University of Antwerp

# Specification and Verification

## Lecture 3: Linear temporal logic

Guillermo A. Pérez

October 7, 2024

# TL;DR: This lecture in short

## What is LTL? Why study it?

A logic to speak about linear-temporal properties

## Main references

- Christel Baier, Joost-Pieter Katoen: **Principles of Model Checking.** MIT Press 2018.
- Mickael Randour: Verification course @ UMONS.

# Required and target competences

## What tools do we need?

Discrete mathematics, Automata theory

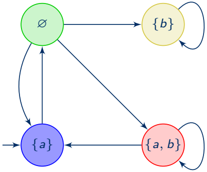
## What skills will we obtain?

- theory: a common language to speak about specifications
- practice: LTL and its variants are used for formal specifications in industry and in applied verification

## How will these skills be useful?

Having a common language will allow us to move forward in this course and will help you formalize what you want from systems you use and program.

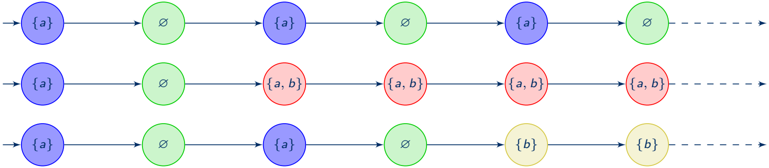
# Linear-time semantics: a reminder



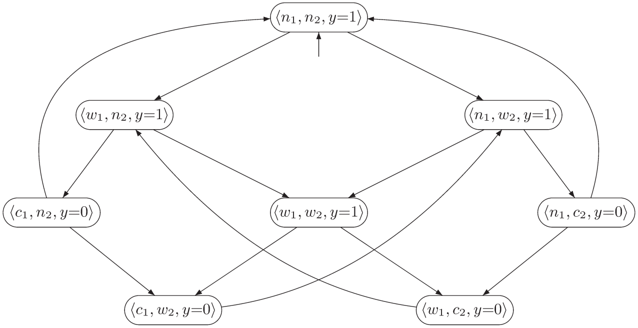

*TS $\mathcal{T}$ with atomic propositions $P = \{a, b\}$ (state and action names are omitted).*

From now on, we assume **no terminal state**

- **Linear-time semantics** deals with *traces* of executions.
  - The language of infinite words described by $\mathcal{T}$

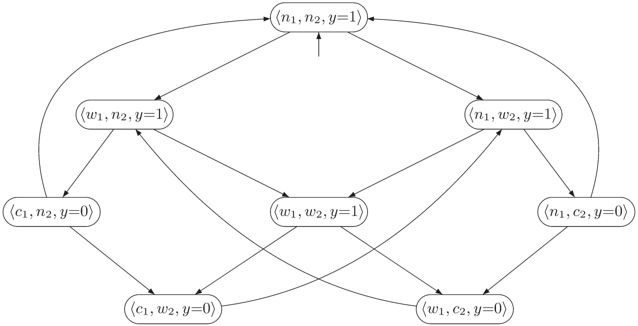


# LT properties: safety



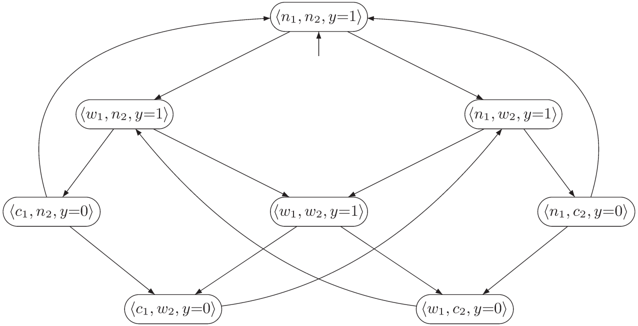

*TS for semaphore-based mutex*

# LT properties: safety




Ensure that $\langle c_1, c_2, y = \ldots\rangle \notin \mathrm{Reach}(\mathcal{T}(PG_1 \mid\mid\mid PG_2))$ or equivalently that $\neg\exists\, \pi \in \mathrm{Paths}(\mathcal{T}),\ \langle c_1, c_2, y = \ldots\rangle \in \pi$

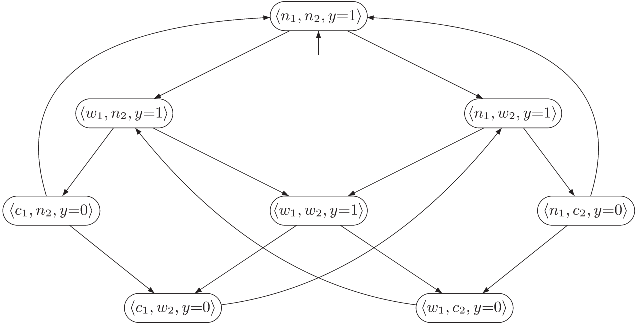
# LT properties: safety




Ensure that $\langle c_1, c_2, y = \dots\rangle \notin \mathrm{Reach}(\mathcal{T}(PG_1 \mid\mid\mid PG_2))$ or equivalently that $\neg\exists\, \pi \in \mathrm{Paths}(\mathcal{T}),\ \langle c_1, c_2, y = \dots\rangle \in \pi$

$\hookrightarrow$ **Satisfied**

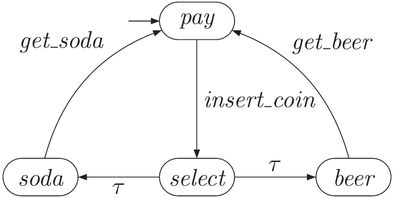
# LT properties: safety




For model checking, we like to use *labels* and *traces*

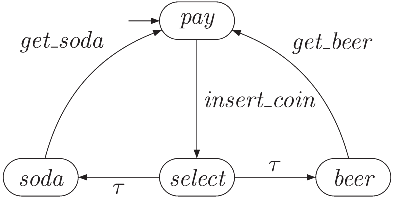
- $P = \{crit_1, crit_2\}$, natural labelling
- Ensure that $\neg\exists\, w \in \mathrm{Traces}(\mathcal{T}),\ \{crit_1, crit_2\} \in w$

# LT properties: liveness




*Beverage vending machine*

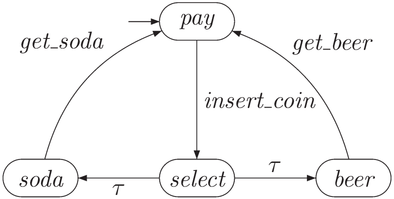
# LT properties: liveness




Ensure that the machine delivers a *drink* infinitely often.

- $P = \{paid, drink\}$, natural labelling
- $\forall\, w \in \mathrm{Traces}(\mathcal{T})$, **for all position** $i$ along $w$, label *drink* **must appear in the future**

# LT properties: liveness

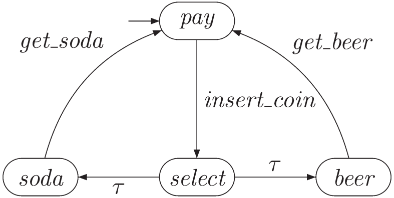



Ensure that the machine delivers a *drink* infinitely often.

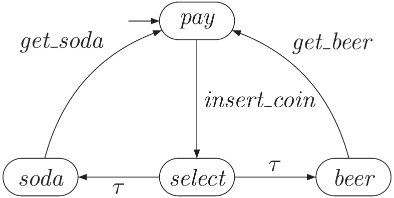
- $P = \{paid, drink\}$, natural labelling
- $\forall\, w \in \mathrm{Traces}(\mathcal{T})$, **for all position** $i$ along $w$, label *drink* **must appear in the future**

$\hookrightarrow$ **Satisfied:** recall we consider *infinite* executions

# LT properties: liveness



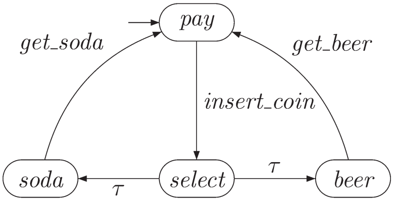
# LT properties: liveness




What if we ask that the machine delivers a *beer* infinitely often?

- $P = \{paid, soda, beer\}$, natural labelling
- $\forall\, w \in \mathrm{Traces}(\mathcal{T})$, for all positions $i$ along $w$, label *beer* must appear in the future

# LT properties: liveness




What if we ask that the machine delivers a *beer* infinitely often.

- $P = \{paid, soda, beer\}$, natural labelling
- $\forall\, w \in \mathrm{Traces}(\mathcal{T})$, for all positions $i$ along $w$, label *beer* must appear in the future

$\hookrightarrow$ **Not satisfied:** $w = (\varnothing\,\{paid\}\,\{paid, soda\})^\omega$

# LT properties: safety vs. liveness

Informally, safety means "something bad never happens"

$\Longrightarrow$ Can easily be satisfied by doing nothing!

# LT properties: safety vs. liveness

Informally, safety means "something bad never happens"

$\Longrightarrow$ Can easily be satisfied by doing nothing!

$\Longrightarrow$ Needs to be complemented with liveness, i.e., "something good **will** happen"

# LT properties: safety vs. liveness

Informally, safety means "something bad never happens"

$\Longrightarrow$ Can easily be satisfied by doing nothing!

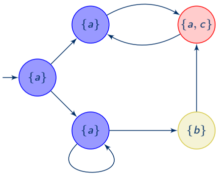
$\Longrightarrow$ Needs to be complemented with liveness, i.e., "something good **will** happen"

**Finite vs. infinite time**

Safety is violated by *finite* executions (i.e., the prefix up to seeing a bad state) whereas liveness is violated by *infinite* ones:

- witnessing that the good behavior never occurs.

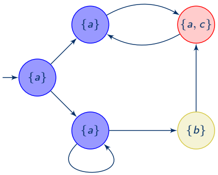
# LT properties: persistence




Ensure that a property **eventually** holds **forever**

- E.g., from some point on, $a$ holds but $b$ does not

# LT properties: persistence



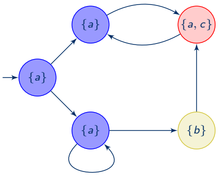

Ensure that a property **eventually** holds **forever**

- E.g., from some point on, $a$ holds but $b$ does not

$\hookrightarrow$ **Satisfied.** Indeed, $\mathrm{Traces}(\mathcal{T}) =$

$$\{a\}\ \left[\{a\}^\omega \,\middle|\, (\{a\}\,\{a,c\})^\omega \,\middle|\, \{a\}^+\,\{b\}\,(\{a,c\}\,\{a\})^\omega\right]$$

# LT properties: persistence



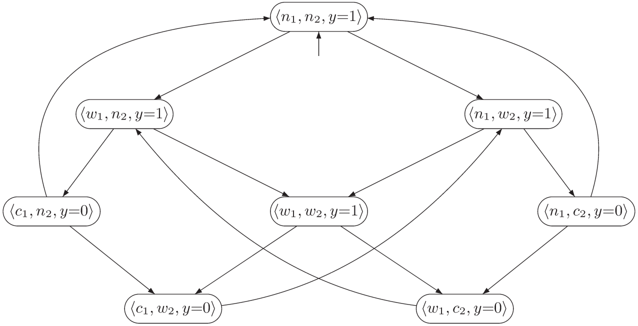

Ensure that a property **eventually** holds **forever**

- E.g., from some point on, $a$ holds but $b$ does not

$\hookrightarrow$ **Satisfied.** Indeed, $\mathrm{Traces}(\mathcal{T}) =$
$$\{a\}\left[\{a\}^{\omega} \;\middle|\; (\{a\}\,\{a,c\})^{\omega} \;\middle|\; \{a\}^{+}\,\{b\}\,(\{a,c\}\,\{a\})^{\omega}\right]$$

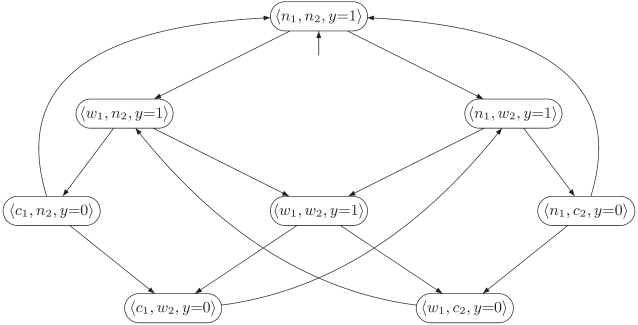
$\Longrightarrow$ **Ultimately periodic traces where $b$ is false and $a$ is true, at all steps after some point**

# LT properties: fairness (1/4)




*TS for semaphore-based mutex*

# LT properties: fairness (1/4)




*TS for semaphore-based mutex*

Ensure that both processes get *fair access* to the critical section

**What is fairness?**

# LT properties: fairness (2/4)

- **Unconditional fairness.** E.g., “every process gets access infinitely often”

# LT properties: fairness (2/4)

- **Unconditional fairness.** E.g., "every process gets access infinitely often"
- **Strong fairness.** E.g., "every process that requests access infinitely often gets access infinitely often"

# LT properties: fairness (2/4)

- **Unconditional fairness.** E.g., “every process gets access infinitely often”
- **Strong fairness.** E.g., “every process that requests access infinitely often gets access infinitely often”
- **Weak fairness.** E.g., “every process that continuously requests access from some point on gets access infinitely often”

# LT properties: fairness (2/4)

- **Unconditional fairness.** E.g., "every process gets access infinitely often"
- **Strong fairness.** E.g., "every process that requests access infinitely often gets access infinitely often"
- **Weak fairness.** E.g., "every process that continuously requests access from some point on gets access infinitely often"

**Unconditional $\Longrightarrow$ strong $\Longrightarrow$ weak**

**Converse not true in general**

# LT properties: fairness (2/4)

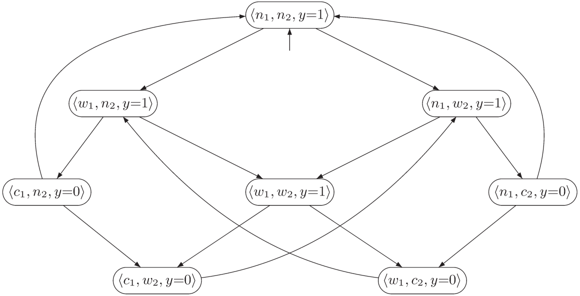
- **Unconditional fairness.** E.g., "every process gets access infinitely often"
- **Strong fairness.** E.g., "every process that requests access infinitely often gets access infinitely often"
- **Weak fairness.** E.g., "every process that continuously requests access from some point on gets access infinitely often"

**Unconditional $\Longrightarrow$ strong $\Longrightarrow$ weak**

**Converse not true in general**

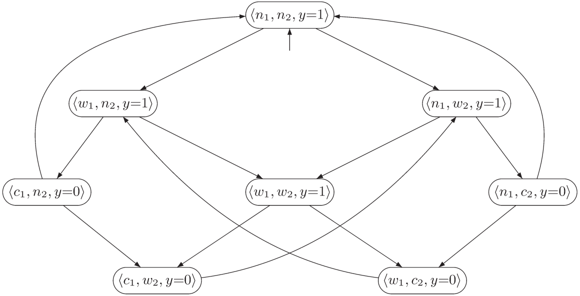
$\Longrightarrow$ **All forms can be formalized in LTL**

# LT properties: fairness (3/4)




The semaphore-based mutex is **not fair** in any sense

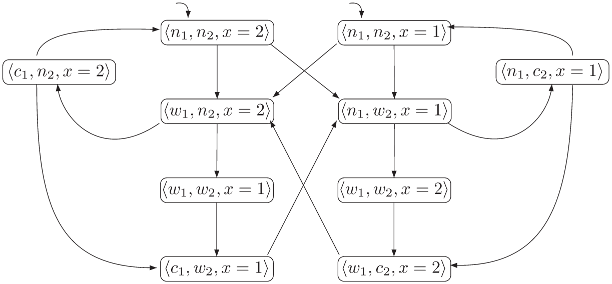
# LT properties: fairness (3/4)




The semaphore-based mutex is **not fair** in any sense We have seen that *starvation* is possible. E.g., execution

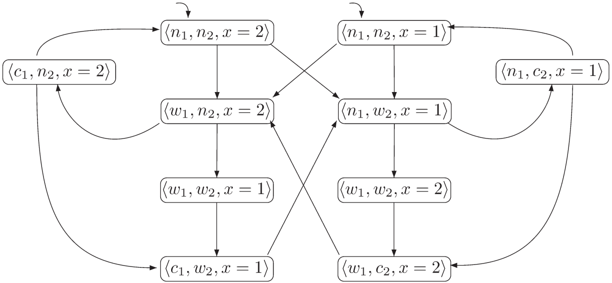
$$\langle n_1, n_2, y = 1\rangle$$
$$\longrightarrow (\langle w_1, n_2, y = 1\rangle \longrightarrow \langle w_1, w_2, y = 1\rangle \longrightarrow \langle w_1, c_2, y = 0\rangle)^\omega$$

# LT properties: fairness (4/4)



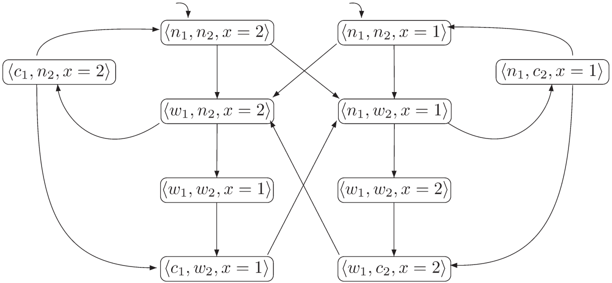

- Peterson's mutex is **strongly fair**

# LT properties: fairness (4/4)




- Peterson's mutex is **strongly fair**
- We saw that it has *bounded waiting*

# LT properties: fairness (4/4)




- Peterson's mutex is **strongly fair**
- We saw that it has *bounded waiting*
- A process requesting access waits at most one turn

$\hookrightarrow$ **Infinitely frequent requests $\Longrightarrow$ infinitely frequent access**

$\Longrightarrow$ **Strong fairness**

# Linear temporal logic

**LT property**

Essentially, a set of acceptable traces over $P$

# Linear temporal logic

**LT property**

Essentially, a set of acceptable traces over $P$

- Often difficult to describe explicitly
- Adequate formalism needed for model checking

# Linear temporal logic

**LT property**

Essentially, a set of acceptable traces over $P$

- Often difficult to describe explicitly
- Adequate formalism needed for model checking

$\Longrightarrow$ **Linear Temporal Logic (LTL):**

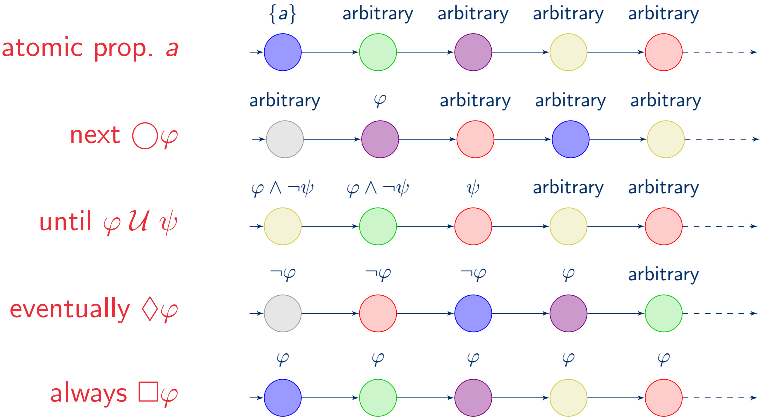
**propositional logic $+$ temporal operators**

# LTL in a nutshell

- **Atomic propositions** $a \in P$; **Boolean combinations of formulas**: $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$; **Temporal operators**:

# LTL in a nutshell

- **Atomic propositions** $a \in P$; **Boolean combinations of formulas**: $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$; **Temporal operators**:



# LTL syntax

## LTL syntax

Given the set of atomic propositions $P$, LTL formulas are formed according to the following grammar:

$$\varphi ::= \top \mid a \mid \varphi \wedge \psi \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi\, \mathcal{U}\, \psi$$

where $a \in P$.

# LTL syntax

## LTL syntax

Given the set of atomic propositions $P$, LTL formulas are formed according to the following grammar:

$$\varphi ::= \top \mid a \mid \varphi \wedge \psi \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi\,\mathcal{U}\,\psi$$

where $a \in P$.

**$\varphi\,\mathcal{U}\,\psi$ requires that $\psi$ holds at some point!**

**(i.e., $\varphi$ forever does not suffice)**

# LTL syntax: derived operators

$$\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$$

# LTL syntax: derived operators

$$\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$$

$$\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$$ *implication*

# LTL syntax: derived operators

$$\varphi \lor \psi \equiv \neg(\neg\varphi \land \neg\psi)$$

$$\varphi \rightarrow \psi \equiv \neg\varphi \lor \psi \qquad \text{*implication*}$$

$$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi) \qquad \text{*equivalence*}$$

# LTL syntax: derived operators

$$\varphi \lor \psi \equiv \neg(\neg\varphi \land \neg\psi)$$

$$\varphi \rightarrow \psi \equiv \neg\varphi \lor \psi$$ *implication*

$$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi)$$ *equivalence*

$$\varphi \oplus \psi \equiv (\varphi \land \neg\psi) \lor (\neg\varphi \land \psi)$$ *exclusive or*

# LTL syntax: derived operators

$$
\begin{aligned}
\varphi \lor \psi &\equiv \neg(\neg\varphi \land \neg\psi) \\
\varphi \rightarrow \psi &\equiv \neg\varphi \lor \psi && \text{*implication*} \\
\varphi \leftrightarrow \psi &\equiv (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi) && \text{*equivalence*} \\
\varphi \oplus \psi &\equiv (\varphi \land \neg\psi) \lor (\neg\varphi \land \psi) && \text{*exclusive or*} \\
\bot &\equiv \neg\top
\end{aligned}
$$

# LTL syntax: derived operators

$$\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$$

$$\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$$ *implication*

$$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$$ *equivalence*

$$\varphi \oplus \psi \equiv (\varphi \wedge \neg\psi) \vee (\neg\varphi \wedge \psi)$$ *exclusive or*

$$\bot \equiv \neg\top$$

$$\Diamond\varphi \equiv \top\,\mathcal{U}\,\varphi$$ ***eventually (or finally)***

# LTL syntax: derived operators

$$\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$$

$$\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$$ *implication*

$$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$$ *equivalence*

$$\varphi \oplus \psi \equiv (\varphi \wedge \neg\psi) \vee (\neg\varphi \wedge \psi)$$ *exclusive or*

$$\bot \equiv \neg\top$$

$$\Diamond\varphi \equiv \top\, \mathcal{U}\, \varphi$$ *eventually (or finally)*

$$\Box\varphi \equiv \neg\Diamond\neg\varphi$$ *always (or globally)*

# LTL syntax: derived operators

$$\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$$

$$\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$$ *implication*

$$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$$ *equivalence*

$$\varphi \oplus \psi \equiv (\varphi \wedge \neg\psi) \vee (\neg\varphi \wedge \psi)$$ *exclusive or*

$$\bot \equiv \neg\top$$

$$\Diamond\varphi \equiv \top\, \mathcal{U}\, \varphi$$ **eventually (or finally)**

$$\Box\varphi \equiv \neg\Diamond\neg\varphi$$ **always (or globally)**

$$\varphi\, \mathcal{W}\, \psi \equiv (\varphi\, \mathcal{U}\, \psi) \vee \Box\varphi$$ **weak until**

- Weak until $\leadsto$ until that does not require $\psi$ to be reached

University of Antwerp

# LTL syntax: derived operators

$$\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$$

$$\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$$ *implication*

$$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$$ *equivalence*

$$\varphi \oplus \psi \equiv (\varphi \wedge \neg\psi) \vee (\neg\varphi \wedge \psi)$$ *exclusive or*

$$\bot \equiv \neg\top$$

$$\Diamond\varphi \equiv \top\,\mathcal{U}\,\varphi$$ **eventually (or finally)**

$$\Box\varphi \equiv \neg\Diamond\neg\varphi$$ **always (or globally)**

$$\varphi\,\mathcal{W}\,\psi \equiv (\varphi\,\mathcal{U}\,\psi) \vee \Box\varphi$$ **weak until**

$$\varphi\,\mathcal{R}\,\psi \equiv \neg(\neg\varphi\,\mathcal{U}\,\neg\psi)$$ **release**

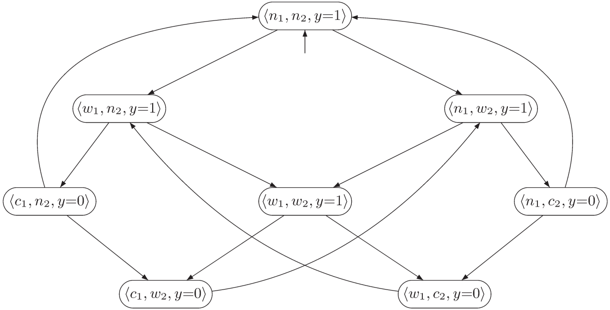
- Weak until $\rightsquigarrow$ until that does not require $\psi$ to be reached
- Release $\rightsquigarrow$ $\psi$ must hold up to the point where $\varphi$ releases it, or forever if $\varphi$ never holds

# LTL syntax: precedence order

## Precedence order

- Unary operators before binary ones,
- $\neg$ and $\bigcirc$ equally strong,
- $\mathcal{U}$ before $\wedge$, $\vee$ and $\rightarrow$

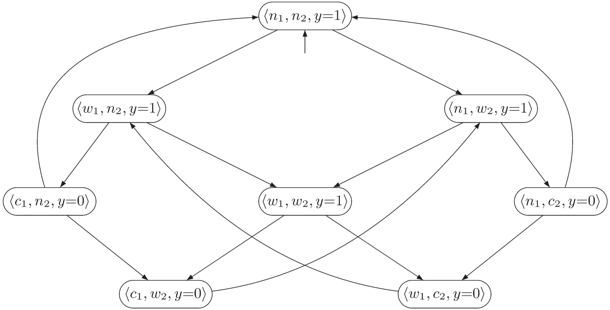
# LT properties in LTL: safety




*TS for semaphore-based mutex*

- $P = \{crit_1, crit_2\}$, natural labelling
- Ensure that $\neg\exists\, w \in \mathrm{Traces}(\mathcal{T}),\ \{crit_1, crit_2\} \in w$

# LT properties in LTL: safety

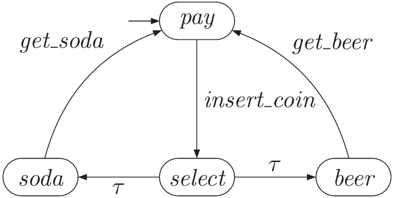



*TS for semaphore-based mutex*

- $P = \{crit_1, crit_2\}$, natural labelling
- Ensure that $\neg\exists\, w \in \mathrm{Traces}(\mathcal{T}),\ \{crit_1, crit_2\} \in w$

$\hookrightarrow$ $\neg\Diamond\,(crit_1 \wedge crit_2)$ or equivalently $\Box\,(\neg crit_1 \vee \neg crit_2)$

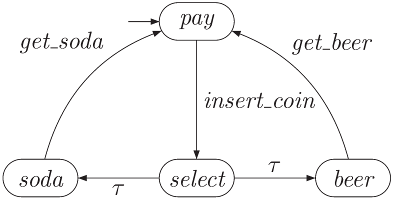
# LT properties in LTL: liveness




*Beverage vending machine*

- $P = \{paid, drink\}$, natural labelling
- $\forall\, w \in \mathrm{Traces}(\mathcal{T})$, for all positions $i$ along $w$, label *drink* must appear in the future

# LT properties in LTL: liveness

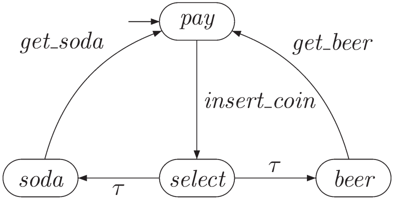



*Beverage vending machine*

- $P = \{paid, drink\}$, natural labelling
- $\forall\, w \in \mathrm{Traces}(\mathcal{T})$, for all positions $i$ along $w$, label *drink* must appear in the future

$\hookrightarrow$ $\Box\Diamond drink$

# LT properties in LTL: liveness



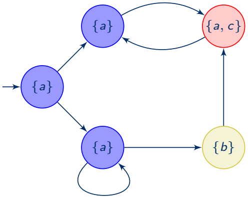

*Beverage vending machine*

- $P = \{paid, drink\}$, natural labelling
- $\forall\, w \in \mathrm{Traces}(\mathcal{T})$, for all positions $i$ along $w$, label *drink* must appear in the future

$\hookrightarrow$ $\Box\Diamond drink$

$\Longrightarrow$ **"infinitely often"**

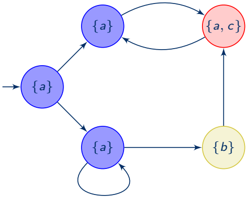
# LT properties in LTL: persistence




Ensure that a property eventually holds forever

- E.g., from some point on, $a$ holds but $b$ does not

# LT properties in LTL: persistence

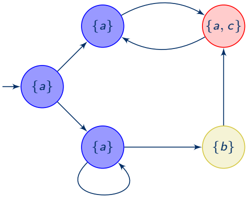



Ensure that a property eventually holds forever

- E.g., from some point on, $a$ holds but $b$ does not

$\hookrightarrow$ $\Diamond\Box(a \land \neg b)$

# LT properties in LTL: persistence




Ensure that a property eventually holds forever

- E.g., from some point on, $a$ holds but $b$ does not

$\hookrightarrow$ $\Diamond\Box(a \land \neg b)$

$\Longrightarrow$ **"eventually always"**

University of Antwerp

# LT properties in LTL: fairness

Assume $k$ processes and $P = \{wait_1, \ldots, wait_k, crit_1, \ldots, crit_k\}$

- **Unconditional fairness.** E.g., "every process gets access infinitely often"

# LT properties in LTL: fairness

Assume $k$ processes and $P = \{wait_1, \ldots, wait_k, crit_1, \ldots, crit_k\}$

- **Unconditional fairness.** E.g., "every process gets access infinitely often"

  $\hookrightarrow \bigwedge_{1 \leq i \leq k} \Box\Diamond crit_i$

# LT properties in LTL: fairness

Assume $k$ processes and $P = \{wait_1, \ldots, wait_k, crit_1, \ldots, crit_k\}$

- **Unconditional fairness.** E.g., "every process gets access infinitely often"

  $\hookrightarrow \bigwedge_{1 \leq i \leq k} \Box\Diamond crit_i$

- **Strong fairness.** E.g., "every process that requests access infinitely often gets access infinitely often"

# LT properties in LTL: fairness

Assume $k$ processes and $P = \{wait_1, \ldots, wait_k, crit_1, \ldots, crit_k\}$

- **Unconditional fairness.** E.g., "every process gets access infinitely often"

  $\hookrightarrow \bigwedge_{1 \leq i \leq k} \Box\Diamond crit_i$

- **Strong fairness.** E.g., "every process that requests access infinitely often gets access infinitely often"

  $\hookrightarrow \bigwedge_{1 \leq i \leq k} (\Box\Diamond wait_i \rightarrow \Box\Diamond crit_i)$

# LT properties in LTL: fairness

Assume $k$ processes and $P = \{wait_1, \ldots, wait_k, crit_1, \ldots, crit_k\}$

- **Unconditional fairness.** E.g., "every process gets access infinitely often"

  $\hookrightarrow$ $\bigwedge_{1 \leq i \leq k} \Box\Diamond crit_i$

- **Strong fairness.** E.g., "every process that requests access infinitely often gets access infinitely often"

  $\hookrightarrow$ $\bigwedge_{1 \leq i \leq k} (\Box\Diamond wait_i \rightarrow \Box\Diamond crit_i)$

- **Weak fairness.** E.g., "every process that continuously requests access from some point on gets access infinitely often"

# LT properties in LTL: fairness

Assume $k$ processes and $P = \{wait_1, \ldots, wait_k, crit_1, \ldots, crit_k\}$

- **Unconditional fairness.** E.g., "every process gets access infinitely often"

  $\hookrightarrow \bigwedge_{1 \leq i \leq k} \Box\Diamond crit_i$

- **Strong fairness.** E.g., "every process that requests access infinitely often gets access infinitely often"

  $\hookrightarrow \bigwedge_{1 \leq i \leq k} (\Box\Diamond wait_i \to \Box\Diamond crit_i)$

- **Weak fairness.** E.g., "every process that continuously requests access from some point on gets access infinitely often"

  $\hookrightarrow \bigwedge_{1 \leq i \leq k} (\Diamond\Box wait_i \to \Box\Diamond crit_i)$

# LTL semantics: over words (1/2)

Given propositions $P$ and LTL formula $\varphi$, the associated LT property is the language of words:

$$\mathrm{Words}(\varphi) = \left\{ w = a_0 a_1 a_2 \ldots \in (2^P)^\omega \;\middle|\; w \models \varphi \right\}$$

where $\models$ is the smallest relation satisfying:

# LTL semantics: over words (1/2)

Given propositions $P$ and LTL formula $\varphi$, the associated LT property is the language of words:

$$\mathrm{Words}(\varphi) = \left\{ w = a_0 a_1 a_2 \ldots \in (2^P)^\omega \;\middle|\; w \models \varphi \right\}$$

where $\models$ is the smallest relation satisfying:

$$w \models \top$$

# LTL semantics: over words (1/2)

Given propositions $P$ and LTL formula $\varphi$, the associated LT property is the language of words:

$$\mathrm{Words}(\varphi) = \left\{ w = a_0 a_1 a_2 \ldots \in (2^P)^\omega \;\middle|\; w \models \varphi \right\}$$

where $\models$ is the smallest relation satisfying:

$$
\begin{array}{lll}
w \models \top & & \textit{Recall letters are subsets of } P \\
w \models a & \text{iff} & a \in a_0
\end{array}
$$

# LTL semantics: over words (1/2)

Given propositions $P$ and LTL formula $\varphi$, the associated LT property is the language of words:

$$\mathrm{Words}(\varphi) = \left\{ w = a_0 a_1 a_2 \ldots \in (2^P)^\omega \;\middle|\; w \models \varphi \right\}$$

where $\models$ is the smallest relation satisfying:

| | | |
|---|---|---|
| $w \models \top$ | | *Recall letters are subsets of $P$* |
| $w \models a$ | iff | $a \in a_0$ |
| $w \models \varphi \wedge \psi$ | iff | $w \models \varphi$ and $w \models \psi$ |

# LTL semantics: over words (1/2)

Given propositions $P$ and LTL formula $\varphi$, the associated LT property is the language of words:

$$\mathrm{Words}(\varphi) = \left\{ w = a_0 a_1 a_2 \ldots \in (2^P)^\omega \;\middle|\; w \models \varphi \right\}$$

where $\models$ is the smallest relation satisfying:

$$
\begin{array}{lll}
w \models \top & & \textit{Recall letters are subsets of } P \\
w \models a & \text{iff} & a \in a_0 \\
w \models \varphi \wedge \psi & \text{iff} & w \models \varphi \text{ and } w \models \psi \\
w \models \neg\varphi & \text{iff} & w \not\models \varphi
\end{array}
$$

# LTL semantics: over words (1/2)

Given propositions $P$ and LTL formula $\varphi$, the associated LT property is the language of words:

$$\mathrm{Words}(\varphi) = \left\{ w = a_0 a_1 a_2 \ldots \in (2^P)^\omega \;\middle|\; w \models \varphi \right\}$$

where $\models$ is the smallest relation satisfying:

$$
\begin{array}{lll}
w \models \top & & \textit{Recall letters are subsets of } P \\
w \models a & \text{iff} & a \in a_0 \\
w \models \varphi \wedge \psi & \text{iff} & w \models \varphi \text{ and } w \models \psi \\
w \models \neg\varphi & \text{iff} & w \not\models \varphi \\
w \models \bigcirc\varphi & \text{iff} & w[1..] = a_1 a_2 \ldots \models \varphi
\end{array}
$$

# LTL semantics: over words (1/2)

Given propositions $P$ and LTL formula $\varphi$, the associated LT property is the language of words:

$$\mathrm{Words}(\varphi) = \left\{ w = a_0 a_1 a_2 \ldots \in (2^P)^\omega \;\middle|\; w \models \varphi \right\}$$

where $\models$ is the smallest relation satisfying:

| | | |
|---|---|---|
| $w \models \top$ | | *Recall letters are subsets of $P$* |
| $w \models a$ | iff | $a \in a_0$ |
| $w \models \varphi \wedge \psi$ | iff | $w \models \varphi$ and $w \models \psi$ |
| $w \models \neg\varphi$ | iff | $w \not\models \varphi$ |
| $w \models \bigcirc\varphi$ | iff | $w[1..] = a_1 a_2 \ldots \models \varphi$ |
| $w \models \varphi \,\mathcal{U}\, \psi$ | iff | $\exists j \geq 0,\ w[j..] \models \psi$ and $\forall 0 \leq i < j,\ w[i..] \models \varphi$ |

# LTL semantics: over words (2/2)

Other common operators:

# LTL semantics: over words (2/2)

Other common operators:

$$w \models \Diamond\varphi \qquad \text{iff} \quad \exists j \geq 0,\ w[j..] \models \varphi$$

University of Antwerp

# LTL semantics: over words (2/2)

Other common operators:

$$
\begin{aligned}
w &\models \Diamond\varphi && \text{iff} \quad \exists j \geq 0,\ w[j..] \models \varphi \\
w &\models \Box\varphi && \text{iff} \quad \forall j \geq 0,\ w[j..] \models \varphi
\end{aligned}
$$

# LTL semantics: over words (2/2)

Other common operators:

$$
\begin{aligned}
w &\models \Diamond\varphi && \text{iff} \quad \exists j \geq 0,\ w[j..] \models \varphi \\
w &\models \Box\varphi && \text{iff} \quad \forall j \geq 0,\ w[j..] \models \varphi \\
w &\models \Box\Diamond\varphi && \text{iff} \quad \forall j \geq 0,\ \exists i \geq j,\ w[i..] \models \varphi
\end{aligned}
$$

# LTL semantics: over words (2/2)

Other common operators:

$$
\begin{array}{lll}
w \models \Diamond\varphi & \text{iff} & \exists j \geq 0,\ w[j..] \models \varphi \\
w \models \Box\varphi & \text{iff} & \forall j \geq 0,\ w[j..] \models \varphi \\
w \models \Box\Diamond\varphi & \text{iff} & \forall j \geq 0,\ \exists i \geq j,\ w[i..] \models \varphi \\
w \models \Diamond\Box\varphi & \text{iff} & \exists j \geq 0,\ \forall i \geq j,\ w[i..] \models \varphi
\end{array}
$$

# LTL semantics: over transition systems

Let $\mathcal{T} = (S, A, \longrightarrow, I, P, L)$ be a TS and $\varphi$ an LTL formula over $P$.

# LTL semantics: over transition systems

Let $\mathcal{T} = (S, A, \longrightarrow, I, P, L)$ be a TS and $\varphi$ an LTL formula over $P$.

- For $\pi \in \mathrm{Paths}(\mathcal{T})$, $\pi \models \varphi$ iff $\mathrm{trace}(\pi) \models \varphi$

# LTL semantics: over transition systems

Let $\mathcal{T} = (S, A, \longrightarrow, I, P, L)$ be a TS and $\varphi$ an LTL formula over $P$.

- For $\pi \in \mathrm{Paths}(\mathcal{T})$, $\pi \models \varphi$ iff $\mathrm{trace}(\pi) \models \varphi$
- For $s \in S$, $s \models \varphi$ iff $\forall \pi \in \mathrm{Paths}(s)$, $\pi \models \varphi$

# LTL semantics: over transition systems

Let $\mathcal{T} = (S, A, \longrightarrow, I, P, L)$ be a TS and $\varphi$ an LTL formula over $P$.

- For $\pi \in \mathrm{Paths}(\mathcal{T})$, $\pi \models \varphi$ iff $\mathrm{trace}(\pi) \models \varphi$
- For $s \in S$, $s \models \varphi$ iff $\forall \pi \in \mathrm{Paths}(s)$, $\pi \models \varphi$
- TS $\mathcal{T}$ satisfies $\varphi$, denoted $\mathcal{T} \models \varphi$ iff $\mathrm{Traces}(\mathcal{T}) \subseteq \mathrm{Words}(\varphi)$

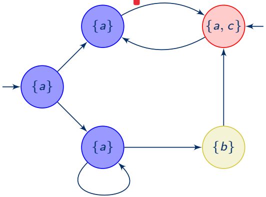
# LTL semantics: over transition systems

Let $\mathcal{T} = (S, A, \longrightarrow, I, P, L)$ be a TS and $\varphi$ an LTL formula over $P$.

- For $\pi \in \mathrm{Paths}(\mathcal{T})$, $\pi \models \varphi$ iff $\mathrm{trace}(\pi) \models \varphi$
- For $s \in S$, $s \models \varphi$ iff $\forall\, \pi \in \mathrm{Paths}(s)$, $\pi \models \varphi$
- TS $\mathcal{T}$ satisfies $\varphi$, denoted $\mathcal{T} \models \varphi$ iff $\mathrm{Traces}(\mathcal{T}) \subseteq \mathrm{Words}(\varphi)$

It follows that $\mathcal{T} \models \varphi$ iff $\forall\, s_0 \in I$, $s_0 \models \varphi$.

# Example




Notice the added initial state

$\mathcal{T} \not\models \Box a$

$\mathcal{T} \not\models \Diamond b$

$\mathcal{T} \models a \,\mathcal{W}\, b$

$\mathcal{T} \models \Box(b \rightarrow \Box\Diamond c)$

$\mathcal{T} \models \Diamond\Box a$

$\mathcal{T} \not\models a \,\mathcal{U}\, b$

$\mathcal{T} \not\models b \,\mathcal{R}\, a$

$\mathcal{T} \models b \rightarrow \Box c$

$\mathcal{T} \models \bigcirc(a \wedge \neg c)$

$\mathcal{T} \models \Box(c \rightarrow \bigcirc a)$

$\mathcal{T} \models \Box\neg c \rightarrow \neg\Diamond b$

$\mathcal{T} \not\models \bigcirc\bigcirc(b \vee c) \vee \Box a$

# Semantics of negation: paths

## Negation for paths

For $\pi \in \mathrm{Paths}(\mathcal{T})$ and an LTL formula $\varphi$ over $P$,

$$\pi \not\models \varphi \iff \pi \models \neg\varphi$$

because $\mathrm{Words}(\neg\varphi) = (2^P)^\omega \setminus \mathrm{Words}(\varphi)$.

# Semantics of negation: transition systems

## Negation for TSs

For TS $\mathcal{T} = (S, A, \longrightarrow, I, P, L)$ and an LTL formula $\varphi$ over $P$:

$$\begin{array}{c} \mathcal{T} \not\models \varphi \\ \not\Downarrow \ \Uparrow \\ \mathcal{T} \models \neg\varphi \end{array}$$

# Semantics of negation: transition systems

## Negation for TSs

For TS $\mathcal{T} = (S, A, \longrightarrow, I, P, L)$ and an LTL formula $\varphi$ over $P$:

$$\begin{array}{c} \mathcal{T} \not\models \varphi \\ \not\Downarrow \; \Uparrow \\ \mathcal{T} \models \neg\varphi \end{array}$$

$$\begin{array}{lll} \text{We have that } \mathcal{T} \not\models \varphi & \text{iff} & \mathrm{Traces}(\mathcal{T}) \not\subseteq \mathrm{Words}(\varphi) \\ & \text{iff} & \mathrm{Traces}(\mathcal{T}) \setminus \mathrm{Words}(\varphi) \neq \varnothing \\ & \text{iff} & \mathrm{Traces}(\mathcal{T}) \cap \mathrm{Words}(\neg\varphi) \neq \varnothing \end{array}$$

University of Antwerp

# Semantics of negation: transition systems

## Negation for TSs

For TS $\mathcal{T} = (S, A, \longrightarrow, I, P, L)$ and an LTL formula $\varphi$ over $P$:

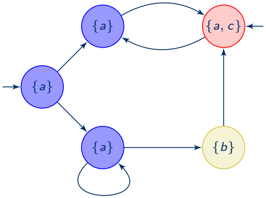
$$\begin{array}{c} \mathcal{T} \not\models \varphi \\ \not\Downarrow \Uparrow \\ \mathcal{T} \models \neg\varphi \end{array}$$

$$\begin{aligned} \text{We have that } \mathcal{T} \not\models \varphi \quad & \text{iff} \quad \mathrm{Traces}(\mathcal{T}) \not\subseteq \mathrm{Words}(\varphi) \\ & \text{iff} \quad \mathrm{Traces}(\mathcal{T}) \setminus \mathrm{Words}(\varphi) \neq \varnothing \\ & \text{iff} \quad \mathrm{Traces}(\mathcal{T}) \cap \mathrm{Words}(\neg\varphi) \neq \varnothing \end{aligned}$$

But it may be the case that $\mathcal{T} \not\models \varphi$ and $\mathcal{T} \not\models \neg\varphi$ if

$$\mathrm{Traces}(\mathcal{T}) \cap \mathrm{Words}(\neg\varphi) \neq \varnothing \text{ and } \mathrm{Traces}(\mathcal{T}) \cap \mathrm{Words}(\varphi) \neq \varnothing.$$

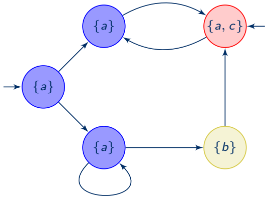
# Semantics of negation: example




We saw that $\mathcal{T} \not\models \Diamond b$

Do we have $\mathcal{T} \models \neg\Diamond b \equiv \Box\neg b$?

# Semantics of negation: example




We saw that $\mathcal{T} \not\models \Diamond b$

Do we have $\mathcal{T} \models \neg\Diamond b \equiv \Box\neg b$?

$\Longrightarrow$ **No.** Because trace $w = \{a\}^2\{b\}(\{a, c\}\{a\})^\omega$ satisfies $\Diamond b$

# Equivalence of LTL formulas: definition

### Equivalence of LTL formulas

LTL formulas $\varphi$ and $\psi$ are *equivalent*, denoted $\varphi \equiv \psi$, if

$$\mathrm{Words}(\varphi) = \mathrm{Words}(\psi).$$

# Back to fairness constraints with LTL

Let $\varphi, \psi$ be LTL formulas representing that "something is enabled" ($\varphi$) and that "something is granted" ($\psi$). Recall the three types of fairness.

- *Unconditional* fairness constraint

$$\textit{ufair} = \Box\Diamond\psi$$

- *Strong* fairness constraint

$$\textit{sfair} = \Box\Diamond\varphi \rightarrow \Box\Diamond\psi$$

- *Weak* fairness constraint

$$\textit{wfair} = \Diamond\Box\varphi \rightarrow \Box\Diamond\psi$$

# Fairness assumptions

Let *fair* denote a conjunction of such assumptions. It is sometimes useful to check that all **fair executions** of a TS satisfy a formula (in contrast to all of them).

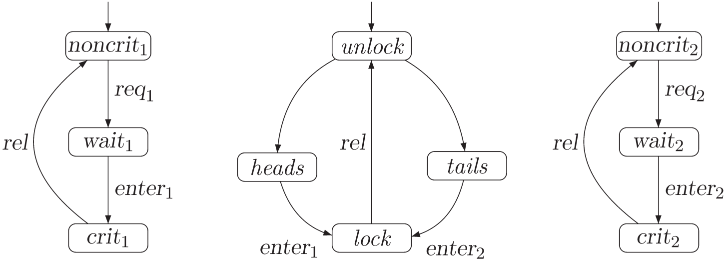
**Fair satisfaction**

Let $\varphi$ be an LTL formula and *fair* an LTL fairness assumption. We have that $\mathcal{T} \models_{fair} \varphi$ iff

$$\forall\, w \in \mathrm{Traces}(\mathcal{T}) \text{ such that } w \models \mathit{fair},\ w \models \varphi.$$

University of Antwerp

# Example: randomized arbiter for mutex




*Mutual exclusion with a randomized arbiter*

The arbiter chooses who gets access by tossing a coin: probabilities are abstracted by non-determinism.

Can process 1 access the section infinitely often?

# Example: randomized arbiter

$\hookrightarrow$ **No**, $\mathcal{T}_1 \mid\mid\mid \mathit{Arbiter} \mid\mid\mid \mathcal{T}_2 \not\models \Box\Diamond \mathit{req}_1 \rightarrow \Box\Diamond \mathit{crit}_1$ because the arbiter can always choose *tails*.

Intuitively, this is *unfair*: a real coin would lead to this with probability zero.

$\Longrightarrow$ LTL fairness assumption: $\Box\Diamond \mathit{heads} \wedge \Box\Diamond \mathit{tails}$.

$\hookrightarrow$ **The property is verified on fair executions**, i.e.,
$\mathcal{T}_1 \mid\mid\mid \mathit{Arbiter} \mid\mid\mid \mathcal{T}_2 \models_{\mathit{fair}} \bigwedge_{i \in \{1,2\}} (\Box\Diamond \mathit{req}_i \rightarrow \Box\Diamond \mathit{crit}_i)$.

# Handling fairness assumptions

Given a formula $\varphi$ and a fairness assumption *fair*, we can reduce $\models_{fair}$ to the classical satisfaction $\models$.

**From $\models_{\mathbf{fair}}$ to $\models$**

$$\mathcal{T} \models_{fair} \varphi \iff \mathcal{T} \models (fair \rightarrow \varphi).$$

# Summary and conclusions

## Linear temporal logic

Three very important things learned today:

- We now have a formal language to specify properties about systems: LTL
- We know how LTL formulas are interpreted over words
- We know how LTL formulas are interpreted over transition systems

## Moving forward

- Given a transition system and a formula, are there algorithms to determine whether the system satisfies the formula?