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, 2024

Temporal Logics

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 (2003). ***Time and modality*****.** John Locke Lecture

- 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

Temporal logic in AI

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

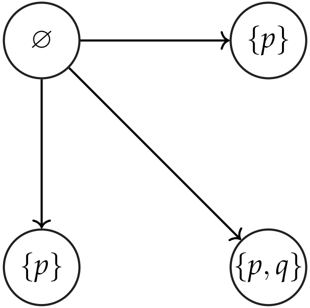
- automated planning
  - temporally extended goals (Bacchus and Kabanza 1998)
  - temporal planning (Fox and Long 2003)
  - timeline-based planning (Della Monica et al. 2017)
- automated synthesis (Jacobs et al. 2017)
- autonomy under uncertainty (Brafman and De Giacomo 2019)
  - specification of goals 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

Modal Logic

- 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

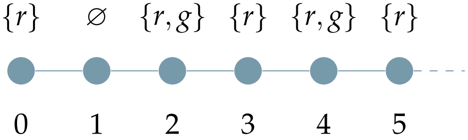


- $\mathcal{AP} = \{p, q\}$
- $\Box p$ is true
- $\Diamond q$ is true
- $\Box q$ is false
- $\Box p \vee \Box q$ is true

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)

$$\mathcal{AP} := \{r, g\}$$



- 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

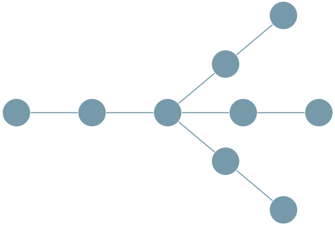
Reference:

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*.

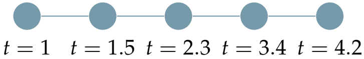
Infinite

Finite

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

Qualitative

Real-time



Representing time

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

Discrete

Dense

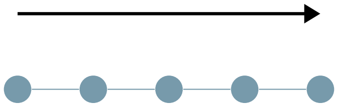
There are many choices to be made for the representation of *time*.

Points

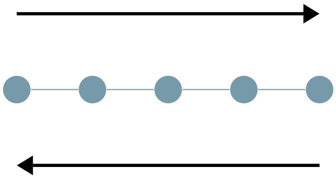
Intervals

Other parameters

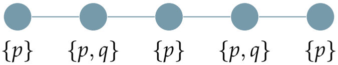
Pure Future



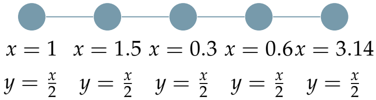
Past-Future



Propositional



First-Order



Our choice

We focus here on:

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

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

$$\begin{aligned} \phi := p \mid \neg\phi \mid \phi \vee \phi & \qquad \text{Boolean Modalities with } p \in \mathcal{AP} \\ \mid \mathsf{X}\phi \mid \phi \,\mathsf{U}\, \phi & \qquad \text{Future Temporal Modalities} \end{aligned}$$

- $\mathsf{X}\phi$ is the Next operator: *at the next time point (tomorrow), the formula ϕ holds*
- $\phi_1 \,\mathsf{U}\, \phi_2$ 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, $\mathsf{F}\phi$: *there exists a time point in the future where ϕ holds*. It is defined as $\mathsf{F}\phi \equiv \top \,\mathsf{U}\, \phi$.
- Globally, $\mathsf{G}\phi$: *for all time points in the future ϕ holds*. It is defined as $\mathsf{G}\phi \equiv \neg(\mathsf{F}\neg\phi)$.

Example:

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

$$\mathsf{GF}(p)$$

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:

$$\mathsf{GF}(p)$$

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:

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

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:

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

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).*

$$\mathsf{G}(r \rightarrow \mathsf{F}(g))$$

Which state sequences are *models* of the formula?

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

Example:

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

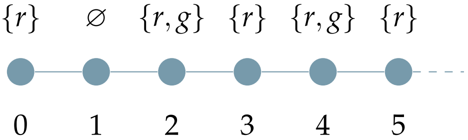
$$\mathsf{G}(r \rightarrow \mathsf{F}(g))$$

Which state sequences are *models* of the formula?

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

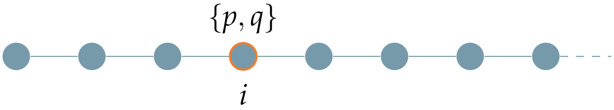
- Given a set of atomic propositions $\mathcal{AP}$, any LTL formula defined over $\mathcal{AP}$ is interpreted over *infinite words* $\sigma \in (2^{\mathcal{AP}})^\omega$.
- Let $\sigma = \langle \sigma_0, \sigma_1, \ldots \rangle$. For each $i \geq 0$, $\sigma_i \subseteq \mathcal{AP}$ is called a 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.

$\mathcal{AP} := \{r, g\}$



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

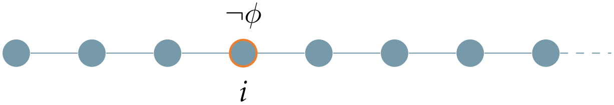
- $\sigma, i \models p$ iff $p \in \sigma_i$



p holds at position i

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

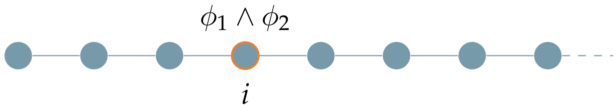
- $\sigma, i \models \neg\phi$ iff $\sigma, i \not\models \phi$



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:

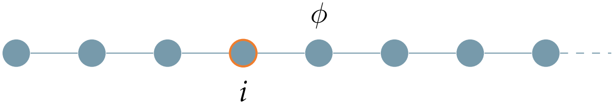
- $\sigma, i \models \phi_1 \wedge \phi_2$ iff $\sigma, i \models \phi_1$ and $\sigma, i \models \phi_2$



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:

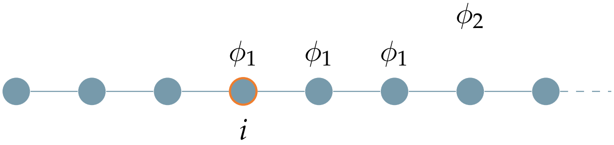
- $\sigma, i \models \mathsf{X}\phi$ iff $\sigma, i+1 \models \phi$



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:

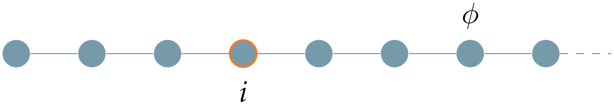
- $\sigma, i \models \phi_1 \mathsf{U} \phi_2$ iff $\exists j \geq i \,.\, \sigma, j \models \phi_2$ and $\forall i \leq k < j \,.\, \sigma, k \models \phi_1$



ϕ_1 holds *until* ϕ_2 holds

Shortcuts:

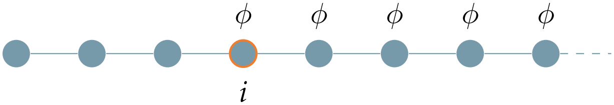
- (*eventually*) $\mathsf{F}\phi \equiv \top \; \mathsf{U} \; \phi$



ϕ will *eventually* hold

Shortcuts:

- (*globally*) $\mathsf{G}\phi \equiv \neg\mathsf{F}\neg\phi$



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

Example:

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

$$\mathsf{F}(p \wedge \mathsf{X}q)$$

Which state sequences are *models* of the formula?

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

Example:

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

$$\mathsf{F}(p \wedge \mathsf{X}q)$$

Which state sequences are *models* of the formula?

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

Example:

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

$$\mathsf{F}(p) \wedge \mathsf{F}(q) \qquad \mathsf{F}(p \wedge \mathsf{F}q) \qquad \mathsf{F}(p \wedge q)$$

Which state sequences are *models* of the formula?

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

Example:

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

$$\mathsf{F}(p) \wedge \mathsf{F}(q) \qquad \mathsf{F}(p \wedge \mathsf{F}q) \qquad \mathsf{F}(p \wedge q)$$

Which state sequences are *models* of the formula?

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

Example:

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

$$p \mathsf{U} (\mathsf{G} q)$$

Write an equivalent ω-regular expression.

Example:

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

$$p \mathsf{U} (\mathsf{G} q)$$

Write an equivalent ω-regular expression.

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

Example:

Consider $\mathcal{AP} = \{p, q\}$.
Is the formula $\mathsf{FX}p$ equivalent to $\mathsf{XF}p$?

Example:

Consider $\mathcal{AP} = \{p, q\}$.
Is the formula $\mathsf{FX}p$ equivalent to $\mathsf{XF}p$?
Yes.

Example:

Consider $\mathcal{AP} = \{p, q\}$.
Is the formula $\mathsf{FX}p$ equivalent to $\mathsf{XF}p$?
Yes.

Exercise:

Consider $\mathcal{AP} = \{p, q\}$.
Write the formula $(\mathsf{G}p)\ \mathsf{U}\ q$ 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.

$$\mathsf{GF}(en) \to \mathsf{GF}(tk)$$

This is very different from $\mathsf{GF}(en \to tk)$.

Compassion:

Consider $\mathcal{AP} = \{en, tk\}$.
It is not possible that a transition is enabled infinitely many times but taken only finitely many times.

$$\mathsf{GF}(en) \rightarrow \mathsf{GF}(tk)$$

This is very different from $\mathsf{GF}(en \rightarrow tk)$.

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.

$$\mathsf{GF}(en) \rightarrow \mathsf{GF}(tk)$$

This is very different from $\mathsf{GF}(en \rightarrow tk)$.

Justice:

Consider $\mathcal{AP} = \{en, tk\}$.
It is never the case that a transition is always enabled but never taken.

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

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

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

- $\sigma, i \models \phi_1 \mathsf{U}^s \phi_2$ 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 $\mathsf{X}\phi$ 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:

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

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

- $\sigma, i \models \phi_1 \, \mathsf{U}^s \, \phi_2$ 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 $\mathsf{X}\phi$ with only the strict version of the until?

$$\mathsf{X}\phi \equiv \bot \, \mathsf{U}^s \, \phi$$

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:

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

Definition (Negation Normal Form)

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

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

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

Definition (Negation Normal Form)

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

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

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

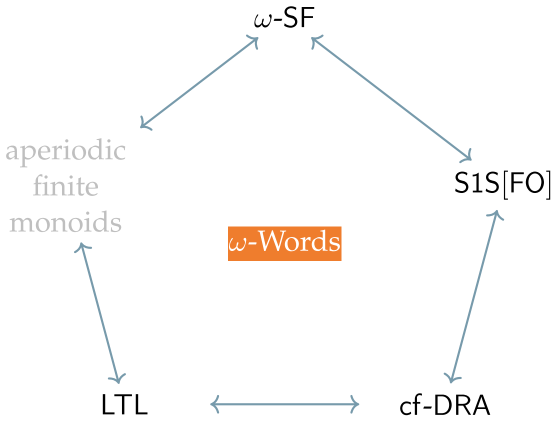
Theorem (Kamp's Theorem over ω-words)

- *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 $p \in \mathcal{AP}$
$\mid \mathsf{X}\phi \mid \phi \,\mathsf{U}\, \phi$	Future Temporal Modalities
$\mid \mathsf{Y}\phi \mid \phi \,\mathsf{S}\, \phi$	Past Temporal Modalities

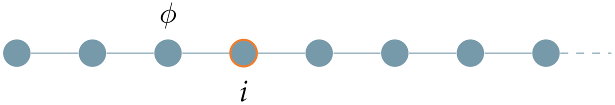
- $\mathsf{Y}\phi$ is the Yesterday operator: *the previous time point exists and it satisfies the formula* ϕ.
- $\phi_1 \,\mathsf{S}\, \phi_2$ is the Since operator: *there exists a time point in the past where* ϕ_2 *is true, and* ϕ_1 *holds since (and excluding) that point up to now.*

Shortcuts:

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

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

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

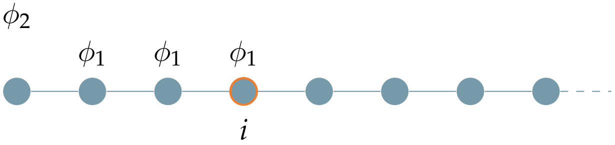


position i has a predecessor and ϕ holds at the *previous* position of i

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

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

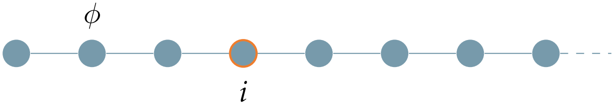
- $\sigma, i \models \phi_1 \mathsf{S}\, \phi_2$ iff $\exists j \leq i \,.\, \sigma, j \models \phi_2$ and $\forall j < k \leq i \,.\, \sigma, k \models \phi_1$



ϕ_1 holds *since* ϕ_2 held

Shortcuts:

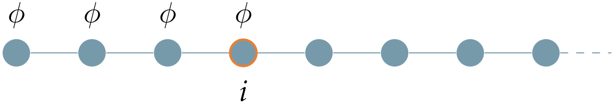
- (*once*) $\mathsf{O}\phi \equiv \top \mathsf{S} \phi$



ϕ *once* held

Shortcuts:

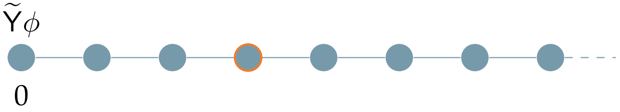
- (*historically*) $\mathsf{H}\phi \equiv \neg\mathsf{O}\neg\phi$



ϕ holds *always in the past*

Shortcuts:

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



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

Note: $\sigma, i \models \widetilde{\mathsf{Y}}\perp$ 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 :)

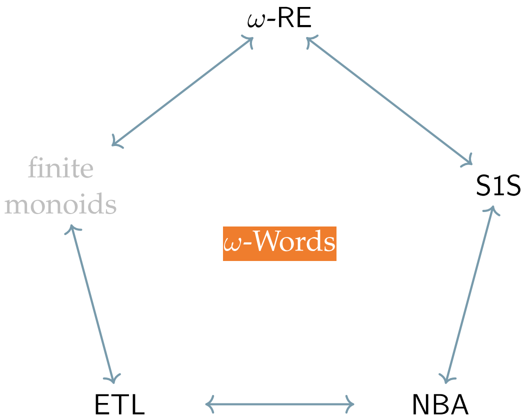
We have seen that LTL captures *star-free* ω-regular languages.
In order to capture all ω-regular languages, one can consider *Extended Linear Temporal Logic* (ETL, for short).

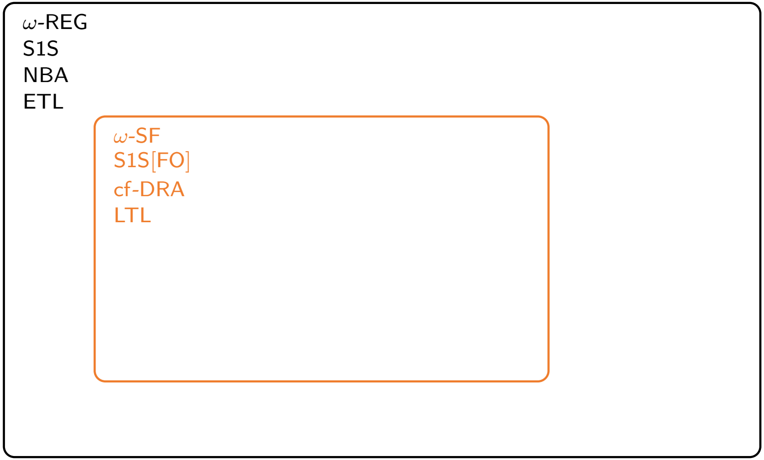
ETL = LTL + operators corresponding to *right-linear grammars*

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

Characterizations of ω-Regular Languages




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.

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.

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 (2003). ***Time and modality*.** John Locke Lecture.

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.