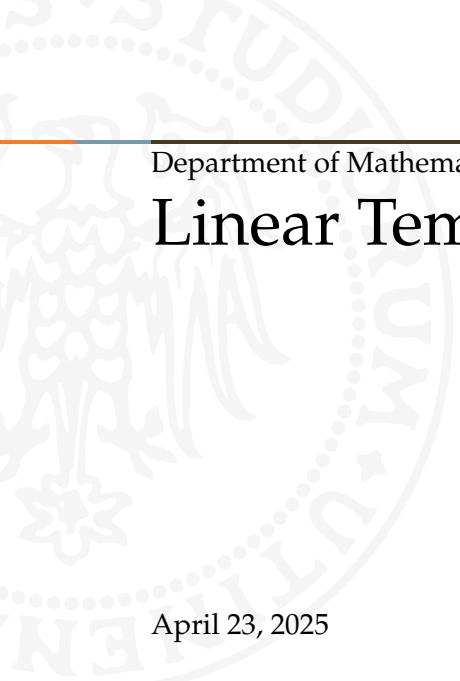




Department of Mathematics, Computer Science and Physics, University of Udine

Linear Temporal Logic

Luca Geatti

luca.geatti@uniud.it

Angelo Montanari

angelo.montanari@uniud.it

April 23, 2025



Temporal logics are the de-facto standard languages for specifying properties of systems in *formal verification* and *artificial intelligence*.

- born in the '50s as a tool for philosophical argumentation about time

Reference:

Arthur N. Prior (1957). *Time and Modality*. London: Oxford University Press

- the idea of its use in formal verification can be traced back to the '70s

Reference:

Amir Pnueli (1977). "The temporal logic of programs". In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE, pp. 46–57.
DOI: 10.1109/SFCS.1977.32



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

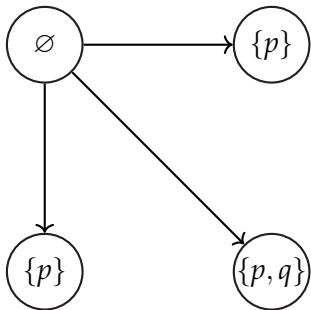
- automated planning
 - temporally extended goals (Bacchus and Kabanza 1998)
 - temporal planning (Fox and Long 2003)
 - timeline-based planning (Della Monica et al. 2017)
- automated synthesis (Jacobs et al. 2017)
- autonomy under uncertainty (Brafman and De Giacomo 2019)
 - specification of goals for planning over MDPs and POMDPs
- reinforcement learning (De Giacomo et al. 2020; Hammond et al. 2021)
 - specification of reward functions and safety conditions
- knowledge representation
 - temporal description logics (Artale et al. 2014)
- multi-agent systems
 - temporal epistemic logics (van Benthem et al. 2009)



Modal Logic extends classic propositional (Boolean) logic with the concepts of *necessity* and *possibility*.

- **World** = set of propositions that are supposed to be true in that world
- Worlds are connected with edges
 - directed graph with labels on the nodes: **Kripke structure**
- in Modal Logic, the truth of a formula depends on the *world* in which is interpreted (many-worlds interpretation) and on the worlds accessible from it.
- Necessity ($\Box$): is asking something to be true in *all* accessible states
- Possibility ($\Diamond$): is asking something to be true in *at least one* accessible state

- Necessity ($\Box$): is asking something to be true in *all* accessible states
- Possibility ($\Diamond$): is asking something to be true in *at least one* accessible state

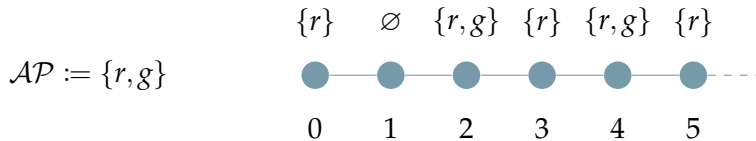


- $\mathcal{AP} = \{p, q\}$
- $\Box p$ is true
- $\Diamond q$ is true
- $\Box q$ is false
- $\Box p \vee \Box q$ is true



Linear Temporal Logic (**LTL**, for short) is a (special case of) Modal Logic.

- World = **State** = set of proposition letters that are supposed to hold (*i.e.*, to be true) in that state
- Kripke Structure = (infinite) **linear order** of states = state sequence = word in a language
 - accessibility relation = temporal ordering
- Necessity ($\Box$) = Always in the future (G)
- Possibility ($\Diamond$) = Sometimes in the future (F)





- introduced by Pnueli in the '70s
- interpreted over *state sequences*
- it extends classical *propositional* logic
- temporal *operators* are used to talk about how propositions change over time

Reference:

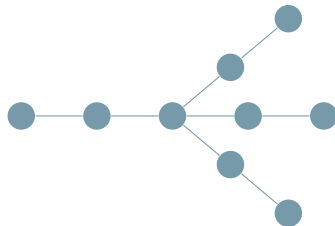
Amir Pnueli (1977). “The temporal logic of programs”. In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE, pp. 46–57.
DOI: 10.1109/SFCS.1977.32

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

Linear



Branching



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

Infinite



Finite

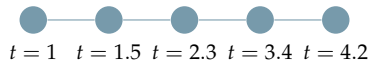


There are many choices to be made for the representation of *time*.

Qualitative



Real-time



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

Discrete



Dense



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

Points



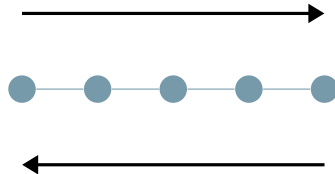
Intervals



Pure Future



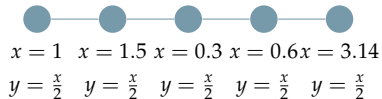
Past-Future



Propositional



First-Order





We focus here on:

- *linear-time*
- *discrete-time*
- *qualitative-time*
- *infinite-time*
- *future only*
- *propositional*



Let $\mathcal{AP} := \{p, q, r, \dots\}$ be a set of *atomic propositions*. The syntax of **LTL** is defined as follows:

$$\phi := p \mid \neg\phi \mid \phi \vee \phi$$

Boolean Modalities with $p \in \mathcal{AP}$

$$\mid X\phi \mid \phi \text{ U } \phi$$

Future Temporal Modalities

- $X\phi$ is the **Next** operator: *at the next time point (tomorrow), the formula ϕ holds*
- $\phi_1 \text{ U } \phi_2$ is the **Until** operator: *there exists a time point in the future where ϕ_2 is true, and ϕ_1 holds from now until (and excluding) that point.*

Shortcuts:

- **Eventually**, $F\phi$: *there exists a time point in the future where ϕ holds.* It is defined as $F\phi \equiv \top \text{ U } \phi$.
- **Globally**, $G\phi$: *for all time points in the future ϕ holds.* It is defined as $G\phi \equiv \neg(F\neg\phi)$.



Example:

Consider $\mathcal{AP} = \{p, q\}$ and the following formula:

$$GF(p)$$

Which state sequences are *models* of the formula?

- $\{p\} \cdot \{q\} \cdot \{p\} \cdot (\{q\})^\omega$
- $(\{p, q\})^\omega$
- $(\{q\} \cdot \{q\} \cdot \{p\} \cdot \{q\})^\omega$
- $(\{p\})^* \cdot (\{q\})^\omega$



Example:

Consider $\mathcal{AP} = \{p, q\}$ and the following formula:

$$GF(p)$$

Which state sequences are *models* of the formula?

- $\{p\} \cdot \{q\} \cdot \{p\} \cdot (\{q\})^\omega$ no
- $(\{p, q\})^\omega$ yes
- $(\{q\} \cdot \{q\} \cdot \{p\} \cdot \{q\})^\omega$ yes
- $(\{p\})^* \cdot (\{q\})^\omega$ no



Example:

Consider $\mathcal{AP} = \{p, q\}$ and the following formula:

$$\text{FG}(q)$$

Which state sequences are *models* of the formula?

- $\{p\} \cdot \{q\} \cdot \{p\} \cdot (\{q\})^\omega$
- $(\{p, q\})^\omega$
- $(\{q\} \cdot \{q\} \cdot \{p\} \cdot \{q\})^\omega$
- $(\{p\})^* \cdot (\{q\})^\omega$



Example:

Consider $\mathcal{AP} = \{p, q\}$ and the following formula:

$$\text{FG}(q)$$

Which state sequences are *models* of the formula?

- $\{p\} \cdot \{q\} \cdot \{p\} \cdot (\{q\})^\omega$ yes
- $(\{p, q\})^\omega$ yes
- $(\{q\} \cdot \{q\} \cdot \{p\} \cdot \{q\})^\omega$ no
- $(\{p\})^* \cdot (\{q\})^\omega$ yes



Example:

Let $\mathcal{AP} = \{r, g\}$. Each request (r) is eventually followed by a grant (g).

$$G(r \rightarrow F(g))$$

Which state sequences are *models* of the formula?

- $(\emptyset)^\omega$
- $\{r\} \cdot \{r\} \cdot \{r\} \cdot (\emptyset)^\omega$
- $\{r\} \cdot \{r\} \cdot \{r\} \cdot \{g\} \cdot (\emptyset)^\omega$
- $(\{r\} \cdot \emptyset \cdot \emptyset \cdot \{g\})^\omega$



Example:

Let $\mathcal{AP} = \{r, g\}$. Each request (r) is eventually followed by a grant (g).

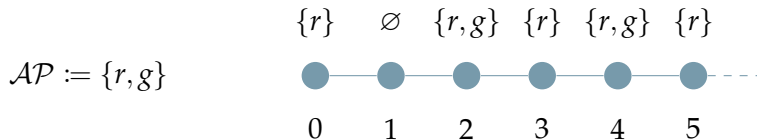
$$G(r \rightarrow F(g))$$

Which state sequences are *models* of the formula?

- $(\emptyset)^\omega$ yes
- $\{r\} \cdot \{r\} \cdot \{r\} \cdot (\emptyset)^\omega$ no
- $\{r\} \cdot \{r\} \cdot \{r\} \cdot \{g\} \cdot (\emptyset)^\omega$ yes
- $(\{r\} \cdot \emptyset \cdot \emptyset \cdot \{g\})^\omega$ yes

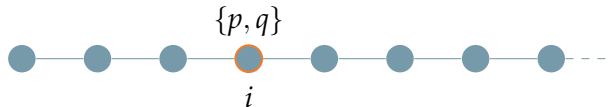


- Given a set of atomic propositions $\mathcal{AP}$, any LTL formula defined over $\mathcal{AP}$ is interpreted over *infinite words* $\sigma \in (2^{\mathcal{AP}})^\omega$.
- Let $\sigma = \langle \sigma_0, \sigma_1, \dots \rangle$. For each $i \geq 0$, $\sigma_i \subseteq \mathcal{AP}$ is called a **state** contains the atomic propositions that are supposed to hold in that state.
- In this context, sequences in $(2^{\mathcal{AP}})^\omega$ are also called **state sequences** or **traces**.



We say that σ satisfies at position i the LTL formula ϕ , written $\sigma, i \models \phi$, iff:

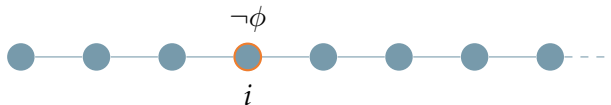
- $\sigma, i \models p$ iff $p \in \sigma_i$



p holds at position i

We say that σ satisfies at position i the LTL formula ϕ , written $\sigma, i \models \phi$, iff:

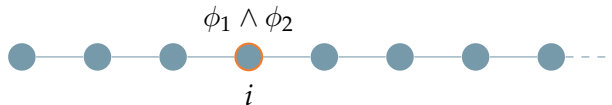
- $\sigma, i \models \neg\phi$ iff $\sigma, i \not\models \phi$



ϕ does not hold at position i

We say that σ satisfies at position i the LTL formula ϕ , written $\sigma, i \models \phi$, iff:

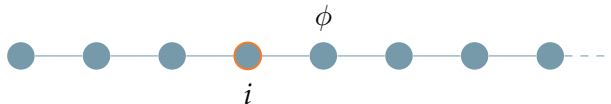
- $\sigma, i \models \phi_1 \wedge \phi_2$ iff $\sigma, i \models \phi_1$ and $\sigma, i \models \phi_2$



ϕ_1 and ϕ_2 hold at position i

We say that σ satisfies at position i the LTL formula ϕ , written $\sigma, i \models \phi$, iff:

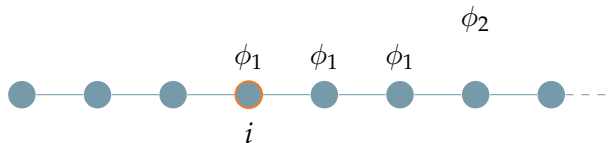
- $\sigma, i \models X\phi$ iff $\sigma, i + 1 \models \phi$



ϕ holds at the *next* position of i

We say that σ satisfies at position i the LTL formula ϕ , written $\sigma, i \models \phi$, iff:

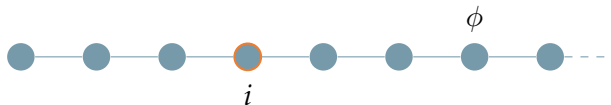
- $\sigma, i \models \phi_1 \cup \phi_2$ iff $\exists j \geq i . \sigma, j \models \phi_2$ and $\forall i \leq k < j . \sigma, k \models \phi_1$



ϕ_1 holds *until* ϕ_2 holds

Shortcuts:

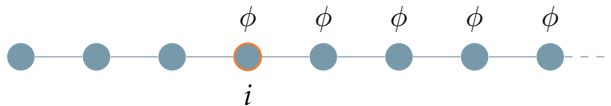
- (eventually) $F\phi \equiv \top U \phi$



ϕ will *eventually* hold

Shortcuts:

- (globally) $G\phi \equiv \neg F\neg\phi$



ϕ holds *always*



- We say that σ *satisfies* ϕ (written $\sigma \models \phi$) iff $\sigma, 0 \models \phi$. In this case, we say that σ is a *model* of ϕ .
- For any LTL formula ϕ , we define *the language of ϕ* as:

$$\mathcal{L}(\phi) = \{\sigma \in (2^{\mathcal{AP}})^\omega \mid \sigma \models \phi\}$$

- We say that ϕ is *satisfiable* iff $\mathcal{L}(\phi) \neq \emptyset$.
- We say that ϕ is *valid* iff $\mathcal{L}(\phi) = (2^{\mathcal{AP}})^\omega$.



Example:

Consider $\mathcal{AP} = \{p, q\}$ and the following formula:

$$F(p \wedge Xq)$$

Which state sequences are *models* of the formula?

- $(\emptyset)^\omega$
- $(\{q\})^\omega$
- $(\emptyset)^* \cdot \{p\} \cdot \emptyset \cdot \{q\} \cdot (\emptyset)^\omega$
- $(\emptyset)^* \cdot \{p\} \cdot \{q\} \cdot (\emptyset)^\omega$



Example:

Consider $\mathcal{AP} = \{p, q\}$ and the following formula:

$$F(p \wedge Xq)$$

Which state sequences are *models* of the formula?

- $(\emptyset)^\omega$ no
- $(\{q\})^\omega$ no
- $(\emptyset)^* \cdot \{p\} \cdot \emptyset \cdot \{q\} \cdot (\emptyset)^\omega$ no
- $(\emptyset)^* \cdot \{p\} \cdot \{q\} \cdot (\emptyset)^\omega$ yes



Example:

Consider $\mathcal{AP} = \{p, q\}$ and the following formulas:

$$F(p) \wedge F(q) \qquad F(p \wedge Fq) \qquad F(p \wedge q)$$

Which state sequences are *models* of the formula?

- $(\emptyset)^\omega$
- $(\emptyset)^* \cdot \{p\} \cdot \emptyset \cdot \{q\} \cdot (\emptyset)^\omega$
- $(\emptyset)^* \cdot \{q\} \cdot \emptyset \cdot \{p\} \cdot (\emptyset)^\omega$
- $(\emptyset)^* \cdot \{p, q\} \cdot (\emptyset)^\omega$



Example:

Consider $\mathcal{AP} = \{p, q\}$ and the following formulas:

$$F(p) \wedge F(q) \qquad F(p \wedge Fq) \qquad F(p \wedge q)$$

Which state sequences are *models* of the formula?

• $(\emptyset)^\omega$	no	no	no
• $(\emptyset)^* \cdot \{p\} \cdot \emptyset \cdot \{q\} \cdot (\emptyset)^\omega$	yes	yes	no
• $(\emptyset)^* \cdot \{q\} \cdot \emptyset \cdot \{p\} \cdot (\emptyset)^\omega$	yes	no	no
• $(\emptyset)^* \cdot \{p, q\} \cdot (\emptyset)^\omega$	yes	yes	yes



Example:

Consider $\mathcal{AP} = \{p, q\}$. What is the language of the following formula?

$$p \cup (Gq)$$

Write an equivalent ω -regular expression.



Example:

Consider $\mathcal{AP} = \{p, q\}$. What is the language of the following formula?

$$p \cup (\text{G}q)$$

Write an equivalent ω -regular expression.

$$\mathcal{L}(p \cup (\text{G}q)) = (\{p\} \cup \{p, q\})^* \cdot (\{q\} \cup \{p, q\})^\omega$$



Example:

Consider $\mathcal{AP} = \{p, q\}$.

Is the formula $\text{FX}p$ equivalent to $\text{XF}p$?



Example:

Consider $\mathcal{AP} = \{p, q\}$.

Is the formula FXp equivalent to XFp ?

Yes.



Example:

Consider $\mathcal{AP} = \{p, q\}$.

Is the formula FXp equivalent to XFp ?

Yes.

Exercise:

Consider $\mathcal{AP} = \{p, q\}$.

Write the formula $(Gp) \cup q$ without using the *Until* operator, that is, using only F , G , and Boolean modalities.



Compassion:

Consider $\mathcal{AP} = \{en, tk\}$.

It is not possible that a transition is enabled infinitely many times but taken only finitely many times.



Compassion:

Consider $\mathcal{AP} = \{en, tk\}$.

It is not possible that a transition is enabled infinitely many times but taken only finitely many times.

$$\neg(\text{GF}(en) \wedge \text{FG}(\neg tk)) \quad \equiv \quad \text{GF}(en) \rightarrow \text{GF}(tk)$$

This is very different from $\text{GF}(en \rightarrow tk)$ and from $\text{G}(en \rightarrow \text{F}(tk))$.



Compassion:

Consider $\mathcal{AP} = \{en, tk\}$.

It is not possible that a transition is enabled infinitely many times but taken only finitely many times.

$$\neg(\text{GF}(en) \wedge \text{FG}(\neg tk)) \quad \equiv \quad \text{GF}(en) \rightarrow \text{GF}(tk)$$

This is very different from $\text{GF}(en \rightarrow tk)$ and from $\text{G}(en \rightarrow \text{F}(tk))$.

Justice:

Consider $\mathcal{AP} = \{en, tk\}$.

It is never the case that a transition is always enabled but never taken.



Compassion:

Consider $\mathcal{AP} = \{en, tk\}$.

It is not possible that a transition is enabled infinitely many times but taken only finitely many times.

$$\neg(\text{GF}(en) \wedge \text{FG}(\neg tk)) \quad \equiv \quad \text{GF}(en) \rightarrow \text{GF}(tk)$$

This is very different from $\text{GF}(en \rightarrow tk)$ and from $\text{G}(en \rightarrow \text{F}(tk))$.

Justice:

Consider $\mathcal{AP} = \{en, tk\}$.

It is never the case that a transition is always enabled but never taken.

$$\neg \text{F}(\text{G}(en) \wedge \text{G}(\neg tk))$$

This is equivalent to $\text{G}(\text{G}(en) \rightarrow \text{F}(tk))$.



It is possible to define a *strict* version of the until as follows:

- $\sigma, i \models \phi_1 \text{ U}^s \phi_2$ iff $\exists j > i . \sigma, j \models \phi_2$ and $\forall i < k < j . \sigma, k \models \phi_1$

How can be encode formulas of type $X\phi$ with only the strict version of the until?

Therefore, if we adopt the *strict* version, then it is possible to define LTL with the only temporal operator being the *until*.

- ... but encoding the standard until with the strict until requires more space:

$$\phi_1 \text{ U } \phi_2 \equiv \phi_2 \vee (\phi_1 \wedge \phi_1 \text{ U}^s \phi_2)$$



It is possible to define a *strict* version of the until as follows:

- $\sigma, i \models \phi_1 \text{ U}^s \phi_2$ iff $\exists j > i . \sigma, j \models \phi_2$ and $\forall i < k < j . \sigma, k \models \phi_1$

How can we encode formulas of type $X\phi$ with only the strict version of the until?

$$X\phi \equiv \perp \text{ U}^s \phi$$

Therefore, if we adopt the *strict* version, then it is possible to define LTL with the only temporal operator being the *until*.

- ... but encoding the standard until with the strict until requires more space:

$$\phi_1 \text{ U } \phi_2 \equiv \phi_2 \vee (\phi_1 \wedge \phi_1 \text{ U}^s \phi_2)$$



Definition (Negation Normal Form)

We define the $\text{nnf}(\cdot) : \text{LTL} \rightarrow \text{LTL}$ (*Negation Normal Form*) function as follows:

- $\text{nnf}(p) = p$
- $\text{nnf}(\phi_1 \wedge \phi_2) = \text{nnf}(\phi_1) \wedge \text{nnf}(\phi_2)$
- $\text{nnf}(\phi_1 \vee \phi_2) = \text{nnf}(\phi_1) \vee \text{nnf}(\phi_2)$
- $\text{nnf}(\text{X}\phi) = \text{X}(\text{nnf}(\phi))$
- $\text{nnf}(\phi_1 \text{ U } \phi_2) = (\text{nnf}(\phi_1)) \text{ U } (\text{nnf}(\phi_2))$
- $\text{nnf}(\phi_1 \text{ R } \phi_2) = (\text{nnf}(\phi_1)) \text{ R } (\text{nnf}(\phi_2))$

For any $\phi \in \text{LTL}$, the formula $\text{nnf}(\phi)$ has *negation only applied to atomic propositions*.



Definition (Negation Normal Form)

We define the $\text{nnf}(\cdot) : \text{LTL} \rightarrow \text{LTL}$ (*Negation Normal Form*) function as follows:

- $\text{nnf}(\neg p) = \neg p$
- $\text{nnf}(\neg\neg\phi) = \text{nnf}(\phi)$
- $\text{nnf}(\neg(\phi_1 \wedge \phi_2)) = \text{nnf}(\neg\phi_1) \vee \text{nnf}(\neg\phi_2)$
- $\text{nnf}(\neg(\phi_1 \vee \phi_2)) = \text{nnf}(\neg\phi_1) \wedge \text{nnf}(\neg\phi_2)$
- $\text{nnf}(\neg X\phi) = X(\text{nnf}(\neg\phi))$
- $\text{nnf}(\neg(\phi_1 \text{ U } \phi_2)) = (\text{nnf}(\neg\phi_1)) \text{ R } (\text{nnf}(\neg\phi_2))$
- $\text{nnf}(\neg(\phi_1 \text{ R } \phi_2)) = (\text{nnf}(\neg\phi_1)) \text{ U } (\text{nnf}(\neg\phi_2))$

For any $\phi \in \text{LTL}$, the formula $\text{nnf}(\phi)$ has *negation only applied to atomic propositions*.



Theorem (Kamp's Theorem over ω -words)

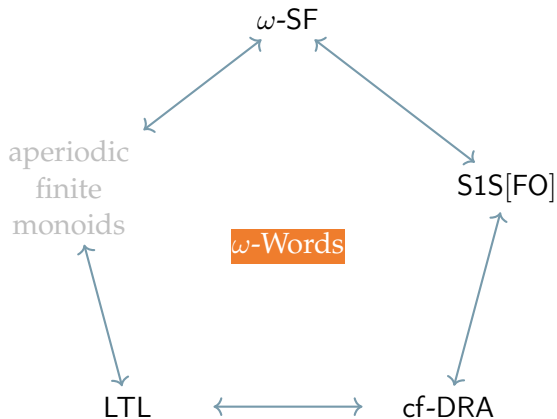
- For each LTL formula ϕ , there exists an S1S[FO] formula ψ such that $\mathcal{L}(\phi) = \mathcal{L}(\psi)$.
- For each S1S[FO] formula ψ , there exists an LTL formula ϕ such that $\mathcal{L}(\psi) = \mathcal{L}(\phi)$.

Reference:

Johan Anthony Wilem Kamp (1968). *Tense logic and the theory of linear order.*
University of California, Los Angeles



Characterizations of ω -Star-free Languages





The syntax of **LTL+P** is defined as follows:

$\phi := p \mid \neg\phi \mid \phi \vee \phi$	Boolean Modalities with $p \in \mathcal{AP}$
$\mid X\phi \mid \phi \text{ U } \phi$	Future Temporal Modalities
$\mid Y\phi \mid \phi \text{ S } \phi$	Past Temporal Modalities

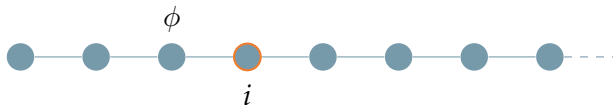
- $Y\phi$ is the **Yesterday** operator: *the previous time point exists and it satisfies the formula ϕ .*
- $\phi_1 \text{ S } \phi_2$ is the **Since** operator: *there exists a time point in the past where ϕ_2 is true, and ϕ_1 holds since (and excluding) that point up to now.*

Shortcuts:

- **Once**, $O\phi$: *there exists a time point in the past where ϕ holds.* $O\phi \equiv \top \text{ S } \phi$.
- **Historically**, $H\phi$: *for all time points in the past ϕ holds.* $H\phi \equiv \neg(O\neg\phi)$.

We say that σ satisfies at position i the LTL formula ϕ , written $\sigma, i \models \phi$, iff:

- $\sigma, i \models Y\phi$ iff $i > 0$ and $\sigma, i - 1 \models \phi$

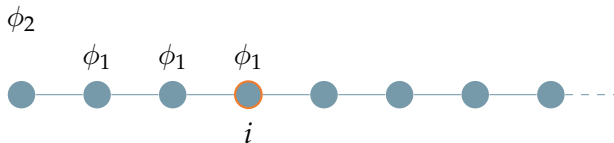


position i has a predecessor and ϕ holds at the *previous* position of i

Note: $\sigma, 0 \models Y\phi$ is always false.

We say that σ satisfies at position i the LTL formula ϕ , written $\sigma, i \models \phi$, iff:

- $\sigma, i \models \phi_1 \text{ S } \phi_2$ iff $\exists j \leq i . \sigma, j \models \phi_2$ and $\forall j < k \leq i . \sigma, k \models \phi_1$

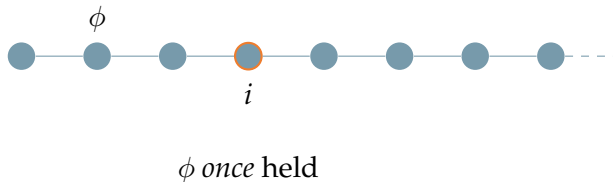


ϕ_1 holds *since* ϕ_2 held



Shortcuts:

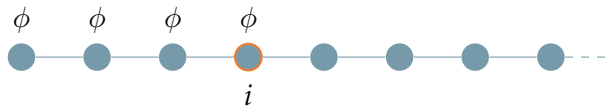
- (once) $O\phi \equiv \top S \phi$





Shortcuts:

- (historically) $H\phi \equiv \neg O\neg\phi$

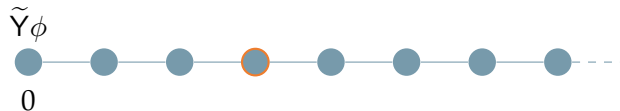


ϕ holds *always in the past*



Shortcuts:

- (*weak yesterday*) $\tilde{Y}\phi \equiv \neg Y\neg\phi$



ϕ holds at the *previous* position of i , if any

Note: $\sigma, i \models \tilde{Y}\perp$ is true iff $i = 0$.



Theorem

LTL+P is expressively equivalent to LTL.

Reference:

Dov M. Gabbay et al. (1980). “On the Temporal Analysis of Fairness”. In: *Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages, Las Vegas, Nevada, USA, January 1980*. Ed. by Paul W. Abrahams, Richard J. Lipton, and Stephen R. Bourne. ACM Press, pp. 163–173. URL: <https://doi.org/10.1145/567446.567462>



Theorem

$LTL+P$ *can be exponentially more succinct than* LTL .

Reference:

Nicolas Markey (2003). “Temporal logic with past is exponentially more succinct”. In: *Bull. EATCS* 79, pp. 122–128

We will see the proof :)



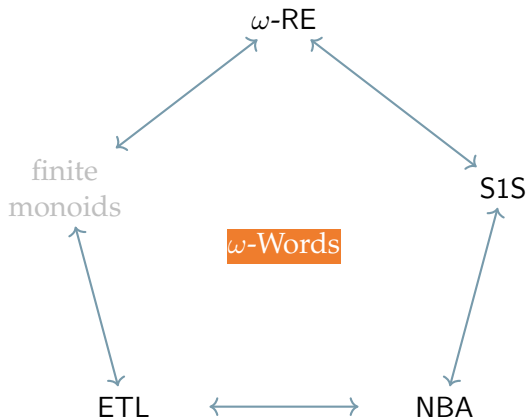
We have seen that LTL captures *star-free* ω -regular languages.

In order to capture all ω -regular languages, one can consider *Extended Linear Temporal Logic* (**ETL**, for short).

ETL = LTL + operators corresponding to *right-linear grammars*

Reference:

Pierre Wolper (1983). “Temporal logic can be more expressive”. In: *Information and control* 56.1-2, pp. 72–99. DOI: 10.1016/S0019-9958(83)80051-5





ω -REG

S1S

NBA

ETL

ω -SF

S1S[FO]

cf-DRA

LTL

REFERENCES



- Alessandro Artale et al. (2014).** “A Cookbook for Temporal Conceptual Data Modelling with Description Logics”. In: *ACM Trans. Comput. Log.* 15.3, 25:1–25:50. DOI: 10.1145/2629565.
- Fahiem Bacchus and Froduald Kabanza (1998).** “Planning for Temporally Extended Goals”. In: *Annals of Mathematics in Artificial Intelligence* 22.1-2, pp. 5–27.
- Ronen I. Brafman and Giuseppe De Giacomo (2019).** “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, pp. 1602–1608. DOI: 10.24963/ijcai.2019/222.
- Giuseppe De Giacomo et al. (2020).** “Imitation Learning over Heterogeneous Agents with Restraining Bolts”. In: *Proceedings of the 13th International Conference on Automated Planning and Scheduling*. AAAI Press, pp. 517–521.



- D. Della Monica et al. (2017).** “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*, pp. 1008–1014. DOI: [10.24963/ijcai.2017/140](https://doi.org/10.24963/ijcai.2017/140).
- Maria Fox and Derek Long (2003).** “PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains”. In: *J. Artif. Intell. Res.* 20, pp. 61–124. DOI: [10.1613/jair.1129](https://doi.org/10.1613/jair.1129).
- Dov M. Gabbay et al. (1980).** “On the Temporal Analysis of Fairness”. In: *Conference Record of the Seventh Annual ACM Symposium on Principles of Programming Languages, Las Vegas, Nevada, USA, January 1980*. Ed. by Paul W. Abrahams, Richard J. Lipton, and Stephen R. Bourne. ACM Press, pp. 163–173. URL: <https://doi.org/10.1145/567446.567462>.



- Lewis Hammond et al. (2021).** “Multi-Agent Reinforcement Learning with Temporal Logic Specifications”. In: *Proceedings of the 20th International Conference on Autonomous Agents and Multiagent Systems*. ACM, pp. 583–592. DOI: 10.5555/3463952.3464024.
- Swen Jacobs et al. (2017).** “The first reactive synthesis competition (SYNTCOMP 2014)”. In: *Int. J. Softw. Tools Technol. Transf.* 19.3, pp. 367–390. DOI: 10.1007/s10009-016-0416-3.
- Johan Anthony Wilem Kamp (1968).** *Tense logic and the theory of linear order*. University of California, Los Angeles.
- Nicolas Markey (2003).** “Temporal logic with past is exponentially more succinct”. In: *Bull. EATCS* 79, pp. 122–128.
- Amir Pnueli (1977).** “The temporal logic of programs”. In: *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE, pp. 46–57. DOI: 10.1109/SFCS.1977.32.



Bibliography IV

- Arthur N. Prior (1957). *Time and Modality*. London: Oxford University Press.
- Johan van Benthem et al. (2009). "Merging Frameworks for Interaction". In: *J. Philos. Log.* 38.5, pp. 491–526. DOI: 10.1007/s10992-008-9099-x.
- Pierre Wolper (1983). "Temporal logic can be more expressive". In: *Information and control* 56.1-2, pp. 72–99. DOI: 10.1016/S0019-9958(83)80051-5.