Linear Temporal Logic

Natasha Alechina Brian Logan

Utrecht University

n.a.alechina@uu.nl b.s.logan@uu.nl

Reading for this lecture: Wojtek Jamroga, *Logical Methods for the Specification and Verification of Multiagent Systems*, Chapter 3.1.1 and 3.1.2.

Linear Time: LTL

- **LTL**: **Linear Temporal Logic**
- reasoning about a single computation, or run, of a system
- time is linear: just one possible future path is considered
- Model: a path
- important distinction: computational vs. behavioral structure

Linear Time: LTL

Definition (Models of LTL)

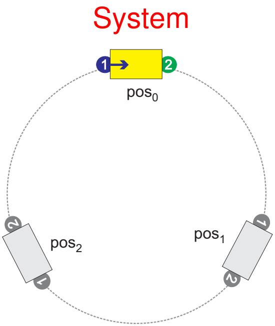
A **model of LTL** is a sequence of time moments (states). We call such models **paths**, and denote them by λ.

Evaluation of atomic propositions at particular time moments is also needed.

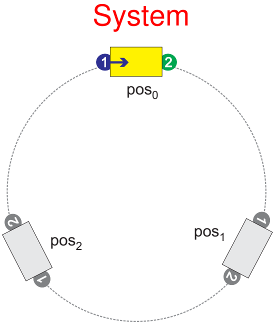
Notation:

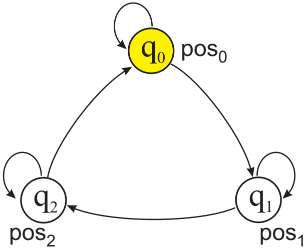
- $\lambda[i]$: ith time moment (starting from 0)
- $\lambda[i \ldots j]$: all time moments between i and j
- $\lambda[i \ldots \infty]$: all timepoints from i on

Computational vs. Behavioral Structures



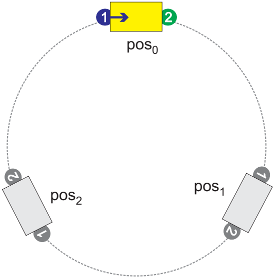
Computational vs. Behavioral Structures



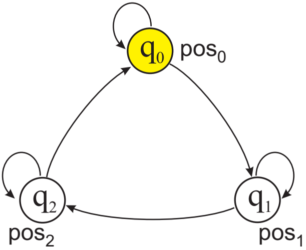


Computational vs. Behavioral Structures

System

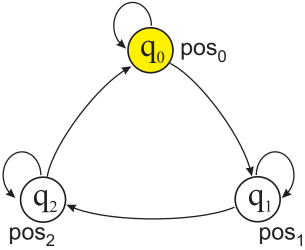


Computational str.



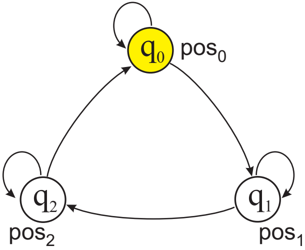
Computational vs. Behavioral Structures

Computational str.

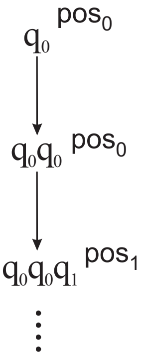


Computational vs. Behavioral Structures

Computational str.



Behavioral str.



Computational vs. Behavioral Structures: Beware

In LTL, models are defined as behavioral structures!

...But input to the verification problem is defined by the computational structure.

Linear Time: LTL

Definition (Semantics of LTL)

$\lambda \models p$ iff p is true at moment $\lambda[0]$ (that is, $\lambda[0] \in \mathcal{V}(p)$);

Linear Time: LTL

Definition (Semantics of LTL)

$\lambda \models p$	iff p is true at moment $\lambda[0]$ (that is, $\lambda[0] \in \mathcal{V}(p)$);
$\lambda \models X\varphi$	iff $\lambda[1] \models \varphi$;

Linear Time: LTL

Definition (Semantics of LTL)

$\lambda \models p$	iff p is true at moment $\lambda[0]$ (that is, $\lambda[0] \in \mathcal{V}(p)$);
$\lambda \models X\varphi$	iff $\lambda[1] \models \varphi$; No!

Linear Time: LTL

Definition (Semantics of LTL)

$\lambda \models p$	iff p is true at moment $\lambda[0]$ (that is, $\lambda[0] \in \mathcal{V}(p)$);
$\lambda \models X\varphi$	iff $\lambda[1..\infty] \models \varphi$;

Linear Time: LTL

Definition (Semantics of LTL)

$\lambda \models p$	iff p is true at moment $\lambda[0]$ (that is, $\lambda[0] \in \mathcal{V}(p)$);
$\lambda \models X\varphi$	iff $\lambda[1..\infty] \models \varphi$;
$\lambda \models F\varphi$	iff $\lambda[i..\infty] \models \varphi$ for some $i \geq 0$;

Linear Time: LTL

Definition (Semantics of LTL)

$\lambda \models p$	iff p is true at moment $\lambda[0]$ (that is, $\lambda[0] \in \mathcal{V}(p)$);
$\lambda \models X\varphi$	iff $\lambda[1..\infty] \models \varphi$;
$\lambda \models F\varphi$	iff $\lambda[i..\infty] \models \varphi$ for some $i \geq 0$;
$\lambda \models G\varphi$	iff $\lambda[i..\infty] \models \varphi$ for all $i \geq 0$;

Linear Time: LTL

Definition (Semantics of LTL)

$\lambda \models p$	iff p is true at moment $\lambda[0]$ (that is, $\lambda[0] \in \mathcal{V}(p)$);
$\lambda \models X\varphi$	iff $\lambda[1..\infty] \models \varphi$;
$\lambda \models F\varphi$	iff $\lambda[i..\infty] \models \varphi$ for some $i \geq 0$;
$\lambda \models G\varphi$	iff $\lambda[i..\infty] \models \varphi$ for all $i \geq 0$;
$\lambda \models \varphi U \psi$	iff $\lambda[i..\infty] \models \psi$ for some $i \geq 0$, and $\lambda[j..\infty] \models \varphi$ for all $0 \leq j < i$.

Linear Time: LTL

Definition (Semantics of LTL)

$\lambda \models p$	iff p is true at moment $\lambda[0]$ (that is, $\lambda[0] \in \mathcal{V}(p)$);
$\lambda \models X\varphi$	iff $\lambda[1..\infty] \models \varphi$;
$\lambda \models F\varphi$	iff $\lambda[i..\infty] \models \varphi$ for some $i \geq 0$;
$\lambda \models G\varphi$	iff $\lambda[i..\infty] \models \varphi$ for all $i \geq 0$;
$\lambda \models \varphi\, U\, \psi$	iff $\lambda[i..\infty] \models \psi$ for some $i \geq 0$, and $\lambda[j..\infty] \models \varphi$ for all $0 \leq j < i$.

$\lambda \models \neg\varphi$	iff not $\lambda \models \varphi$;
$\lambda \models \varphi \wedge \psi$	iff $\lambda \models \varphi$ and $\lambda \models \psi$.

Linear Time: LTL

Definition (Semantics of LTL)

$\lambda \models p$	iff p is true at moment $\lambda[0]$ (that is, $\lambda[0] \in \mathcal{V}(p)$);
$\lambda \models X\varphi$	iff $\lambda[1..\infty] \models \varphi$;
$\lambda \models F\varphi$	iff $\lambda[i..\infty] \models \varphi$ for some $i \geq 0$;
$\lambda \models G\varphi$	iff $\lambda[i..\infty] \models \varphi$ for all $i \geq 0$;
$\lambda \models \varphi\, U\, \psi$	iff $\lambda[i..\infty] \models \psi$ for some $i \geq 0$, and $\lambda[j..\infty] \models \varphi$ for all $0 \leq j < i$.

$\lambda \models \neg\varphi$	iff not $\lambda \models \varphi$;
$\lambda \models \varphi \wedge \psi$	iff $\lambda \models \varphi$ and $\lambda \models \psi$.

Note that:

$$G\varphi \equiv \neg F \neg \varphi$$

$$F\varphi \equiv \neg G \neg \varphi$$

$$F\varphi \equiv \top\, U\, \varphi$$

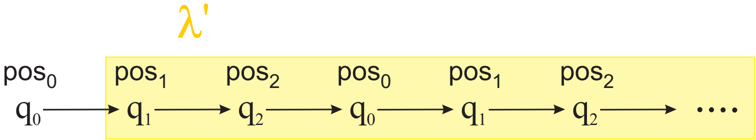
Semantics of LTL: $X\,\mathrm{pos}_1$

$$\lambda$$

$$\begin{array}{ccccccccccccc}
\mathrm{pos}_0 & & \mathrm{pos}_1 & & \mathrm{pos}_2 & & \mathrm{pos}_0 & & \mathrm{pos}_1 & & \mathrm{pos}_2 & & \\
q_0 & \longrightarrow & q_1 & \longrightarrow & q_2 & \longrightarrow & q_0 & \longrightarrow & q_1 & \longrightarrow & q_2 & \longrightarrow & \cdots
\end{array}$$

$$\lambda \models X\,\mathrm{pos}_1$$

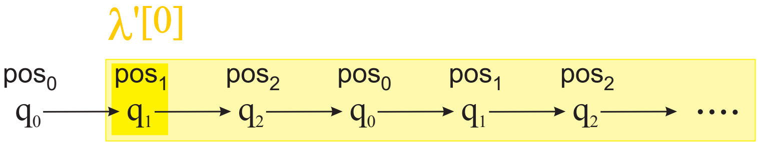
Semantics of LTL: $X\,\mathsf{pos}_1$



$$\lambda \models X\,\mathsf{pos}_1$$

$$\lambda' = \lambda[1..\infty] \models \mathsf{pos}_1$$

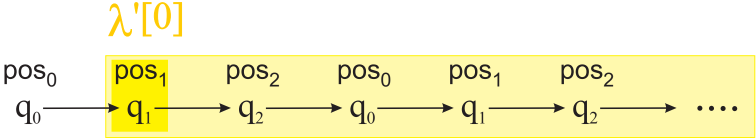
Semantics of LTL: $X\mathsf{pos}_1$



$$\lambda \models X\mathsf{pos}_1$$

$$\lambda' = \lambda[1..\infty] \models \mathsf{pos}_1$$
$$\lambda'[0] \in \mathcal{V}(\mathsf{pos}_1)$$

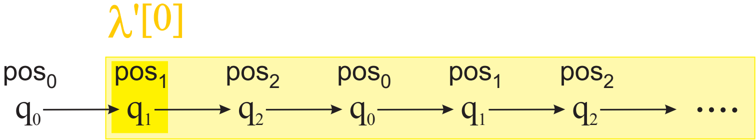
Semantics of LTL: $pos_0 \; U \; pos_1$



$$\lambda \models pos_0 \; U \; pos_1$$

$$\lambda' = \lambda[1..\infty] \models pos_1$$
$$\forall i < 1, \lambda[i..\infty] \models pos_0$$
$$\lambda'[0] \in \mathcal{V}(pos_1)$$
$$\lambda[0] \in \mathcal{V}(pos_0)$$

Semantics of LTL: $F\,\mathrm{pos}_1$



$$\lambda \models F\,\mathrm{pos}_1$$

$$\lambda' = \lambda[1..\infty] \models \mathrm{pos}_1$$
$$\lambda'[0] \in \mathcal{V}(\mathrm{pos}_1)$$

Semantics of LTL: $G\,F\,\mathrm{pos}_1$

λ

$$\begin{array}{ccccccccccccc}
\mathrm{pos}_0 & & \mathrm{pos}_1 & & \mathrm{pos}_2 & & \mathrm{pos}_0 & & \mathrm{pos}_1 & & \mathrm{pos}_2 & & \\
q_0 & \longrightarrow & q_1 & \longrightarrow & q_2 & \longrightarrow & q_0 & \longrightarrow & q_1 & \longrightarrow & q_2 & \longrightarrow & \cdots
\end{array}$$

$$\lambda \models G\,F\,\mathrm{pos}_1$$

Semantics of LTL: $G\,F\,\mathrm{pos}_1$

$\lambda[0..\infty]$

pos_0	pos_1	pos_2	pos_0	pos_1	pos_2	
$q_0 \longrightarrow$	$q_1 \longrightarrow$	$q_2 \longrightarrow$	$q_0 \longrightarrow$	$q_1 \longrightarrow$	$q_2 \longrightarrow$	$\cdots\cdot$

$$\lambda \models G\,F\,\mathrm{pos}_1$$

$$\lambda[0..\infty] \models F\,\mathrm{pos}_1$$

Semantics of LTL: $G\,F\,\mathrm{pos}_1$

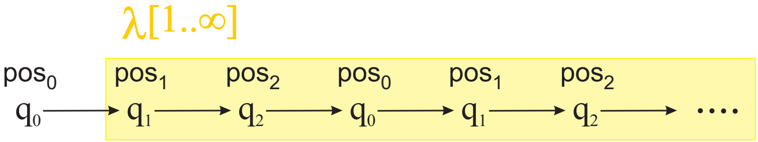
$\lambda[0..\infty]$

pos_0	pos_1	pos_2	pos_0	pos_1	pos_2	
$q_0 \longrightarrow$	$q_1 \longrightarrow$	$q_2 \longrightarrow$	$q_0 \longrightarrow$	$q_1 \longrightarrow$	$q_2 \longrightarrow$	$\cdots$

$$\lambda \models G\,F\,\mathrm{pos}_1$$

$$\lambda[0..\infty] \models F\,\mathrm{pos}_1$$

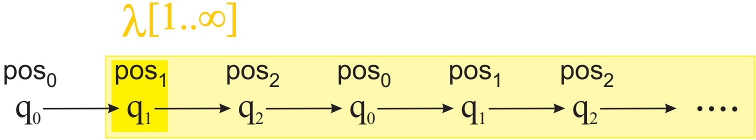
Semantics of LTL: $G\,F\,\mathrm{pos}_1$



$$\lambda \models G\,F\,\mathrm{pos}_1$$

$$\lambda[0..\infty] \models F\,\mathrm{pos}_1$$
$$\lambda[1..\infty] \models F\,\mathrm{pos}_1$$

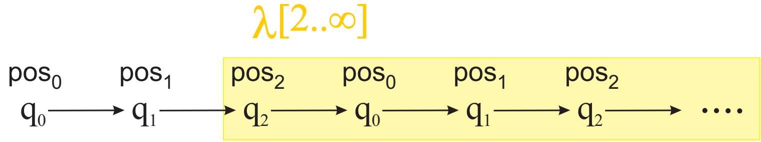
Semantics of LTL: $G\,F\,\text{pos}_1$



$$\lambda \models G\,F\,\text{pos}_1$$

$$\lambda[0..\infty] \models F\,\text{pos}_1$$
$$\lambda[1..\infty] \models F\,\text{pos}_1$$

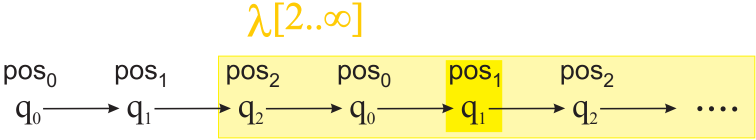
Semantics of LTL: $G\,F\,\mathrm{pos}_1$



$$\lambda \models G\,F\,\mathrm{pos}_1$$

$$\lambda[0..\infty] \models F\,\mathrm{pos}_1$$
$$\lambda[1..\infty] \models F\,\mathrm{pos}_1$$
$$\lambda[2..\infty] \models F\,\mathrm{pos}_1$$

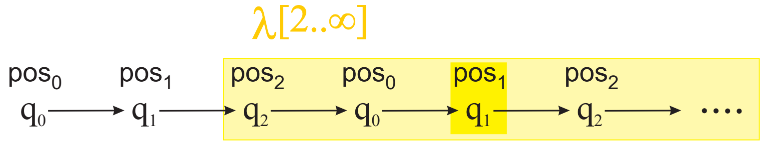
Semantics of LTL: $G\,F\,\mathrm{pos}_1$



$$\lambda \models G\,F\,\mathrm{pos}_1$$

$$\lambda[0..\infty] \models F\,\mathrm{pos}_1$$
$$\lambda[1..\infty] \models F\,\mathrm{pos}_1$$
$$\lambda[2..\infty] \models F\,\mathrm{pos}_1$$

Semantics of LTL: $G\,F\,\mathrm{pos}_1$



$$\lambda \models G\,F\,\mathrm{pos}_1$$

$$\lambda[0..\infty] \models F\,\mathrm{pos}_1$$
$$\lambda[1..\infty] \models F\,\mathrm{pos}_1$$
$$\lambda[2..\infty] \models F\,\mathrm{pos}_1$$
$$\ldots$$

Example: specifying reward functions

- generate a reward function for an RL agent from a temporal logic specification
- e.g., ‘bring gems to the shed, always avoid zombies’

from Camacho et al. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning. IJCAI 2019: 6065-6073

Minecraft: specification of behaviour

- E1 Collect wood and iron in any order, and use the factory afterwards
- E2 If it is night time, stay in the shed until daylight
- E3 Always avoid zombies
- E4 While there are gems on the ground, put them in your bag. When your bag is full, deliver the gems to the shed, and get an empty bag.

Minecraft: specification of behaviour

E1 Collect wood and iron in any order, and use the factory afterwards

$$F(\text{got\_wood} \wedge F\,\text{used\_factory}) \wedge F(\text{got\_iron} \wedge F\,\text{used\_factory})$$

E2 If it is night time, stay in the shed until daylight

$$G(\text{is\_night} \rightarrow \text{at\_shed})$$

E3 Always avoid zombies

$$G\,\neg\text{near\_zombie}$$

Minecraft: specification of behaviour 2

E4 While there are gems on the ground, put them in your bag. When your bag is full, deliver the gems to the shed, and get an empty bag.

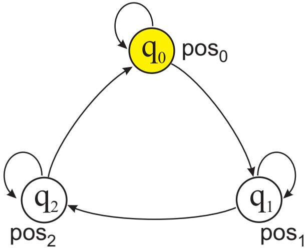
$$G(\neg \text{is\_night} \wedge \neg \text{near\_zombie} \rightarrow (\text{gems} \wedge \neg \text{bag\_full} \rightarrow X\,\text{got\_gem}) \wedge (\text{bag\_full} \rightarrow F(\text{at\_shed} \wedge \neg \text{bag\_full})))$$

Semantics of LTL in Computational Structures

Definition (Semantics of LTL in Transition Systems)

$M, q \models \varphi \quad$ iff $\lambda \models \varphi$ for every path λ in M starting from q.

Example: Robot and carriage



$M, q_0 \not\models G\,\mathrm{pos}_0$

(but also: $M, q_0 \not\models \neg G\,\mathrm{pos}_0$!)

$M, q_0 \not\models F\,\mathrm{pos}_1$

(but also: $M, q_0 \not\models \neg F\,\mathrm{pos}_1$!)

$M, q_0 \models (\neg G\,\mathrm{pos}_0) \rightarrow F\,\mathrm{pos}_1$

Verification using LTL

Definition (Model checking problem for LTL)

Given a finite state transition system M, a state q in M, and an LTL formula φ, check whether $M, q \models \varphi$.

Remember that we need to check whether $\lambda \models \varphi$ for every path λ in M starting from q.

Complexity of the LTL model checking problem

- LTL model checking is usually done using Büchi automata (automata over infinite strings)
- Given M, q and φ, two automata are constructed:
  - $\mathcal{A}_{\neg\varphi}$ that accepts all paths satisfying $\neg\varphi$
  - $\mathcal{A}_{M,q}$ that accepts all paths in M starting from q
- then a check is performed for whether the sets of paths accepted by $\mathcal{A}_{\neg\varphi}$ has a non-empty intersection with the set of paths accepted by $\mathcal{A}_{M,q}$

Complexity of the LTL model checking problem cont.

- the non-emptiness check is done by constructing a product automaton of $\mathcal{A}_{\neg\varphi}$ and $\mathcal{A}_{M,q}$ and checking non-emptyness of its language (the paths it accepts)
- the non-emptyness check can be done in linear time in the size of the product automaton
- $\mathcal{A}_{M,q}$ is of size linear in $|M|$, but unfortunately $\mathcal{A}_{\neg\varphi}$ is exponential in $|\varphi|$
- so the whole procedure is polynomial in $|M|$ but exponential in $|\varphi|$
- the problem itself is PSPACE-complete, so it is very unlikely one can do much better

What cannot be expressed in LTL?

- LTL can express very useful properties of agent behaviour
- any kind of regular expression pattern can be expressed in LTL
- however it cannot express that it is possible for an agent to have a choice between two or more actions
- sometimes we want to express that something may happen
- next week: branching time temporal logic

Reading for week 2

- Wojtek Jamroga, *Logical Methods for the Specification and Verification of Multiagent Systems*, Chapter 3.1.3, 3.1.4 (only the beginning; modal μ-calculus optional), 3.2.1.