

INF5140 – Specification and Verification of Parallel Systems

Spring 2018

Institutt for informatikk, Universitetet i Oslo

February 16, 2018



Linear-Time Temporal Logic (LTL)

Temporal Logic?

- Temporal logic is the logic of “time”^a
- It is a *modal* logic.
- There are different ways of modeling time.
 - linear time vs. branching time
 - time instances vs. time intervals
 - discrete time vs. continuous time
 - past and future vs. future only

^apay attention, it will be something kind of abstract, it's mostly not what's known as *real-time*, but there are variants of temporal logics which can handle real-time. They *won't* occur in this lecture.

First Order Logic

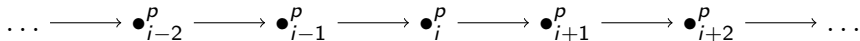
- We have used FOL to express properties of states.
 - $\langle x : 21, y : 49 \rangle \models x < y$
 - $\langle x : 21, y : 7 \rangle \not\models x < y$
- A computation is a sequence of states.
- To express properties of computations, we need to extend FOL.
- This we can do using temporal logic.

LTL: speaking about “time”

In **Linear Temporal Logic (LTL)** (also called linear-time temporal logic) we can describe such properties as follows: assume time is a **sequence**¹ of discrete points i in time, then: if i is *now*,

- p holds in i and every following point (the future)
- p holds in i and every preceding point (the past)

We will only be concerned with the future.



¹a sequence is linear

LTL operators

We extend our first-order language² \mathcal{L} to a temporal language \mathcal{L}_T by adding the **temporal operators** \square , \diamond , \bigcirc , U , R and W .

Interpretation of the operators

$\square\varphi$	φ will <i>always</i> (in every state) hold
$\diamond\varphi$	φ will <i>eventually</i> (in some state) hold
$\bigcirc\varphi$	φ will hold at the <i>next</i> point in time
$\varphi U\psi$	ψ will eventually hold, and <i>until</i> that point φ will hold
$\varphi R\psi$	ψ holds until (incl.) the point (if any) where φ holds (<i>release</i>)
$\varphi W\psi$	φ will hold until ψ holds (<i>weak until</i> or <i>waiting for</i>)

²Note: it's equally ok to extend a propositional language the same way. The difference is between a first-order LTL or propositional LTL.

We define **LTL formulae** as follows.

Definition

- $\mathcal{L} \subseteq \mathcal{L}_T$: first-order formulae are also LTL formulae.
- If φ is an LTL formula, so are the following.

$$\Box\varphi \quad \Diamond\varphi \quad \bigcirc\varphi \quad \neg\varphi$$

- If φ and ψ are LTL formulae, so are

$$\begin{aligned} &\varphi U \psi \quad \varphi R \psi \quad (\varphi W \psi) \\ &(\varphi \vee \psi) \quad (\varphi \wedge \psi) \quad (\varphi \rightarrow \psi) \quad (\varphi \leftrightarrow \psi) \end{aligned}$$

- nothing else

Definition

- A **path** is an infinite sequence

$$\sigma = s_0, s_1, s_2, \dots$$

of states.

- σ^k denotes the *path* $s_k, s_{k+1}, s_{k+2}, \dots$
- σ_k denotes the *state* s_k .
- All computations are paths, but not vice versa.

Definition

We define the notion that an LTL formula φ is **true** (**false**) relative to a path σ , written $\sigma \models \varphi$ ($\sigma \not\models \varphi$) as follows.

$\sigma \models \varphi$ iff $\sigma_0 \models \varphi$ when $\varphi \in \mathcal{L}$

$\sigma \models \neg\varphi$ iff $\sigma \not\models \varphi$

$\sigma \models \varphi \vee \psi$ iff $\sigma \models \varphi$ or $\sigma \models \psi$

$\sigma \models \Box\varphi$ iff $\sigma^k \models \varphi$ for all $k \geq 0$

$\sigma \models \Diamond\varphi$ iff $\sigma^k \models \varphi$ for some $k \geq 0$

$\sigma \models \bigcirc\varphi$ iff $\sigma^1 \models \varphi$

(cont.)

Definition

(cont.)

$\sigma \models \varphi U \psi$ iff $\sigma^k \models \psi$ for some $k \geq 0$, and
 $\sigma^i \models \varphi$ for every i such that $0 \leq i < k$

$\sigma \models \varphi R \psi$ iff for every $j \geq 0$,
 if $\sigma^i \not\models \varphi$ for every $i < j$ then $\sigma^j \models \psi$

$\sigma \models \varphi W \psi$ iff $\sigma \models \varphi U \psi$ or $\sigma \models \Box \varphi$

Definition

- We say that φ is **(temporally) valid**, written $\models \varphi$, if
$$\sigma \models \varphi \text{ for all paths } \sigma.$$
- We say that φ and ψ are **equivalent**, written $\varphi \sim \psi$, if
$$\models \varphi \leftrightarrow \psi \text{ (i.e. } \sigma \models \varphi \text{ iff } \sigma \models \psi, \text{ for all } \sigma).$$

Example

\Box distributes over \wedge , while \Diamond distributes over \vee .

$$\Box(\varphi \wedge \psi) \sim (\Box\varphi \wedge \Box\psi)$$

$$\Diamond(\varphi \vee \psi) \sim (\Diamond\varphi \vee \Diamond\psi)$$

$$\sigma \models \Box p$$



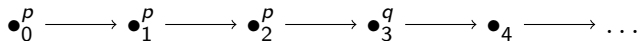
$$\sigma \models \Diamond p$$



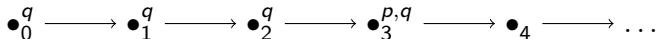
$$\sigma \models \bigcirc p$$



$\sigma \models pUq$ (sequence of p 's is **finite**)



$\sigma \models pRq$ (The sequence of qs may be infinite)



$\sigma \models pWq$. The sequence of ps may be infinite.
($pWq \sim pUq \vee \Box p$).



Observation

- [Manna and Pnueli, 1992] uses pairs (σ, j) of paths and positions instead of just the path σ because they have **past-formulae**: formulae without future operators (the ones we use) but possibly with **past operators**, like \square^{-1} and \diamond^{-1} .

$$(\sigma, j) \models \square^{-1}\varphi \quad \text{iff} \quad (\sigma, k) \models \varphi \text{ for all } k, 0 \leq k \leq j$$

$$(\sigma, j) \models \diamond^{-1}\varphi \quad \text{iff} \quad (\sigma, k) \models \varphi \text{ for some } k, 0 \leq k \leq j$$

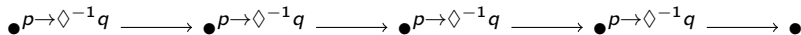
- However, it can be shown that for any formula φ , there is a **future-formula** (formulae without past operators) ψ such that

$$(\sigma, 0) \models \varphi \quad \text{iff} \quad (\sigma, 0) \models \psi$$

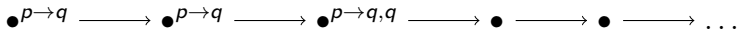
Example

What is a future version of $\Box(p \rightarrow \Diamond^{-1}q)$?

$(\sigma, 0) \models \Box(p \rightarrow \Diamond^{-1}q)$



$(\sigma, 0) \models qR(p \rightarrow q)$



Example

$\varphi \rightarrow \Diamond\psi$: If φ holds initially, then ψ holds eventually.



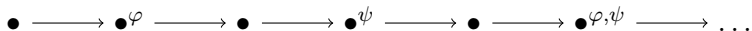
This formula will also hold in every path where φ does not hold initially.



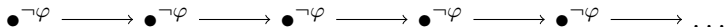
Example (Response)

$\Box(\varphi \rightarrow \Diamond\psi)$

Every φ -position coincides with or is followed by a ψ -position.



This formula will also hold in every path where φ never holds.



Example

$\Box\Diamond\psi$

There are infinitely many ψ -positions.

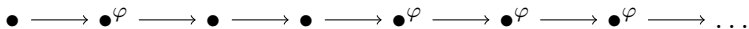


This formula can be obtained from the previous one, $\Box(\varphi \rightarrow \Diamond\psi)$, by letting $\varphi = \top$: $\Box(\top \rightarrow \Diamond\psi)$.

Example

$\diamond\Box\varphi$

Eventually φ will hold permanently.

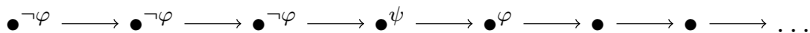


Equivalently: there are **finitely** many $\neg\varphi$ -positions.

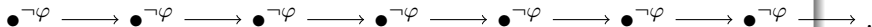
Example

$(\neg\varphi)W\psi$

The first φ -position must coincide or be preceded by a ψ -position.



φ may never hold



Example

$\Box(\varphi \rightarrow \psi W \chi)$

Every φ -position initiates a sequence of ψ -positions, and if terminated, by a χ -position.



The sequence of ψ -positions need not terminate.



Nested waiting-for

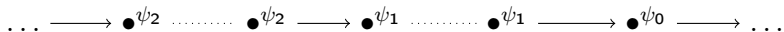
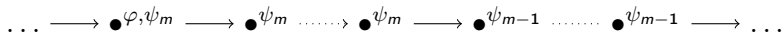
A **nested waiting-for formula** is of the form

$$\Box(\varphi \rightarrow (\psi_m W (\psi_{m-1} W \cdots (\psi_1 W \psi_0) \cdots))),$$

where $\varphi, \psi_0, \dots, \psi_m \in \mathcal{L}$. For the sake of convenience, we write

$$\Box(\varphi \rightarrow \psi_m W \psi_{m-1} W \cdots W \psi_1 W \psi_0).$$

Every φ -position initiates a succession of intervals, beginning with a ψ_m -interval, ending with a ψ_1 -interval and possibly terminated by a ψ_0 -position. Each interval may be empty or extend to infinity.



Capturing informally understood temporal specifications formally

It can be difficult to correctly formalize informally stated requirements in temporal logic.

Example

How does one formalize the informal requirement “ φ implies ψ ”?

- $\varphi \rightarrow \psi$? $\varphi \rightarrow \psi$ holds in the initial state.
- $\Box(\varphi \rightarrow \psi)$? $\varphi \rightarrow \psi$ holds in every state.
- $\varphi \rightarrow \Diamond\psi$? φ holds in the initial state, ψ will hold in some state.
- $\Box(\varphi \rightarrow \Diamond\psi)$? We saw this earlier.
- None of these is necessarily what we intended

Definition (Duals)

For binary boolean connectives^a \circ and \bullet , we say that \bullet is the **dual** of \circ if

$$\neg(\varphi \circ \psi) \sim (\neg\varphi \bullet \neg\psi).$$

Similarly for unary connectives: \bullet is the dual of \circ if $\neg \circ \varphi \sim \bullet \neg \varphi$.

^aThose are not concrete connectives or operators, they are meant as “placeholders”

Duality is symmetric:

- If \bullet is the dual of \circ then
- \circ is the dual of \bullet , thus
- we may refer to two connectives as dual (of each other).

Which connectives are duals?

- \wedge and \vee are duals:

$$\neg(\varphi \wedge \psi) \sim (\neg\varphi \vee \neg\psi).$$

- \neg is its own dual:

$$\neg\neg\varphi \sim \varphi.$$

- What is the dual of \rightarrow ? It's \leftarrow :

$$\begin{aligned}\neg(\varphi \rightarrow \psi) &\sim \varphi \leftarrow \psi \\ &\sim \psi \rightarrow \varphi \\ &\sim \neg\varphi \rightarrow \neg\psi\end{aligned}$$

Complete sets of connectives

- A set of connectives is **complete** (for boolean formulae) if every other connective can be defined in terms of them.
- Our set of connectives is complete (e.g., \nrightarrow can be defined), but also subsets of it, so we don't actually need all the connectives.

Example

$\{\vee, \neg\}$ is complete.

- \wedge is the dual of \vee .
- $\varphi \rightarrow \psi$ is equivalent to $\neg\varphi \vee \psi$.
- $\varphi \leftrightarrow \psi$ is equivalent to $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$.
- \top is equivalent to $p \vee \neg p$
- \perp is equivalent to $p \wedge \neg p$

We can extend the notions of duality and completeness to temporal formulae.

Duals of temporal operators

- What is the dual of \Box ? And of \Diamond ?
- \Box and \Diamond are duals.

$$\neg\Box\varphi \sim \Diamond\neg\varphi$$

$$\neg\Diamond\varphi \sim \Box\neg\varphi$$

- Any other?
- U and R are duals.

$$\neg(\varphi U\psi) \sim (\neg\varphi)R(\neg\psi)$$

$$\neg(\varphi R\psi) \sim (\neg\varphi)U(\neg\psi)$$

We don't need all our temporal operators either.

Proposition

$\{\vee, \neg, U, \bigcirc\}$ is complete for LTL.

- Proof:**
- $\diamond\varphi \sim \top U\varphi$
 - $\Box\varphi \sim \perp R\varphi$
 - $\varphi R\psi \sim \neg(\neg\varphi U\neg\psi)$
 - $\varphi W\psi \sim \Box\varphi \vee (\varphi U\psi)$

□

We can classify properties expressible in LTL.

Classification

safety $\Box\varphi$

liveness $\Diamond\varphi$

obligation $\Box\varphi \vee \Diamond\psi$

recurrence $\Box\Diamond\varphi$

persistence $\Diamond\Box\varphi$

reactivity $\Box\Diamond\varphi \vee \Diamond\Box\psi$

- important basic class of properties
- relation to testing and run-time verification
- “nothing bad ever happens”

Definition (Safety)

- A **safety** formula is of the form

$$\Box\varphi$$

for some first-order formula φ .

- A **conditional safety** formula is of the form

$$\varphi \rightarrow \Box\psi$$

for (first-order) formulae φ and ψ .

- Safety formulae express *invariance* of some state property φ : that φ holds in every state of the computation.

Example

- *Mutual exclusion* is a safety property. Let C_i denote that process P_i is executing in the critical section. Then

$$\Box \neg (C_1 \wedge C_2)$$

expresses that it should always be the case that not both P_1 and P_2 are executing in the critical section.

- Observe that the negation of a safety formula is a liveness formula; the negation of the formula above is the liveness formula

$$\Diamond (C_1 \wedge C_2)$$

which expresses that eventually it *is* the case that both P_1 and P_2 are executing in the critical section.

Definition (Liveness)

- A **liveness** formula is of the form

$$\diamond\varphi$$

for some first-order formula φ .

- A **conditional liveness** formula is of the form

$$\varphi \rightarrow \diamond\psi$$

for first-order formulae φ and ψ .

- Liveness formulae *guarantee* that some event φ eventually happens: that φ holds in at least one state of the computation.

Observation

- **Partial correctness** is a safety property. Let P be a program and ψ the post condition.

$$\Box(\textit{terminated}(P) \rightarrow \psi)$$

- In the case of **full partial correctness**, where there is a precondition φ , we get a *conditional safety* formula,

$$\varphi \rightarrow \Box(\textit{terminated}(P) \rightarrow \psi),$$

which we can express as $\{ \varphi \} P \{ \psi \}$ in Hoare Logic.

Observation

- **Total correctness** is a liveness property. Let P be a program and ψ the post condition.

$$\diamond(\textit{terminated}(P) \wedge \psi)$$

- In the case of **full total correctness**, where there is a precondition φ , we get a *conditional liveness* formula,

$$\varphi \rightarrow \diamond(\textit{terminated}(P) \wedge \psi).$$

Observation

Partial and total correctness are dual.

Let

$$PC(\psi) \triangleq \Box(\textit{terminated} \rightarrow \psi)$$

$$TC(\psi) \triangleq \Diamond(\textit{terminated} \wedge \psi)$$

Then

$$\neg PC(\psi) \sim PC(\neg\psi)$$

$$\neg TC(\psi) \sim TC(\neg\psi)$$

Definition (Obligation)

- A **simple obligation** formula is of the form

$$\Box\varphi \vee \Diamond\psi$$

for first-order formula φ and ψ .

- An equivalent form is

$$\Diamond\chi \rightarrow \Diamond\psi$$

which states that some state satisfies χ only if some state satisfies ψ .

Proposition

Every safety and liveness formula is also an obligation formula.

Proof: This is because of the following equivalences.

$$\Box\varphi \sim \Box\varphi \vee \Diamond\perp$$

$$\Diamond\varphi \sim \Box\perp \vee \Diamond\varphi$$

and the facts that $\models \neg\Box\perp$ and $\models \neg\Diamond\perp$.

□

Definition (Recurrence)

- A **recurrence** formula is of the form

$$\Box\Diamond\varphi$$

for some first-order formula φ .

- It states that infinitely many positions in the computation satisfies φ .

Observation

A response formula, of the form $\Box(\varphi \rightarrow \Diamond\psi)$, is equivalent to a recurrence formula, of the form $\Box\Diamond\chi$, if we allow χ to be a past-formula.

$$\Box(\varphi \rightarrow \Diamond\psi) \sim \Box\Diamond(\neg\varphi)W^{-1}\psi$$

Proposition

Weak fairness^a can be specified as the following recurrence formula.

$$\Box\Diamond(\text{enabled}(\tau) \rightarrow \text{taken}(\tau))$$

^aweak and strong fairness will be “recurrent” (sorry for the pun) themes. For instance they will show up again in the TLA presentation.

Observation

An equivalent form is

$$\Box(\Box\text{enabled}(\tau) \rightarrow \Diamond\text{taken}(\tau)),$$

which looks more like the first-order formula we saw last time.

Definition (Persistence)

- A **persistence** formula is of the form

$$\diamond \Box \varphi$$

for some first-order formula φ .

- It states that all but finitely many positions satisfy φ ^a
- Persistence formulae are used to describe the eventual **stabilization** of some state property.

^aIn other words: only finitely ("but") many position satisfy $\neg\varphi$. So at some point onwards, it's always φ .

Observation

Recurrence and persistence are duals.

$$\neg(\Box\Diamond\varphi) \sim (\Diamond\Box\neg\varphi)$$

$$\neg(\Diamond\Box\varphi) \sim (\Box\Diamond\neg\varphi)$$

Definition (Reactivity)

- A **simple reactivity** formula is of the form

$$\Box\Diamond\varphi \vee \Diamond\Box\psi$$

for first-order formula φ and ψ .

- A very general class of formulae are conjunctions of reactivity formulae.
- An equivalent form is

$$\Box\Diamond\chi \rightarrow \Box\Diamond\psi,$$

which states that if the computation contains infinitely many χ -positions, it must also contain infinitely many ψ -positions.

Proposition

Strong fairness can be specified as the following reactivity formula.

$$\square\diamond enabled(\tau) \rightarrow \square\diamond taken(\tau)$$

GCD Example

Below is a computation σ of our recurring GCD program.

- a and b are fixed: $\sigma \models \Box(a \doteq 21 \wedge b \doteq 49)$.
- $at(l)$ denotes the formulae ($\pi \doteq \{l\}$).
- *terminated* denotes the formula $at(l_8)$.

P-computation

States are of the form $\langle \pi, x, y, g \rangle$.

$$\begin{aligned} \sigma : \quad & \langle l_1, 21, 49, 0 \rangle \rightarrow \langle l_2^b, 21, 49, 0 \rangle \rightarrow \langle l_6, 21, 49, 0 \rangle \rightarrow \\ & \langle l_1, 21, 28, 0 \rangle \rightarrow \langle l_2^b, 21, 28, 0 \rangle \rightarrow \langle l_6, 21, 28, 0 \rangle \rightarrow \\ & \langle l_1, 21, 7, 0 \rangle \rightarrow \langle l_2^a, 21, 7, 0 \rangle \rightarrow \langle l_4, 21, 7, 0 \rangle \rightarrow \\ & \langle l_1, 14, 7, 0 \rangle \rightarrow \langle l_2^a, 14, 7, 0 \rangle \rightarrow \langle l_4, 14, 7, 0 \rangle \rightarrow \\ & \langle l_1, 7, 7, 0 \rangle \rightarrow \langle l_7, 7, 7, 0 \rangle \rightarrow \langle l_8, 7, 7, 7 \rangle \rightarrow \dots \end{aligned}$$

Does the following properties hold for σ ? And why?

1. \Box *terminated* (safety)
2. $at(l_1) \rightarrow$ *terminated*
3. $at(l_8) \rightarrow$ *terminated*
4. $at(l_7) \rightarrow \Diamond$ *terminated* (conditional liveness)
5. $\Diamond at(l_7) \rightarrow \Diamond$ *terminated* (obligation)
6. $\Box(\text{gcd}(x, y) \doteq \text{gcd}(a, b))$ (safety)
7. \Diamond *terminated* (liveness)
8. $\Diamond \Box(y \doteq \text{gcd}(a, b))$ (persistence)
9. $\Box \Diamond$ *terminated* (recurrence)

Exercises

1. Show that the following formulae are (not) LTL-valid.
 - 1.1 $\Box\varphi \leftrightarrow \Box\Box\varphi$
 - 1.2 $\Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi$
 - 1.3 $\neg\Box\varphi \rightarrow \Box\neg\Box\varphi$
 - 1.4 $\Box(\Box\varphi \rightarrow \psi) \rightarrow \Box(\Box\psi \rightarrow \varphi)$
 - 1.5 $\Box(\Box\varphi \rightarrow \psi) \vee \Box(\Box\psi \rightarrow \varphi)$
 - 1.6 $\Box\Diamond\Box\varphi \rightarrow \Diamond\Box\varphi$
 - 1.7 $\Box\Diamond\varphi \leftrightarrow \Box\Diamond\Box\Diamond\varphi$
2. A *modality* is a sequence of \neg , \Box and \Diamond , including the empty sequence ϵ . Two modalities σ and τ are *equivalent* if $\sigma\varphi \leftrightarrow \tau\varphi$ is valid.
 - 2.1 Which are the non-equivalent modalities in LTL, and
 - 2.2 what are their relationship (ie. implication-wise)?

References I

- [Andrews, 2000] Andrews, G. R. (2000).
Foundations of Multithreaded, Parallel, and Distributed Programming.
Addison-Wesley.
- [Blackburn et al., 2001] Blackburn, P., de Rijke, M., and Venema, Y. (2001).
Modal Logic.
Cambridge University Press.
- [Bowen and Hinchey, 2005] Bowen, J. P. and Hinchey, M. G. (2005).
Ten commandments revisited: a ten-year perspective on the industrial application of formal methods.
In *FMICS '05: Proceedings of the 10th international workshop on Formal methods for industrial critical systems*, pages 8–16, New York, NY, USA. ACM Press.
- [Garfinkel, 2005] Garfinkel, S. (2005).
History's worst software bugs.
Available at <http://archive.wired.com/software/coolapps/news/2005/11/69355?currentPage=all>.
- [Harel et al., 2000] Harel, D., Kozen, D., and Tiuryn, J. (2000).
Dynamic Logic.
Foundations of Computing. MIT Press.
- [Manna and Pnueli, 1992] Manna, Z. and Pnueli, A. (1992).
The temporal logic of reactive and concurrent systems—Specification.
Springer Verlag, New York.
- [Peled, 2001] Peled, D. (2001).
Software Reliability Methods.
Springer Verlag.
- [Schneider, 2004] Schneider, K. (2004).
Verification of Reactive Systems.
Springer Verlag.