Hustadt and Boris Konev. "TRP++2.0: A Temporal Resolution Prover." In: *Proc. of the 19th International Conference on Automated Deduction*. Vol. 2741. LNCS. Springer, 2003, pp. 274–278. doi: `10.1007/978-3-540-45085-6_21`.

[Jac+17] Swen Jacobs, Roderick Bloem, Romain Brenguier, Rüdiger Ehlers, Timotheus Hell, Robert Könighofer, Guillermo A. Pérez, Jean-François Raskin, Leonid Ryzhyk, Ocan Sankur, Martina Seidl, Leander Tentrup, and Adam Walker. "The first reactive synthesis competition (SYNTCOMP 2014)." In: *Int. J. Softw. Tools Technol. Transf.* 19.3 (2017), pp. 367–390. doi: `10.1007/s10009-016-0416-3`.

[Kes+93] Y. Kesten, Z. Manna, H. McGuire, and A. Pnueli. "A Decision Algorithm for Full Propositional Temporal Logic." In: *Proc. of the 5th International Conference on Computer Aided Verification*. Vol. 697. LNCS. Springer, 1993, pp. 97–109. doi: `10.1007/3-540-56922-7\_9`.

References

[Li+14] Jianwen Li, Yinbo Yao, Geguang Pu, Lijun Zhang, and Jifeng He. "Aalta: an LTL satisfiability checker over Infinite/Finite traces." In: *Proc. of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*. Ed. by Shing-Chi Cheung, Alessandro Orso, and Margaret-Anne D. Storey. ACM, 2014, pp. 731–734. doi: `10.1145/2635868.2661669`.

[LP00] O. Lichtenstein and A. Pnueli. "Propositional Temporal Logics: Decidability and Completeness." In: *Logic Journal of the IGPL* 8.1 (2000), pp. 55–85. doi: `10.1093/jigpal/8.1.55`.

[Mar03] N. Markey. "Temporal Logic with Past is Exponentially More Succinct." In: *Bulletin of the EATCS* 79 (2003), pp. 122–128.

[May+07] Marta Cialdea Mayer, Carla Limongelli, Andrea Orlandini, and Valentina Poggioni. "Linear temporal logic as an executable semantics for planning languages." In: *Journal of Logic, Language and Information* 16.1 (2007), pp. 63–89. doi: `10.1007/s10849-006-9022-1`.

References

[MR17] J. Christopher McCabe-Dansted and M. Reynolds. "A Parallel Linear Temporal Logic Tableau." In: *Proc. of the 8th International Symposium on Games, Automata, Logics and Formal Verification*. Ed. by P. Bouyer, A. Orlandini, and P. San Pietro. Vol. 256. EPTCS. 2017, pp. 166–179.

[Pnu77] A. Pnueli. "The Temporal Logic of Programs." In: *Proc. of the 18th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1977, pp. 46–57. doi: `10.1109/SFCS.1977.32`.

[Pri57] A.N. Prior. *Time and Modality*. Clarendon Press, 1957.

[Rey09] Mark Reynolds. "A Tableau for CTL*." In: *Proceedings of the Second World Congress on Formal Methods*. Vol. 5850. Lecture Notes in Computer Science. Springer, 2009, pp. 403–418. doi: `10.1007/978-3-642-05089-3\_26`.

[Rey14] Mark Reynolds. "A Tableau for Temporal Logic over the Reals." In: *Advances in Modal Logic*. 2014, pp. 439–458.

References

[Rey16] M. Reynolds. “A New Rule for LTL Tableaux.” In: *Proc. of the 7th International Symposium on Games, Automata, Logics and Formal Verification*. Vol. 226. EPTCS. 2016, pp. 287–301. doi: `10.4204/EPTCS.226.20`.

[SC85] A. P. Sistla and E. M. Clarke. “The Complexity of Propositional Linear Temporal Logics.” In: *Journal of ACM* 32.3 (1985), pp. 733–749. doi: `10.1145/3828.3837`.

[Sch98] S. Schwendimann. “A New One-Pass Tableau Calculus for PLTL.” In: *Proc. of the 7th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods*. Vol. 1397. LNCS. Springer, 1998, pp. 277–292. doi: `10.1007/3-540-69778-0\_28`.

[Smu95] R.M. Smullyan. *First-order Logic*. Dover books on advanced mathematics. Dover, 1995. isbn: 9780486683706. url: `https://books.google.it/books?id=kgvhQ-oSZiUC`.

[Wol83] Pierre Wolper. “Temporal Logic Can Be More Expressive.” In: *Information and Control* 56.1/2 (1983), pp. 72–99. doi: `10.1016/S0019-9958(83)80051-5`.

References

[Wol85] P. Wolper. "The Tableau Method for Temporal Logic: An Overview." In: *Logique et Analyse* 28 (1985), pp. 119–136.