Linear Temporal Logic (LTL) is a formalism used to describe the behavior of systems over time, particularly useful in formal verification. LTL can describe the past, present, and future states of a system. Here is a detailed comparison of LTL and its various extensions:

**Linear Temporal Logic (LTL)**

* **Focus**: Describes the temporal sequence of system behaviors.
* **Operators**: Includes “Next” (X), “Until” (U), “Finally” (F), and “Globally” (G).
* **Application**: Widely used in formal verification to ensure that systems meet specified properties across all possible execution paths.

**Future Time LTL**

* **Focus**: Specifically describes the properties of a system’s future states.
* **Operators**: Uses future temporal operators such as “Finally” (F), “Globally” (G), “Until” (U), and “Next” (X).
* **Application**: Suitable for verifying future behaviors of systems, ensuring that certain conditions will eventually be true (Pnueli, 1977).

**Past Time LTL**

* **Focus**: Specifically describes the properties of a system’s past states.
* **Operators**: Introduces past temporal operators like “Previously” (P), “Historically” (H), “Since” (S), and “Once” (O).
* **Application**: Useful in runtime verification and debugging, where understanding the historical behavior of a system is crucial (Havelund and Peled, 2019).

**Metric Temporal Logic (MTL)**

* **Focus**: Describes system behaviors within specific time intervals.
* **Operators**: Extends LTL with time-constrained operators such as “Finally within interval I” (F\_I) and “Globally within interval I” (G\_I).
* **Application**: Ideal for real-time systems where precise timing constraints are essential, such as ensuring a response within a certain time frame (Koymans, 1990).

**Mission Time LTL (MLTL)**

* **Focus**: Specifies requirements for mission-based system operations.
* **Operators**: Similar to MTL but often uses integer bounds for time intervals.
* **Application**: Tailored for mission-critical systems like unmanned aerial vehicles, where tasks must be completed within known durations (Li, Vardi, and Rozier, 2019).

**Summary of Differences**

* **Temporal Focus**: Future Time LTL and Past Time LTL focus on future and past states, respectively, while LTL can describe all temporal states.
* **Time Constraints**: MTL and MLTL introduce specific time constraints. MTL is more general, while MLTL is designed for mission-based systems.
* **Application Scenarios**: LTL is used for broad formal verification, Future Time LTL and Past Time LTL for future and past behavior verification, MTL for real-time systems, and MLTL for mission-critical systems.

These logics extend LTL to address different temporal aspects, providing powerful tools for specifying and verifying system behaviors in various contexts.

References:

* Pnueli, A. (1977) ‘The Temporal Logic of Programs’, *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*. IEEE, pp. 46-57.
* Havelund, K. and Peled, D. (2019) ‘An Extension of LTL with Rules and its Application to Runtime Verification’, *Runtime Verification*, pp. 1-22.
* Koymans, R. (1990) ‘Specifying real-time properties with metric temporal logic’, *Real-Time Systems*, 2(4), pp. 255-299.
* Li, J., Vardi, M.Y. and Rozier, K.Y. (2019) ‘Satisfiability checking for Mission-time LTL (MLTL)’, *Information and Computation*. Available at: <https://doi.org/10.1016/j.ic.2022.104923>.
* Put some examples, focus on the difference between Future time and mission time , in terms of time intervals

**运算符和语法上的相似性与不同**

**线性时态逻辑（LTL）**

* **运算符**：
  + **X（Next）**：在下一个状态中某个条件成立。
  + **U（Until）**：某个条件会一直成立直到另一个条件成立。
  + **F（Finally）**：某个条件最终会成立。
  + **G（Globally）**：某个条件在所有未来状态中都成立。
* **语法**：LTL公式由命题变量和时态运算符构成，描述系统的时间序列行为。

**未来时间LTL（Future Time LTL）**

* **运算符**：与LTL相同，专注于未来状态。
  + **X（Next）**、**U（Until）**、**F（Finally）**、**G（Globally）**。
* **语法**：与LTL相同，但仅描述未来状态。

**过去时间LTL（Past Time LTL）**

* **运算符**：
  + **P（Previously）**：在前一个状态中某个条件成立。
  + **H（Historically）**：某个条件在所有过去状态中一直成立。
  + **S（Since）**：某个条件自另一个条件成立以来一直成立。
  + **O（Once）**：某个条件在过去某个时刻成立。
* **语法**：扩展LTL，加入过去时态运算符，描述系统的历史行为。

**度量时态逻辑（MTL）**

* **运算符**：
  + **F\_I（Finally within interval I）**：在时间间隔I内某个条件最终会成立。
  + **G\_I（Globally within interval I）**：在时间间隔I内某个条件始终成立。
  + **U\_I（Until within interval I）**：某个条件会一直成立直到另一个条件在时间间隔I内成立。
* **语法**：扩展LTL，加入时间约束运算符，描述系统在特定时间范围内的行为。

**任务时间LTL（MLTL）**

* **运算符**：与MTL相似，但通常使用整数边界的时间区间。
  + **F\_I（Finally within interval I）**、**G\_I（Globally within interval I）**、**U\_I（Until within interval I）**。
* **语法**：类似于MTL，但专注于任务驱动系统的时间约束。

**相似性**

* **基本运算符**：所有逻辑都基于LTL的基本运算符（X、U、F、G）。
* **时态描述**：都用于描述系统在时间维度上的行为。

**不同**

* **时间方向**：未来时间LTL专注于未来，过去时间LTL专注于过去，LTL涵盖过去、现在和未来。
* **时间约束**：MTL和MLTL引入具体的时间约束，MLTL特别适用于任务驱动系统。
* **应用场景**：LTL用于广泛的形式验证，未来时间LTL和过去时间LTL分别用于未来和过去行为的验证，MTL用于实时系统的时间约束验证，MLTL用于任务关键系统的时间约束。

这些逻辑在运算符和语法上的扩展，使得它们能够更精确地描述和验证系统在不同时间维度和应用场景中的行为。