# Outline





# Static and dynamic properties

**Example: Static properties**

Mutual exclusion

Safety properties are often static.

They can be reduced to reachability.

**Example: Dynamic properties**

Every elevator request should be eventually granted.

The elevator should not cross a level for which a call is pending without stopping.



# Temporal Structures

**Definition: Flows of time**

A *flow of time* is a strict order $(\mathbb{T}, <)$ where $\mathbb{T}$ is the nonempty set of *time points* and $<$ is an irreflexive transitive relation on $\mathbb{T}$.

**Example: Flows of time**

- $(\{0, \ldots, n\}, <)$: Finite runs of sequential systems.
- $(\mathbb{N}, <)$: Infinite runs of sequential systems.
- $(\mathbb{R}, <)$: runs of real-time sequential systems.
- Trees: Finite or infinite run-trees of sequential systems.
- Mazurkiewicz traces: runs of distributed systems (partial orders).
- and also $(\mathbb{Z}, <)$ or $(\mathbb{Q}, <)$ or $(\omega^2, <)$, ...

**Definition: Temporal Structures**

Let $\mathrm{AP}$ be a set of atoms (atomic propositions).

A *temporal structure* over a class $\mathcal{C}$ of time flows and $\mathrm{AP}$ is a triple $(\mathbb{T}, <, h)$ where $(\mathbb{T}, <)$ is a time flow in $\mathcal{C}$ and $h : \mathrm{AP} \to 2^{\mathbb{T}}$ is an assignment.

If $p \in \mathrm{AP}$ then $h(p) \subseteq \mathbb{T}$ gives the time points where $p$ holds.



# Linear behaviors and specifications

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure.

**Definition: Runs as temporal structures**

An infinite run $\sigma = s_0 s_1 s_2 \cdots$ of $M$ with $(s_i, s_{i+1}) \in T$ for all $i \geq 0$ defines a *linear* temporal structure $\ell(\sigma) = (\mathbb{N}, <, h)$ where $h(p) = \{i \in \mathbb{N} \mid p \in \ell(s_i)\}$.

Such a temporal structure can be seen as an infinite word over $\Sigma = 2^{\mathrm{AP}}$:
$\ell(\sigma) = \ell(s_0)\ell(s_1)\ell(s_2) \cdots = (\mathbb{N}, <, w)$ with $w(i) = \ell(s_i) \in \Sigma$.

Linear specifications only depend on runs.

Example: The printer manager is fair.

On each run, whenever some process requests the printer, it eventually gets it.

**Remark:**

Two Kripke structures having the same linear temporal structures satisfy the same linear specifications.

# Branching behaviors and specifications

The system has an infinite active run, but it may always reach an inactive state.

**Definition: Computation-tree or run-tree : unfolding of the TS**

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure. Wlog. $I = \{s_0\}$ is a singleton.
Let $D$ be a finite set with $|D|$ the outdegree of the transition relation $T$.
The computation-tree of $M$ is an unordered tree $t : D^* \to S$ (partial map) s.t.

- $t(\varepsilon) = s_0$,
- For every node $u \in \mathrm{dom}(t)$ labelled $s = t(u)$, if $T(s) = \{s_1, \ldots, s_k\}$ then $u$ has exactly $k$ children which are labelled $s_1, \ldots, s_k$

Associated temporal structure $\ell(t) = (\mathrm{dom}(t), <, h)$ where

- $<$ is the strict prefix relation over $D^*$,
- and $h(p) = \{u \in \mathrm{dom}(t) \mid p \in \ell(t(u))\}$.

(Linear) runs of $M$ are branches of the computation-tree $t$.



# First-order Specifications

**Definition: Syntax of $\mathrm{FO}(\mathrm{AP}, <)$**

Let $\mathrm{Var} = \{x, y, \ldots\}$ be first-order variables.

$$\varphi ::= \bot \mid p(x) \mid x = y \mid x < y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\, \varphi$$

where $p \in \mathrm{AP}$.

**Definition: Semantics of $\mathrm{FO}(\mathrm{AP}, <)$**

Let $w = (\mathbb{T}, <, h)$ be a temporal structure over AP.
Let $\nu : \mathrm{Var} \to \mathbb{T}$ be an assignment of first-order variables to time points.

$$\begin{array}{lll} w, \nu \models p(x) & \text{if} & \nu(x) \in h(p) \\ w, \nu \models x = y & \text{if} & \nu(x) = \nu(y) \\ w, \nu \models x < y & \text{if} & \nu(x) < \nu(y) \\ w, \nu \models \exists x\, \varphi & \text{if} & w, \nu[x \mapsto t] \models \varphi \text{ for some } t \in \mathbb{T} \end{array}$$

where $\nu[x \mapsto t]$ maps $x$ to $t$ and $y \neq x$ to $\nu(y)$.

Previous specifications can be written in $\mathrm{FO}(<)$ (except the branching one).



# First-order vs Temporal

**First-order logic**

- $\mathrm{FO}(<)$ has a good expressive power
  ... but $\mathrm{FO}(<)$-formulae are not easy to write and to understand.
- $\mathrm{FO}(<)$ is decidable
  ... but satisfiability and model checking are non elementary.

**Temporal logics**

- no variables: time is implicit.
- quantifications and variables are replaced by modalities.
- Usual specifications are easy to write and read.
- Good complexity for satisfiability and model checking problems.
- Good expressive power.

Linear Temporal Logic (LTL) over $(\mathbb{N}, <)$ introduced by Pnueli (1977) as a convenient specification language for verification of systems.



# Temporal Specifications

**Definition: Syntax of $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$**

$$\varphi ::= \bot \mid p\ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi\, \mathsf{SU}\, \varphi \mid \varphi\, \mathsf{SS}\, \varphi$$

**Definition: Semantics: $w = (\mathbb{T}, <, h)$ temporal structure and $i \in \mathbb{T}$**

$$\begin{array}{lll} w, i \models p & \text{if} & i \in h(p) \\ w, i \models \neg\varphi & \text{if} & w, i \not\models \varphi \\ w, i \models \varphi \vee \psi & \text{if} & w, i \models \varphi \text{ or } w, i \models \psi \\ w, i \models \varphi\, \mathsf{SU}\, \psi & \text{if} & \exists k\ i < k \text{ and } w, k \models \psi \text{ and } \forall j\ (i < j < k \to w, j \models \varphi) \\ w, i \models \varphi\, \mathsf{SS}\, \psi & \text{if} & \exists k\ i > k \text{ and } w, k \models \psi \text{ and } \forall j\ (i > j > k \to w, j \models \varphi) \end{array}$$

Previous specifications can be written in $\mathrm{TL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$
(except the branching one).

# Temporal Specifications

**Definition: non-strict versions of until and since**

$$\varphi \,\mathsf{U}\, \psi \stackrel{\text{def}}{=} \psi \vee (\varphi \wedge \varphi \,\mathsf{SU}\, \psi) \qquad \varphi \,\mathsf{S}\, \psi \stackrel{\text{def}}{=} \psi \vee (\varphi \wedge \varphi \,\mathsf{SS}\, \psi)$$

$w, i \models \varphi \,\mathsf{U}\, \psi$ if $\exists k\ i \leq k$ and $w, k \models \psi$ and $\forall j\ (i \leq j < k \to w, j \models \varphi)$

$w, i \models \varphi \,\mathsf{S}\, \psi$ if $\exists k\ i \geq k$ and $w, k \models \psi$ and $\forall j\ (i \geq j > k \to w, j \models \varphi)$

**Definition: Derived modalities**

$\mathsf{X}\, \varphi \stackrel{\text{def}}{=} \bot \,\mathsf{SU}\, \varphi$ Next $\qquad \mathsf{Y}\, \varphi \stackrel{\text{def}}{=} \bot \,\mathsf{SS}\, \varphi$ Yesterday

$w, i \models \mathsf{X}\, \varphi$ if $\exists k\ i < k$ and $w, k \models \varphi$ and $\neg \exists j\ (i < j < k)$

$w, i \models \mathsf{Y}\, \varphi$ if $\exists k\ i > k$ and $w, k \models \varphi$ and $\neg \exists j\ (i > j > k)$

$$\mathsf{F}\, \varphi \stackrel{\text{def}}{=} \top \,\mathsf{U}\, \varphi \qquad \mathsf{P}\, \varphi \stackrel{\text{def}}{=} \top \,\mathsf{S}\, \varphi$$

$$\mathsf{G}\, \varphi \stackrel{\text{def}}{=} \neg \mathsf{F} \neg \varphi \qquad \mathsf{H}\, \varphi \stackrel{\text{def}}{=} \neg \mathsf{P} \neg \varphi$$

$\varphi \,\mathsf{W}\, \psi \stackrel{\text{def}}{=} (\mathsf{G}\, \varphi) \vee (\varphi \,\mathsf{U}\, \psi)$ Weak Until

$\varphi \,\mathsf{R}\, \psi \stackrel{\text{def}}{=} (\mathsf{G}\, \psi) \vee (\psi \,\mathsf{U}\, (\varphi \wedge \psi))$ Release



# Temporal Specifications

**Example: Specifications on the time flow $(\mathbb{N}, <)$**

- Safety: $\mathsf{G}\,\mathrm{good}$
- MutEx: $\neg \mathsf{F}(\mathrm{crit}_1 \wedge \mathrm{crit}_2)$
- Liveness: $\mathsf{G}\,\mathsf{F}\,\mathrm{active}$
- Response: $\mathsf{G}(\mathrm{request} \to \mathsf{F}\,\mathrm{grant})$
- Response': $\mathsf{G}(\mathrm{request} \to (\neg\mathrm{request} \,\mathsf{SU}\, \mathrm{grant}))$
- Release: $\mathrm{reset} \,\mathsf{R}\, \mathrm{alarm}$
- Strong fairness: $(\mathsf{G}\,\mathsf{F}\,\mathrm{request}) \to (\mathsf{G}\,\mathsf{F}\,\mathrm{grant})$
- Weak fairness: $(\mathsf{F}\,\mathsf{G}\,\mathrm{request}) \to (\mathsf{G}\,\mathsf{F}\,\mathrm{grant})$



# Discrete linear time flows

**Definition: discrete linear time flows $(\mathbb{T}, <)$**

A linear time flow is discrete if $\mathsf{SF}\,\top \to \mathsf{X}\,\top$ and $\mathsf{SP}\,\top \to \mathsf{Y}\,\top$ are valid formulae.

$(\mathbb{N}, <)$ and $(\mathbb{Z}, <)$ are discrete.

$(\mathbb{Q}, <)$ and $(\mathbb{R}, <)$ are not discrete.

**Exercise: For discrete linear time flows $(\mathbb{T}, <)$**

$$\varphi \,\mathsf{SU}\, \psi \equiv \mathsf{X}(\varphi \,\mathsf{U}\, \psi)$$

$$\varphi \,\mathsf{SS}\, \psi \equiv \mathsf{Y}(\varphi \,\mathsf{S}\, \psi)$$

$$\neg \mathsf{X}\, \varphi \equiv \neg \mathsf{X}\,\top \vee \mathsf{X} \neg \varphi$$

$$\neg \mathsf{Y}\, \varphi \equiv \neg \mathsf{Y}\,\top \vee \mathsf{Y} \neg \varphi$$

$$\neg(\varphi \,\mathsf{U}\, \psi) \equiv (\mathsf{G} \neg \psi) \vee (\neg \psi \,\mathsf{U}\, (\neg \varphi \wedge \neg \psi)) \equiv \neg \psi \,\mathsf{W}\, (\neg \varphi \wedge \neg \psi) \equiv \neg \varphi \,\mathsf{R}\, \neg \psi$$



# Model checking for linear behaviors

**Definition: Model checking problem**

Input: A Kripke structure $M = (S, T, I, \mathrm{AP}, \ell)$
A formula $\varphi \in \mathrm{LTL}(\mathrm{AP}, \mathsf{SU}, \mathsf{SS})$

Question: Does $M \models \varphi$ ?

- Universal MC: $M \models_\forall \varphi$ if $\ell(\sigma), 0 \models \varphi$ for all initial infinite runs $\sigma$ of $M$.
- Existential MC: $M \models_\exists \varphi$ if $\ell(\sigma), 0 \models \varphi$ for some initial infinite run $\sigma$ of $M$.

$$M \models_\forall \varphi \quad \text{iff} \quad M \not\models_\exists \neg\varphi$$

**Theorem [10, Sistla, Clarke 85], [9, Lichtenstein & Pnueli 85]**

The Model checking problem for LTL is PSPACE-complete. Proof later

## Weaknesses of linear behaviors

Example:

$\varphi$: Whenever $p$ holds, it is possible to reach a state where $q$ holds.

$\varphi$ cannot be checked on linear behaviors.

We need to consider the computation-trees.



## Weaknesses of FO specifications

Example:

$\psi$: The system has an infinite active run, but it may always reach an inactive state.

$\psi$ cannot be expressed in FO.

We need quantifications on runs: $\psi = \mathsf{EG}(\text{Active} \land \mathsf{EF}\, \neg\text{Active})$

- E: for some infinite run
- A: for all infinite runs



## MSO Specifications

Definition: Syntax of $\mathrm{MSO}(\mathrm{AP}, <)$

$$\varphi ::= \bot \mid p(x) \mid x = y \mid x < y \mid x \in X \mid \neg\varphi \mid \varphi \lor \varphi \mid \exists x\, \varphi \mid \exists X\, \varphi$$

where $p \in \mathrm{AP}$, $x, y$ are first-order variables and $X$ is a second-order variable.

Definition: Semantics of $\mathrm{MSO}(\mathrm{AP}, <)$

Let $w = (\mathbb{T}, <, h)$ be a temporal structure over $\mathrm{AP}$.
An assignment $\nu$ maps first-order variables to time points in $\mathbb{T}$
and second-order variables to sets of time points.

The semantics of first-order constructs is unchanged.

$$w, \nu \models x \in X \quad \text{if} \quad \nu(x) \in \nu(X)$$
$$w, \nu \models \exists X\, \varphi \quad \text{if} \quad w, \nu[X \mapsto T] \models \varphi \text{ for some } T \subseteq \mathbb{T}$$

where $\nu[X \mapsto T]$ maps $X$ to $T$ and keeps unchanged the other assignments.



## MSO vs Temporal

MSO logic

- $\mathrm{MSO}(<)$ has a good expressive power
  ...but $\mathrm{MSO}(<)$-formulae are not easy to write and to understand.
- $\mathrm{MSO}(<)$ is decidable on computation trees
  ...but satisfiability and model checking are non elementary.

We need a temporal logic

- with no explicit variables,
- allowing quantifications over runs,
- usual specifications should be easy to write and read,
- with good complexity for satisfiability and model checking problems,
- with good expressive power.

Computation Tree Logic $\mathrm{CTL}^*$ introduced by Emerson & Halpern (1986).

## $\mathrm{CTL}^*$ (Emerson & Halpern 86)

**Definition: Syntax of the Computation Tree Logic $\mathrm{CTL}^*(\mathrm{AP}, \mathsf{SU})$**

$$\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \,\mathsf{SU}\, \varphi \mid \mathsf{E}\,\varphi \mid \mathsf{A}\,\varphi$$

We may also add the past modality SS

**Definition: Semantics of $\mathrm{CTL}^*(\mathrm{AP}, \mathsf{SU})$**

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure.
Let $\sigma = s_0 s_1 s_2 \cdots$ be an infinte run of $M$.

$M, \sigma, i \models p$ if $p \in \ell(s_i)$
$M, \sigma, i \models \varphi \,\mathsf{SU}\, \psi$ if $\exists k > i,\ M, \sigma, k \models \psi$ and $\forall i < j < k,\ M, \sigma, j \models \varphi$

$M, \sigma, i \models \mathsf{E}\varphi$ if $M, \sigma', i \models \varphi$ for some infinite run $\sigma'$ such that $\sigma'[i] = \sigma[i]$
$M, \sigma, i \models \mathsf{A}\varphi$ if $M, \sigma', i \models \varphi$ for all infinite runs $\sigma'$ such that $\sigma'[i] = \sigma[i]$

where $\sigma[i] = s_0 \cdots s_i$.

**Remark:**

- $\sigma'[i] = \sigma[i]$ means that future is branching but past is not.



## $\mathrm{CTL}^*$ (Emerson & Halpern 86)

**Example: Some specifications**

- $\mathsf{EF}\,\varphi$: $\varphi$ is possible
- $\mathsf{AG}\,\varphi$: $\varphi$ is an invariant
- $\mathsf{AF}\,\varphi$: $\varphi$ is unavoidable
- $\mathsf{EG}\,\varphi$: $\varphi$ holds globally along some path

**Remark: Some equivalences**

- $\mathsf{A}\,\varphi \equiv \neg\,\mathsf{E}\,\neg\varphi$
- $\mathsf{E}(\varphi \vee \psi) \equiv \mathsf{E}\,\varphi \vee \mathsf{E}\,\psi$
- $\mathsf{A}(\varphi \wedge \psi) \equiv \mathsf{A}\,\varphi \wedge \mathsf{A}\,\psi$



## Model checking of $\mathrm{CTL}^*$

**Definition: Existential and universal model checking**

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and $\varphi \in \mathrm{CTL}^*$ a formula.

$M \models_\exists \varphi$ if $M, \sigma, 0 \models \varphi$ for some initial infinite run $\sigma$ of $M$.
$M \models_\forall \varphi$ if $M, \sigma, 0 \models \varphi$ for all initial infinite runs $\sigma$ of $M$.

Remark: $M \models_\forall \varphi$ iff $M \not\models_\exists \neg\varphi$

**Definition: Model checking problems $\mathrm{MC}^\forall_{\mathrm{CTL}^*}$ and $\mathrm{MC}^\exists_{\mathrm{CTL}^*}$**

Input: A Kripke structure $M = (S, T, I, \mathrm{AP}, \ell)$ and a formula $\varphi \in \mathrm{CTL}^*$

Question: Does $M \models_\forall \varphi$ ? or Does $M \models_\exists \varphi$ ?

**Theorem:**

The model checking problem for $\mathrm{CTL}^*$ is PSPACE-complete. Proof later



## State formulae and path formulae

**Definition: State formulae**

$\varphi \in \mathrm{CTL}^*$ is a state formula if $\forall M, \sigma, \sigma', i, j$ such that $\sigma(i) = \sigma'(j)$ we have

$$M, \sigma, i \models \varphi \iff M, \sigma', j \models \varphi$$

If $\varphi$ is a state formula and $M = (S, T, I, \mathrm{AP}, \ell)$, define

$M, s \models \varphi$ if $M, \sigma, 0 \models \varphi$ for some infinite run $\sigma$ of $M$ with $\sigma(0) = s$

and $[\![\varphi]\!]^M = \{s \in S \mid M, s \models \varphi\}$

**Example: State formulae**

Atomic propositions are state formulae: $[\![p]\!] = \{s \in S \mid p \in \ell(s)\}$
State formulae are closed under boolean connectives.

$$[\![\neg\varphi]\!] = S \setminus [\![\varphi]\!] \qquad [\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$$

Formulae of the form $\mathsf{E}\,\varphi$ or $\mathsf{A}\,\varphi$ are state formulae, provided $\varphi$ is future.

Remark: $M \models_\exists \varphi$ iff $I \cap [\![\mathsf{E}\,\varphi]\!] \neq \emptyset$ $\quad M \models_\forall \varphi$ iff $M \not\models_\exists \neg\varphi$

**Definition: Alternative syntax**

State formulae $\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{E}\,\psi \mid \mathsf{A}\,\psi$
Path formulae $\psi ::= \varphi \mid \neg\psi \mid \psi \vee \psi \mid \psi \,\mathsf{SU}\, \psi$

# CTL (Clarke & Emerson 81)

**Definition: Computation Tree Logic CTL(AP, X, U)**

Syntax:

$$\varphi ::= \bot \mid p \ (p \in \mathrm{AP}) \mid \neg\varphi \mid \varphi \vee \varphi \mid \mathsf{EX}\,\varphi \mid \mathsf{AX}\,\varphi \mid \mathsf{E}\,\varphi\,\mathsf{U}\,\varphi \mid \mathsf{A}\,\varphi\,\mathsf{U}\,\varphi$$

The semantics is inherited from $\mathrm{CTL}^*$.

**Remark: All CTL formulae are state formulae**

$$[\![\varphi]\!]^M = \{s \in S \mid M, s \models \varphi\}$$

**Examples: Macros**

- $\mathsf{EF}\,\varphi = \mathsf{E}\top\mathsf{U}\,\varphi$ and $\mathsf{AG}\,\varphi = \neg\,\mathsf{EF}\,\neg\varphi$
- $\mathsf{AF}\,\varphi = \mathsf{A}\top\mathsf{U}\,\varphi$ and $\mathsf{EG}\,\varphi = \neg\,\mathsf{AF}\,\neg\varphi$
- $\mathsf{AG}(\mathrm{req} \to \mathsf{EF}\,\mathrm{grant})$
- $\mathsf{AG}(\mathrm{req} \to \mathsf{AF}\,\mathrm{grant})$



# CTL (Clarke & Emerson 81)

**Definition: Semantics**

All CTL-formulae are state formulae. Hence, we have a simpler semantics.
Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure without deadlocks and let $s \in S$.

$M, s \models p$ if $p \in \ell(s)$

$M, s \models \mathsf{EX}\,\varphi$ if $\exists s \to s'$ with $M, s' \models \varphi$

$M, s \models \mathsf{AX}\,\varphi$ if $\forall s \to s'$ we have $M, s' \models \varphi$

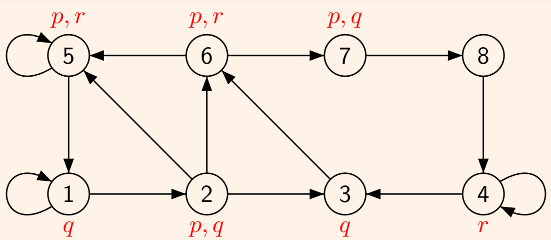
$M, s \models \mathsf{E}\,\varphi\,\mathsf{U}\,\psi$ if $\exists s = s_0 \to s_1 \to s_2 \to \cdots s_k$ finite path, with $M, s_k \models \psi$ and $M, s_j \models \varphi$ for all $0 \le j < k$

$M, s \models \mathsf{A}\,\varphi\,\mathsf{U}\,\psi$ if $\forall s = s_0 \to s_1 \to s_2 \to \cdots$ infinite paths, $\exists k \ge 0$ with $M, s_k \models \psi$ and $M, s_j \models \varphi$ for all $0 \le j < k$



# CTL (Clarke & Emerson 81)

**Example:**




$[\![\mathsf{EX}\,p]\!] =$

$[\![\mathsf{AX}\,p]\!] =$

$[\![\mathsf{EF}\,p]\!] =$

$[\![\mathsf{AF}\,p]\!] =$

$[\![\mathsf{E}\,q\,\mathsf{U}\,r]\!] =$

$[\![\mathsf{A}\,q\,\mathsf{U}\,r]\!] =$



# CTL (Clarke & Emerson 81)

**Remark: Equivalent formulae**

- $\mathsf{AX}\,\varphi \equiv \neg\,\mathsf{EX}\,\neg\varphi$,
- $\neg(\varphi\,\mathsf{U}\,\psi) \equiv \mathsf{G}\,\neg\psi \vee (\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi))$
- $\mathsf{A}\,\varphi\,\mathsf{U}\,\psi \equiv \neg\,\mathsf{EG}\,\neg\psi \wedge \neg\,\mathsf{E}(\neg\psi\,\mathsf{U}\,(\neg\varphi \wedge \neg\psi))$
- $\mathsf{AG}(\mathrm{req} \to \mathsf{F}\,\mathrm{grant}) \equiv \mathsf{AG}(\mathrm{req} \to \mathsf{AF}\,\mathrm{grant})$
- $\mathsf{A}\,\mathsf{G}\,\mathsf{F}\,\varphi \equiv \mathsf{AG}\,\mathsf{AF}\,\varphi$
- $\mathsf{E}\,\mathsf{F}\,\mathsf{G}\,\varphi \equiv \mathsf{EF}\,\mathsf{EG}\,\varphi$
- $\mathsf{EG}\,\mathsf{EF}\,\varphi \not\equiv \mathsf{E}\,\mathsf{G}\,\mathsf{F}\,\varphi \not\equiv \mathsf{EG}\,\mathsf{AF}\,\varphi$
- $\mathsf{AF}\,\mathsf{AG}\,\varphi \not\equiv \mathsf{A}\,\mathsf{F}\,\mathsf{G}\,\varphi \not\equiv \mathsf{AF}\,\mathsf{EG}\,\varphi$
- $\mathsf{EG}\,\mathsf{EX}\,\varphi \not\equiv \mathsf{E}\,\mathsf{G}\,\mathsf{X}\,\varphi \not\equiv \mathsf{EG}\,\mathsf{AX}\,\varphi$

# Model checking of CTL

**Definition: Existential and universal model checking**

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and $\varphi \in \mathrm{CTL}$ a formula.

$M \models_\exists \varphi$ if $M, s \models \varphi$ for some $s \in I$.
$M \models_\forall \varphi$ if $M, s \models \varphi$ for all $s \in I$.

**Remark:**

$M \models_\exists \varphi$ iff $I \cap [\![\varphi]\!] \neq \emptyset$
$M \models_\forall \varphi$ iff $I \subseteq [\![\varphi]\!]$
$M \models_\forall \varphi$ iff $M \not\models_\exists \neg\varphi$

**Definition: Model checking problems $\mathrm{MC}^\forall_{\mathrm{CTL}}$ and $\mathrm{MC}^\exists_{\mathrm{CTL}}$**

Input: A Kripke structure $M = (S, T, I, \mathrm{AP}, \ell)$ and a formula $\varphi \in \mathrm{CTL}$
Question: Does $M \models_\forall \varphi$ ? or Does $M \models_\exists \varphi$ ?

**Theorem:**

Let $M = (S, T, I, \mathrm{AP}, \ell)$ be a Kripke structure and $\varphi \in \mathrm{CTL}$ a formula.
The model checking problem $M \models_\exists \varphi$ is decidable in time $\mathcal{O}(|M| \cdot |\varphi|)$



# References


[1] Christel Baier and Joost-Pieter Katoen.
*Principles of Model Checking.*
MIT Press, 2008.

[2] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, Ph. Schnoebelen.
*Systems and Software Verification. Model-Checking Techniques and Tools.*
Springer, 2001.

[3] E.M. Clarke, O. Grumberg, D.A. Peled.
*Model Checking.*
MIT Press, 1999.

[4] Z. Manna and A. Pnueli.
*The Temporal Logic of Reactive and Concurrent Systems: Specification.*
Springer, 1991.

[5] Z. Manna and A. Pnueli.
*Temporal Verification of Reactive Systems: Safety.*
Springer, 1995.




# References


[6] S. Demri and P. Gastin.
*Specification and Verification using Temporal Logics.*
In Modern applications of automata theory, IISc Research Monographs 2.
World Scientific, 2012.
http://www.lsv.ens-cachan.fr/~gastin/mes-publis.php

[7] D. Gabbay, I. Hodkinson and M. Reynolds.
*Temporal logic: mathematical foundations and computational aspects.*
Vol 1, Clarendon Press, Oxford, 1994.

[8] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi.
On the temporal analysis of fairness.
In *7th Annual ACM Symposium PoPL'80*, 163–173. ACM Press.

[9] O. Lichtenstein and A. Pnueli.
Checking that finite state concurrent programs satisfy their linear specification.
In *ACM Symposium PoPL'85*, 97–107.

[10] A. Sistla and E. Clarke.
The complexity of propositional linear temporal logic.
*Journal of the Association for Computing Machinery.* **32** (3), 733–749, (1985).