Model Checking

Computation Tree Logic (CTL)

[Baier & Katoen, Chapter 6.1–6.3]

Joost-Pieter Katoen and Tim Quatmann

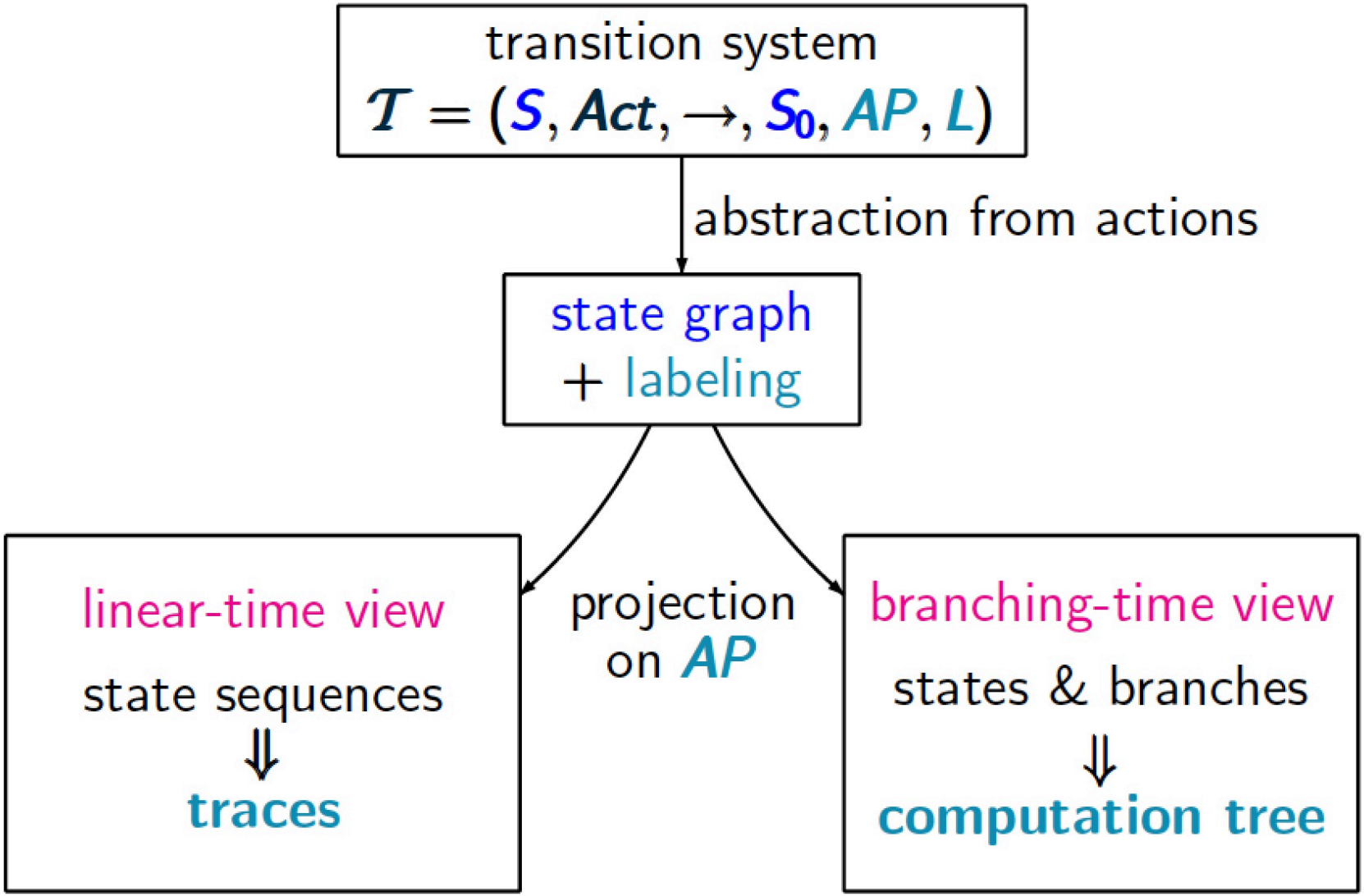
Software Modeling and Verification Group

RWTH Aachen, SoSe 2022

Overview

Overview

Linear Time Versus Branching Time

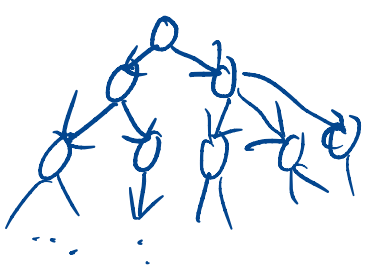


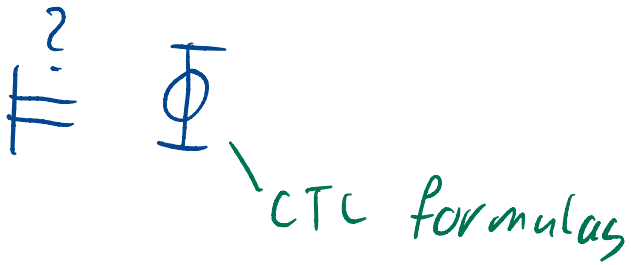
Linear Time Versus Branching Time

- Linear Temporal Logic (LTL) is interpreted over $\underbrace{\text{infinite sequences}}_{\text{traces}}$
- Traces are obtained from paths in a transition system.

Linear Time Versus Branching Time

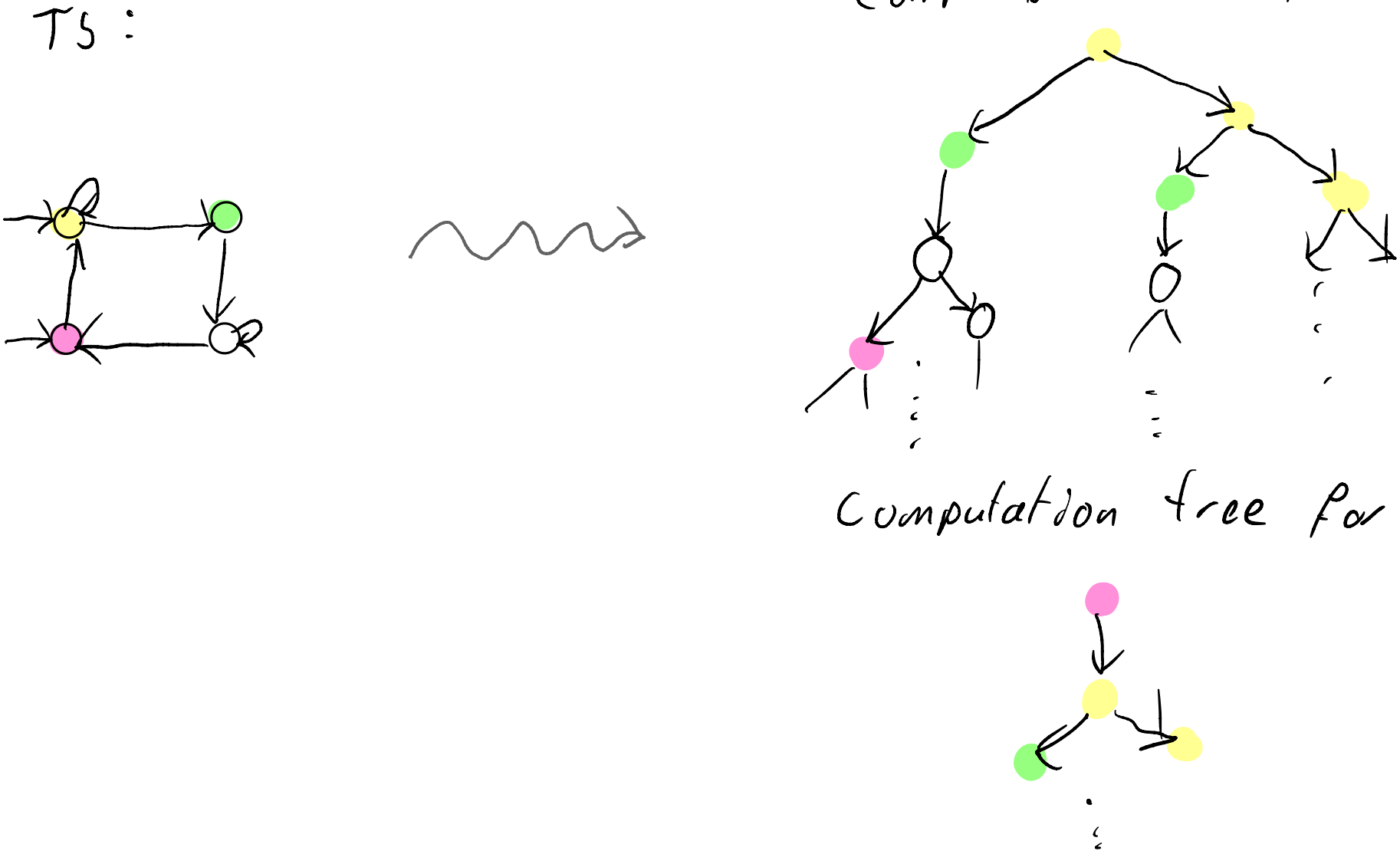
- Linear Temporal Logic (LTL) is interpreted over $\underbrace{\text{infinite sequences}}_{\text{traces}}$
- Traces are obtained from paths in a transition system.





- Computation Tree Logic (CTL) is interpreted over $\underbrace{\text{infinite trees}}_{\text{computation trees}}$
- Computation trees are infinite trees whose nodes are labelled with sets of propositions
  - They are obtained by unfolding a transition system
  - Such trees keep track of branching between several traces

Unfolding a Transition System



Linear Time Versus Branching Time

	linear time	branching time
behavior	path based traces	state based computation tree
temporal logic	**LTL** path formulas	**CTL** state formulas
model checking	PSPACE-complete $\mathcal{O}(\mathit{size}(\mathcal{T}) \cdot \exp(\lvert\varphi\rvert))$	PTIME $\mathcal{O}(\mathit{size}(\mathcal{T}) \cdot \lvert\Phi\rvert)$
impl. relation	trace inclusion trace equivalence PSPACE-complete	simulation bisimulation PTIME

Overview

Computation Tree Logic

Edmund M. Clarke, Jr.
(1945–2020)

E. Allen Emerson
(1954–)

CTL Syntax

Definition: Syntax Computation Tree Logic

- CTL state-formulas with $a \in AP$ obey the grammar:

$$\Phi ::= \text{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi \mid \forall\varphi$$

- and φ is a path-formula formed by the grammar:

$$\varphi ::= \bigcirc\Phi \mid \Phi_1 \,\mathsf{U}\, \Phi_2.$$

Example CTL State-formulas

- $\forall\Box\exists\bigcirc a \equiv \neg\forall\Diamond\neg\exists\bigcirc a \equiv \neg\forall(\text{true}\ \mathsf{U}\ \neg\exists\bigcirc a)$
- $\exists(\forall\Box a)\,\mathsf{U}\, b$

CTL Syntax

Definition: Syntax Computation Tree Logic

- CTL state-formulas with $a \in AP$ obey the grammar:

$$\Phi \ ::= \ \text{true} \ \Big| \ a \ \Big| \ \Phi_1 \wedge \Phi_2 \ \Big| \ \neg\Phi \ \Big| \ \exists\varphi \ \Big| \ \forall\varphi$$

- and φ is a path-formula formed by the grammar:

$$\varphi \ ::= \ \bigcirc \Phi \ \Big| \ \Phi_1 \, \mathsf{U} \, \Phi_2.$$

Intuition

- $s \vDash \forall\varphi$ if all paths starting in s fulfill φ
- $s \vDash \exists\varphi$ if some path starting in s fulfill φ

Derived CTL Operators

potentially Φ:	$\exists\Diamond\,\Phi$	$=$	$\exists(\text{true}\,\mathsf{U}\,\Phi)$
inevitably Φ:	$\forall\Diamond\,\Phi$	$=$	$\forall(\text{true}\,\mathsf{U}\,\Phi)$
potentially always Φ:	$\exists\Box\Phi$	$:=$	$\neg\forall\Diamond\,\neg\Phi$
invariantly Φ:	$\forall\Box\Phi$	$=$	$\neg\exists\Diamond\,\neg\Phi$

Derived CTL Operators

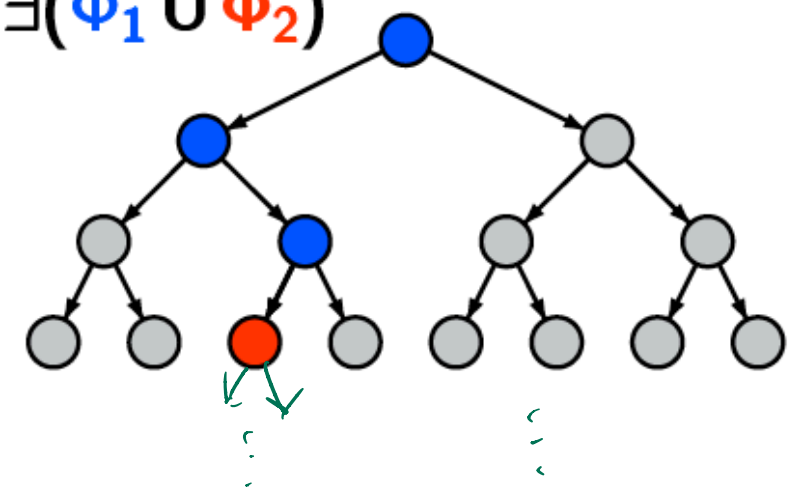
LTL: $\varphi W \psi \equiv \varphi U \psi \vee \Box \varphi$

potentially Φ:	$\exists \Diamond \Phi$	$=$	$\exists(\text{true}\, U\, \Phi)$
inevitably Φ:	$\forall \Diamond \Phi$	$=$	$\forall(\text{true}\, U\, \Phi)$
potentially always Φ:	$\exists \Box \Phi$	$:=$	$\neg \forall \Diamond \neg \Phi$
invariantly Φ:	$\forall \Box \Phi$	$=$	$\neg \exists \Diamond \neg \Phi$
weak until:	$\exists(\Phi\, W\, \Psi)$	$=$	$\neg \forall\big((\Phi \wedge \neg\Psi)\, U\, (\neg\Phi \wedge \neg\Psi)\big)$
	$\forall(\Phi\, W\, \Psi)$	$=$	$\neg \exists\big((\Phi \wedge \neg\Psi)\, U\, (\neg\Phi \wedge \neg\Psi)\big)$

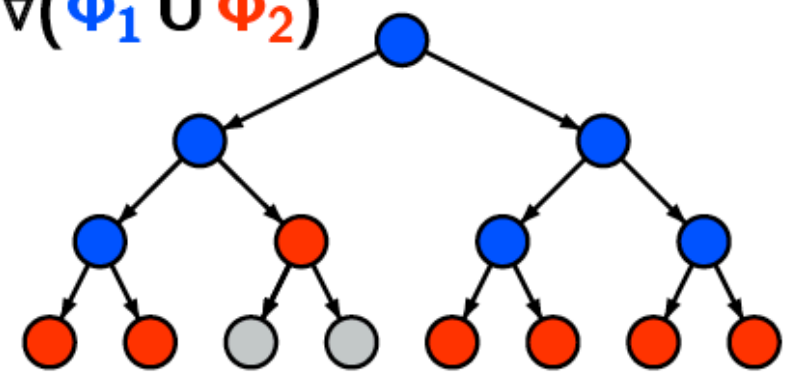
The Boolean connectives are derived as usual

Intuitive CTL Semantics

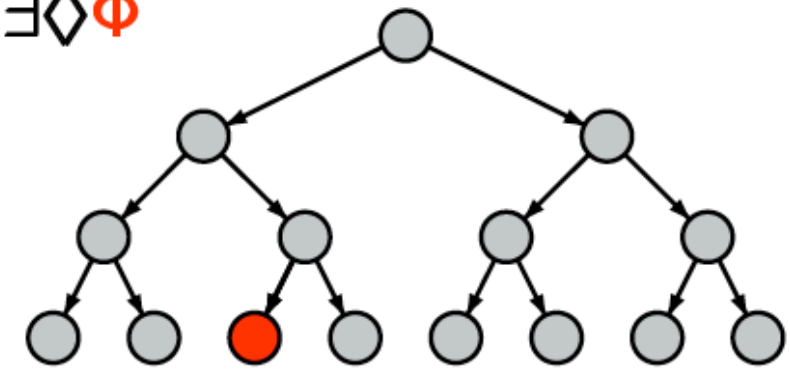
$\exists(\Phi_1 \, \mathsf{U} \, \Phi_2)$



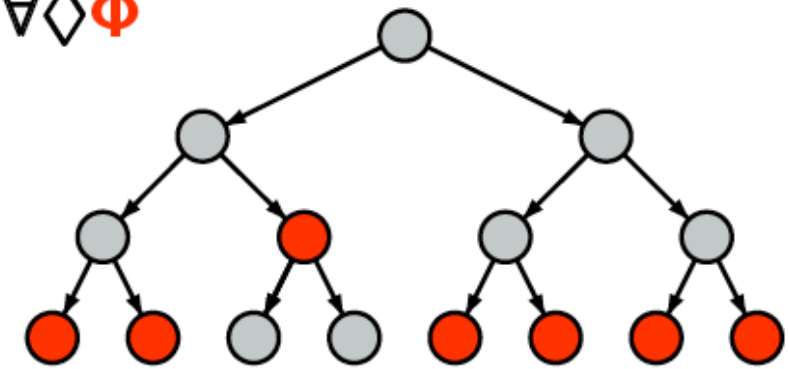
$\forall(\Phi_1 \, \mathsf{U} \, \Phi_2)$



$\exists \Diamond \Phi$

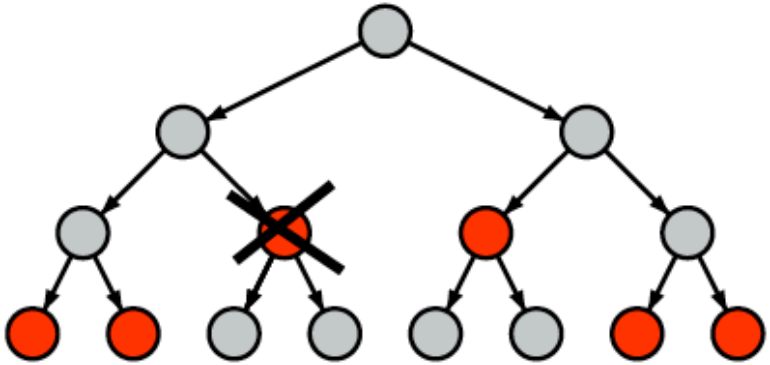


$\forall \Diamond \Phi$

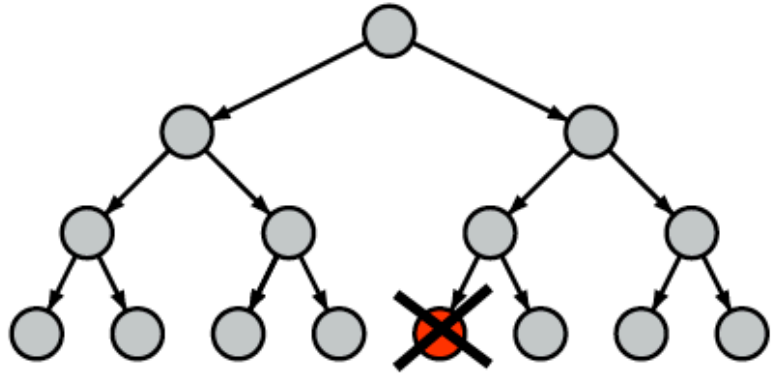


Intuitive CTL Semantics

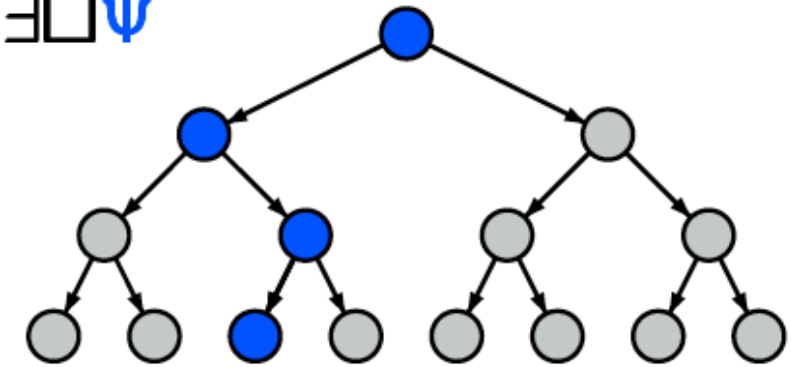
$\neg A \Diamond \phi$



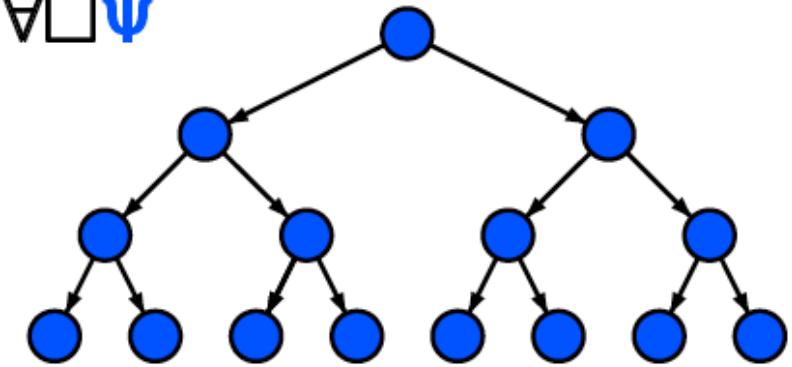
$\neg E \Diamond \phi$



$E \Box \psi$



$A \Box \psi$



Example CTL Formulas

- Mutual Exclusion over $AP = \{\texttt{crit}_1, \texttt{crit}_2\}$

$$\forall\Box\neg(crit_1 \wedge crit_2)$$

- Starvation Freedom over $AP = \{\texttt{req}, \texttt{resp}\}$

$$\forall\Box(req \Rightarrow \forall\Diamond resp)$$

- "A reset is always possible" over $AP = \{\texttt{start}\}$

$$\forall\Box\exists\Diamond start$$

Overview

CTL Semantics

$\models$

Define a satisfaction relation for CTL-formulas over AP for a given transition system TS without terminal states.

Two parts:

- Interpretation of state-formulas over states of TS

$s \models \Phi$ – CTL state form

- Interpretation of path-formulas over paths of TS

$\pi \models \varphi$ – CTL path formula

CTL Semantics (1)

Notation

$TS, s \models \Phi$ if and only if state-formula Φ holds in state s of transition system TS. As TS is known from the context we simply write $s \models \Phi$.

Definition: Satisfaction relation for CTL state-formulas

The satisfaction relation $\models$ is defined for CTL state-formulas by:

$$
\begin{array}{lll}
s \models a & \text{iff} & a \in L(s) \\
s \models \neg \Phi & \text{iff} & \text{not } (s \models \Phi) \\
s \models \Phi \wedge \Psi & \text{iff} & (s \models \Phi) \text{ and } (s \models \Psi)
\end{array}
$$

CTL Semantics (1)

Notation

$TS, s \models \Phi$ if and only if state-formula Φ holds in state s of transition system TS. As TS is known from the context we simply write $s \models \Phi$.

Definition: Satisfaction relation for CTL state-formulas

The satisfaction relation $\models$ is defined for CTL state-formulas by:

$$
\begin{array}{lll}
s \models a & \text{iff} & a \in L(s) \\
s \models \neg \Phi & \text{iff} & \text{not } (s \models \Phi) \\
s \models \Phi \wedge \Psi & \text{iff} & (s \models \Phi) \text{ and } (s \models \Psi) \\
s \models \exists \varphi & \text{iff} & \text{there exists } \pi \in \mathit{Paths}(s).\ \pi \models \varphi \\
s \models \forall \varphi & \text{iff} & \text{for all } \pi \in \mathit{Paths}(s).\ \pi \models \varphi
\end{array}
$$

where the semantics of CTL path-formulas is defined on the next slide.

CTL Semantics (2)

Definition: satisfaction relation for CTL path-formulas

Given path π and CTL path-formula φ, the satisfaction relation $\models$ where $\pi \models \varphi$ if and only if path π satisfies φ is defined as follows:

$$\pi \models \bigcirc \Phi \quad \text{iff } \pi[1] \models \Phi$$

$$\pi \models \Phi \,\mathsf{U}\, \Psi \quad \text{iff } (\exists\, j \geq 0.\ \pi[j] \models \Psi \text{ and } (\forall\, 0 \leq i < j.\ \pi[i] \models \Phi))$$

where $\pi[i]$ denotes the state s_i in the path $\pi = s_0\, s_1\, s_2 \ldots$.

LTL: $\pi[j \ldots] = s_j\, s_{j+1}\, s_{j+2} \ldots \models \varphi$

Transition System Semantics

- For CTL-state-formula Φ, the satisfaction set $Sat(\Phi)$ is defined by:

$$Sat(\Phi) = \{ s \in S \mid s \models \Phi \} \subseteq S$$

- TS satisfies CTL-formula Φ iff Φ holds in all its initial states:

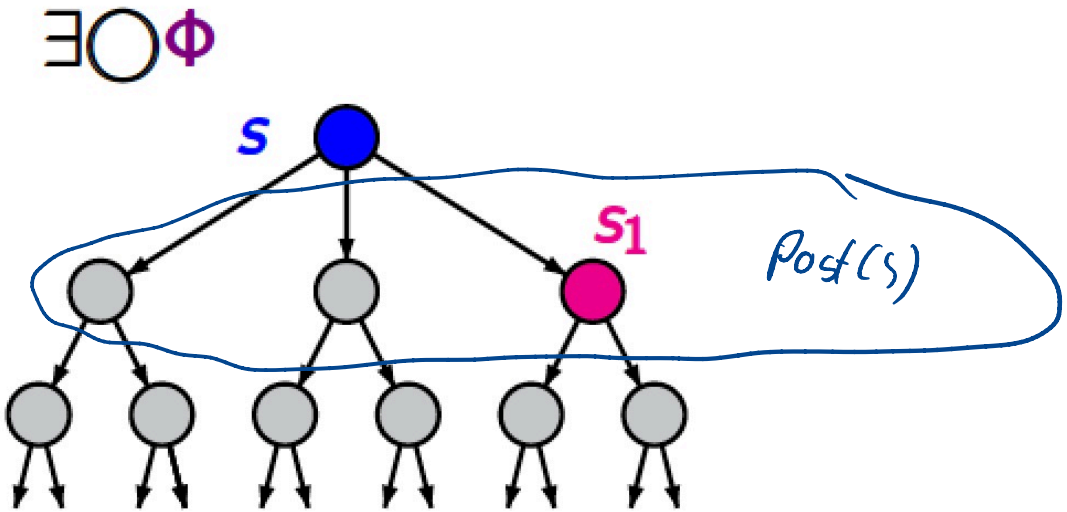
$$TS \models \Phi \quad \text{if and only if} \quad \forall s_0 \in I.\ s_0 \models \Phi$$

$$\text{iff} \quad I \subseteq Sat(\phi)$$

- Point of attention: $TS \not\models \Phi$ is not equivalent to $TS \models \neg\Phi$

  because of several initial states, e.g., $s_0 \models \exists\Box\Phi$ and $s_0' \not\models \exists\Box\Phi$

Semantics of ◯-Operator

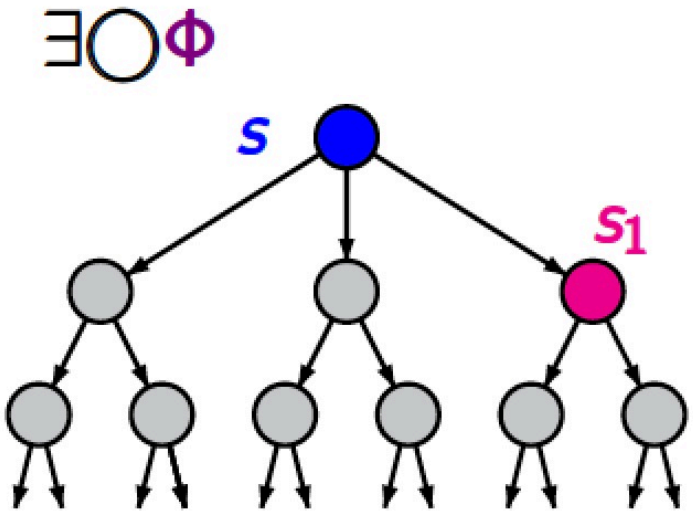


$Post(s) \cap Sat(\Phi) \neq \varnothing$

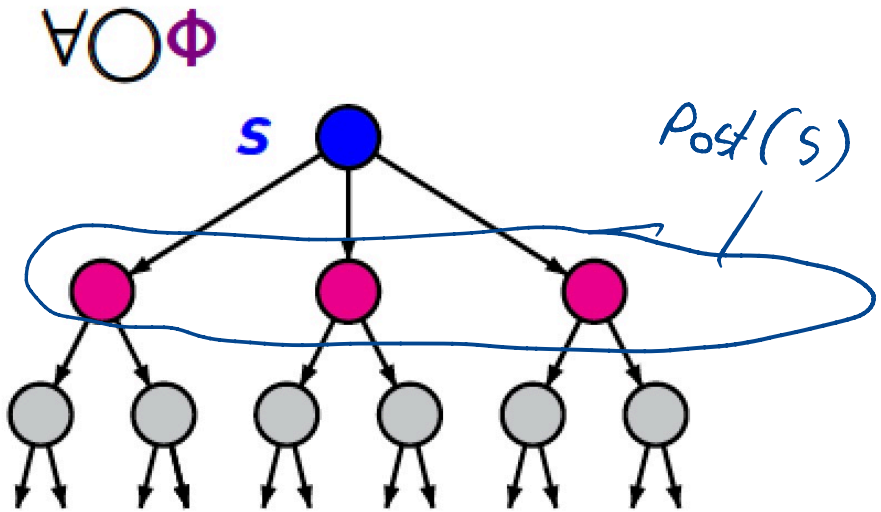
$s \models \exists \bigcirc \Phi$ iff $\exists \pi = s\, s_1\, s_2 \ldots \in Paths(s).\ \pi \models \bigcirc \Phi$, that is: $s_1 \models \Phi$

iff $Post(s) \cap Sat(\Phi) \neq \emptyset$

Semantics of ◯-Operator



$Post(s) \cap Sat(\Phi) \neq \varnothing$

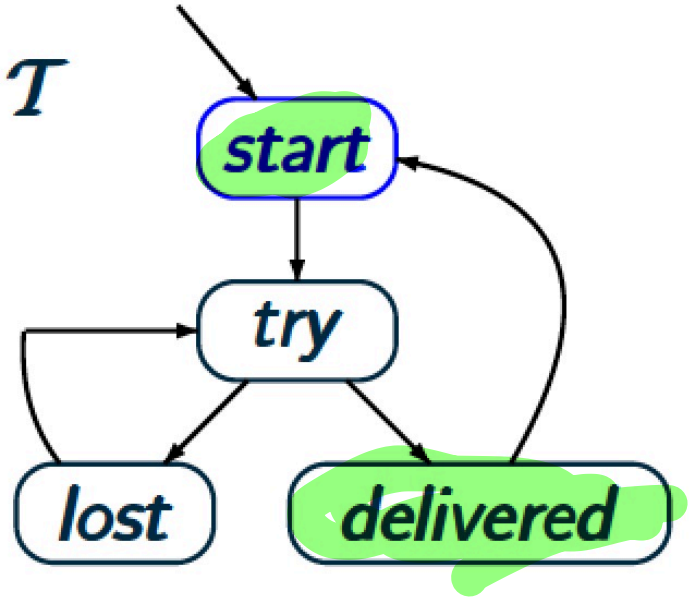


$Post(s) \subseteq Sat(\Phi)$

$s \models \exists \bigcirc \Phi$ iff $\exists \pi = s\, s_1\, s_2 \ldots \in Paths(s).\ \pi \models \bigcirc \Phi$, that is: $s_1 \models \Phi$

$s \models \forall \bigcirc \Phi$ iff $\forall \pi = s\, s_1\, s_2 \ldots \in Paths(s).\ \pi \models \bigcirc \Phi$, that is: $s_1 \models \Phi$

Example



$T \not\models \forall\Box\forall\Diamond start \quad \therefore \overline{\phi}$

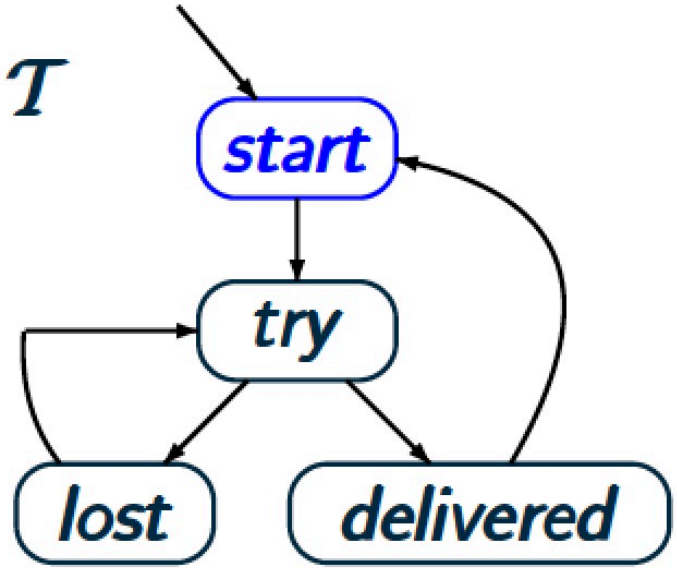
$Sat(start) = \{start\}$

$Sat(\forall\Diamond\, start) = \{start, del\}$

$Sat(\forall\Box\,\bigcirc) = \emptyset$

$1 \notin \emptyset \rightsquigarrow T \not\models \overline{\phi}$

Example



$$\mathcal{T} \not\models \forall\Box\forall\Diamond start$$

CTL formula

$$\Phi = \forall\Box\forall\Diamond start \quad \widehat{=} \quad \forall\Box(start \vee delivered)$$

$$Sat(\forall\Diamond start) = \{start, delivered\}$$

$$Sat(\Phi) = \varnothing$$

Infinitely Often

$s \models \forall\Box\forall\Diamond\, \Phi$ iff $\forall \pi \in \mathit{Paths}(s)$ a Φ-state is visited infinitely often.

For $\Phi = a \in AP$ we get

$$\underbrace{s \models \forall\Box\forall\Diamond\, a}_{\text{CTL}} \quad \text{iff} \quad \underbrace{s \models \Box\Diamond\, a}_{\text{LTL}}$$

Proof for: $\underbrace{s \models \forall\Box\forall\Diamond a}_{\text{CTL}}$ iff $\underbrace{s \models \Box\Diamond a}_{\text{LTL}}$

"$\Rightarrow$" Assume $s \models \forall\Box\forall\Diamond a$

Let $\pi = \underbrace{s_0}_{s} s_1 s_2 \cdots \in Paths(s)$. We have

$s \models \forall\Box\forall\Diamond a$

$\Rightarrow \pi \models \Box\forall\Diamond a$

$\Rightarrow \forall j \geq 0 \; s_j \models \forall\Diamond a$

$\Rightarrow \forall j \geq 0 \; \underbrace{\pi[j\ldots] = s_j s_{j+1} \ldots}_{\in Paths(s_j)} \models \Diamond a$

$\Rightarrow \forall j \geq 0 \; \exists i \geq j \quad s_i = \pi[i] \models a$

$\Rightarrow \pi[i] \models a$ for ∞ many $i \geq 0$

$\Rightarrow s \models \Box\Diamond a$

Proof for: $\underbrace{s \models \forall\Box\forall\Diamond a}_{\text{CTL}}$ iff $\underbrace{s \models \Box\Diamond a}_{\text{LTL}}$

"$\Leftarrow$"

Let $\pi = s_0 s_1 \ldots \in Paths(s)$, with $s_0 = s$

We have to show: $\pi \models \Box\forall\Diamond a$, i.e.

$\forall j \geq 0 : s_j \models \forall\Diamond a$

($=$) $\forall j \geq 0 : \forall \pi' = s_j' s_{j+1}' s_{j+2}' \ldots \in Paths(s_j) :$

$\pi' \models \Diamond a$

Since $s \not\models \Box\Diamond a$: $\pi'' = s_0 s_1 \ldots s_j' s_{j+1}' s_{j+2}' \ldots$ with $s_0 = s$, $s_j' = s_j$

$\in Paths(s)$

$\rightsquigarrow \pi'' \not\models \Box\Diamond a$ i.e. $\forall k \geq 0 \; \exists \ell \geq k : s_\ell \not\models a$

In particular:

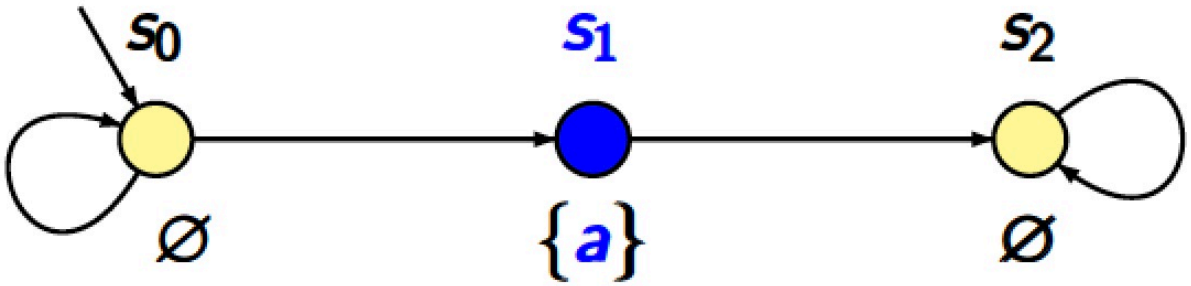
$\pi' \not\models \Diamond a$

$\rightsquigarrow s_j \not\models \forall\Diamond a$

Thus: $\pi \not\models \Box\forall\Diamond a$ for all $\pi \in Paths(s)$

Thus: $s \not\models \forall\Box\forall\Diamond a$

Example



(1) Does $TS \models \exists\bigcirc\forall\Box\neg a$?

$Sat(\neg a) = \{s_0, s_2\}$

$Sat(\forall\Box\neg a) = \{s_2\}$

$Sat(\exists\bigcirc\forall\Box\neg a)$
$= Sat(\exists\bigcirc "s_2") = \{s_1, s_2\}$
$I = \{s_0\} \notin \{s_1, s_2\} \rightarrow$ ✗

(2) Does $TS \models \forall\Box\exists\bigcirc\neg a$?

$Sat(\exists\bigcirc\neg a) = \{s_0, s_1, s_2\} = S$

$Sat(\forall\Box "S") = Sat(\forall\Box true) = S \supseteq I$ ✓

Overview

CTL Equivalence

Definition: CTL equivalence

CTL-formulas Φ and Ψ (both over AP) are equivalent:

$$\Phi \equiv_{CTL} \Psi \quad \text{if and only if} \quad Sat(\Phi) = Sat(\Psi) \quad \text{for any } TS \text{ (over } AP)$$

If it is clear from the context that we deal with CTL-formulas, we simply write $\Phi \equiv \Psi$.

Equivalently,

$$\Phi \equiv \Psi \quad \text{iff} \quad \big(\forall\, TS \;.\; TS \models \Phi \quad \text{iff} \quad TS \models \Psi \big)$$

Duality

$$\forall \bigcirc \Phi \equiv \neg \exists \bigcirc \neg \Phi$$

$$\exists \bigcirc \Phi \equiv \neg \forall \bigcirc \neg \Phi$$

$$\forall \Diamond \Phi \equiv \neg \exists \Box \neg \Phi$$

$$\exists \Diamond \Phi \equiv \neg \forall \Box \neg \Phi$$

$$\forall (\Phi \, \mathsf{U} \, \Psi) \equiv \neg \exists ((\Phi \wedge \neg \Psi) \, \mathsf{W} \, (\neg \Phi \wedge \neg \Psi))$$

Distributive Laws

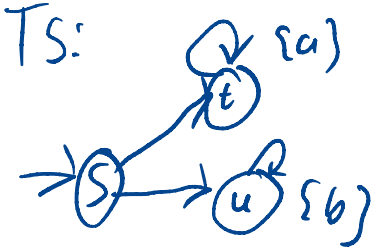
$$\forall\Box(\Phi \wedge \Psi) \equiv \forall\Box\Phi \wedge \forall\Box\Psi$$

$$\exists\Diamond(\Phi \vee \Psi) \equiv \exists\Diamond\Phi \vee \exists\Diamond\Psi$$

But: $\forall\Diamond(\Phi \vee \Psi) \not\equiv \forall\Diamond\Phi \vee \forall\Diamond\Psi$

$\Phi = a \quad \Psi = b$

TS:



$s \models \forall\Diamond(a \vee b)$ since $Sat(a \vee b) = \{t, u\}$

$\rightarrow Sat(\forall\Diamond(a \vee b)) = \{s, t, u\}$

$s \not\models \forall\Diamond a \vee \forall\Diamond b$ since $s \notin Sat(\forall\Diamond a)$ because

$s\,u^{\omega} \in Paths(s)$ violates $\Diamond a$

similarly $s \notin Sat(\forall\Diamond b)$

Distributive Laws

$$\forall\Box(\Phi \wedge \Psi) \equiv \forall\Box\Phi \wedge \forall\Box\Psi$$

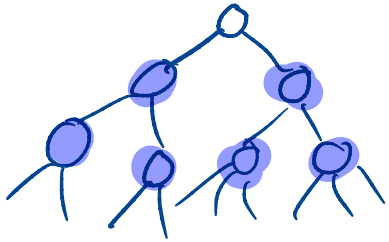
$$\exists\Diamond(\Phi \vee \Psi) \equiv \exists\Diamond\Phi \vee \exists\Diamond\Psi$$

$$\text{But: } \forall\Diamond(\Phi \vee \Psi) \not\equiv \forall\Diamond\Phi \vee \forall\Diamond\Psi$$

$$\exists\Box(\Phi \wedge \Psi) \not\equiv \exists\Box\Phi \wedge \exists\Box\Psi$$

Duality $\bigcirc$ and $\Box$ — Correct or Wrong?

$$\forall\bigcirc\forall\Box a \equiv \forall\Box\forall\bigcirc a$$



Duality ◯ and □ — Correct or Wrong?

$$\forall\bigcirc\forall\Box a \equiv \forall\Box\forall\bigcirc a$$

correct.

$$\exists\bigcirc\exists\Box a \equiv \exists\Box\exists\bigcirc a$$

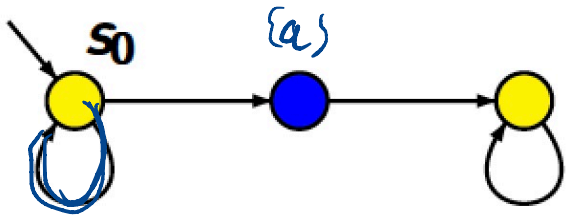
Duality ○ and □ — Correct or Wrong?

$$\forall\bigcirc\forall\Box a \equiv \forall\Box\forall\bigcirc a$$

correct.

$$\exists\bigcirc\exists\Box a \equiv \exists\Box\exists\bigcirc a$$

wrong, e.g.,



$$s_0 \not\models \exists\bigcirc\exists\Box a$$
$$s_0 \models \exists\Box\exists\bigcirc a$$

$$s_0 \models \exists\bigcirc a$$
$$\Longrightarrow \quad s_0\, s_0\, s_0 \ldots \models \Box\exists\bigcirc a$$
$$\Longrightarrow \quad s_0 \models \exists\Box\exists\bigcirc a$$

Expansion Laws

Recall in LTL: $\varphi \mathsf{U} \psi \equiv \psi \vee (\varphi \wedge \bigcirc(\varphi \mathsf{U} \psi))$

Expansion Laws

Recall in LTL: $\varphi \,\mathsf{U}\, \psi \;\equiv\; \psi \vee (\varphi \wedge \bigcirc(\varphi \,\mathsf{U}\, \psi))$

CTL expansion laws

For any CTL-formula Φ and Ψ:

$$\forall(\Phi \,\mathsf{U}\, \Psi) \;\equiv\; \Psi \vee (\Phi \wedge \forall\bigcirc \forall(\Phi \,\mathsf{U}\, \Psi))$$

$$\forall\Diamond \Phi \;\equiv\; \Phi \vee \forall\bigcirc \forall\Diamond \Phi$$

$$\forall\Box\Phi \;\equiv\; \Phi \wedge \forall\bigcirc \forall\Box\Phi$$

$$\exists(\Phi \,\mathsf{U}\, \Psi) \;\equiv\; \Psi \vee (\Phi \wedge \exists\bigcirc \exists(\Phi \,\mathsf{U}\, \Psi))$$

$$\exists\Diamond \Phi \;\equiv\; \Phi \vee \exists\bigcirc \exists\Diamond \Phi$$

$$\exists\Box\Phi \;\equiv\; \Phi \wedge \exists\bigcirc \exists\Box\Phi$$

Overview

Equivalence of CTL and LTL Formulas

Definition: equivalence of LTL and CTL formulas

CTL-formula Φ and LTL-formula φ (both over AP) are equivalent, denoted $\Phi \equiv \varphi$, if for any transition system TS (over AP):

$$TS \models \Phi \quad \text{if and only if} \quad TS \models \varphi.$$

Examples

- "Next a": $\forall \bigcirc a \equiv \bigcirc a$
- "Eventually a": $\forall \Diamond a \equiv \Diamond a$
- "Infinitely often a": $\forall \Box \forall \Diamond a \equiv \Box \Diamond a$ (slide 22)

What about e.g. $\forall \Diamond (a \wedge \forall \bigcirc a) \equiv \ldots$?

LTL and CTL are Incomparable

- Some LTL-formulas cannot be expressed in CTL, e.g.,
  - $\Diamond\Box\, a$
  - $\Diamond\,(a \wedge \bigcirc\, a)$

  There does not exist an equivalent CTL formula

- Some CTL-formulas cannot be expressed in LTL, e.g.,
  - $\forall\Diamond\forall\Box a$
  - $\forall\Diamond\,(a \wedge \forall\bigcirc\, a)$, and
  - $\forall\Box\exists\Diamond\, a$

  There does not exist an equivalent LTL formula

How to prove this formally?

From CTL to LTL

[Clarke & Draghicescu]

Let Φ be a CTL-formula, and φ the LTL-formula obtained by eliminating all path quantifiers in Φ. Then:

either $\Phi \equiv \varphi$ or there is no LTL-formula equivalent to Φ.

Examples

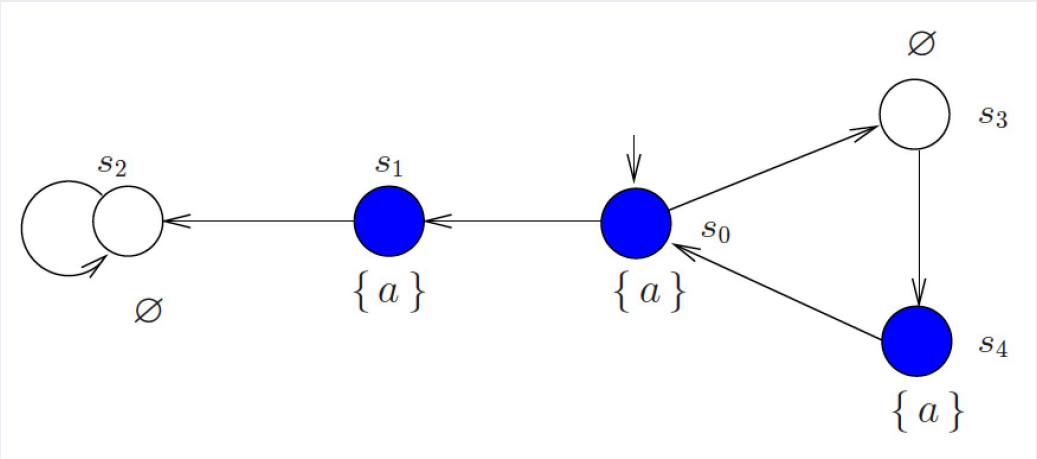
- $\forall \bigcirc a \mapsto \bigcirc a$ ✓
- $\forall \Diamond (a \wedge \forall \bigcirc a) \mapsto \Diamond (a \wedge \bigcirc a)$ ✗ (next slide)
- $\forall \Box \forall \Diamond a \mapsto \Box \Diamond a$ ✓ (inf. often a)
- $\forall \Box \exists \Diamond a \mapsto \Box \Diamond a$ ✗

From CTL To LTL (1)

CTL-formula $\forall\Diamond(a \wedge \forall\bigcirc a)$ cannot be expressed in LTL.

Proof.

We show that $\underbrace{\forall\Diamond(a \wedge \forall\bigcirc a)}_{\text{CTL-formula } \Phi} \not\equiv \underbrace{\Diamond(a \wedge \bigcirc a)}_{\Phi \text{ with path-quantifiers removed}}$.



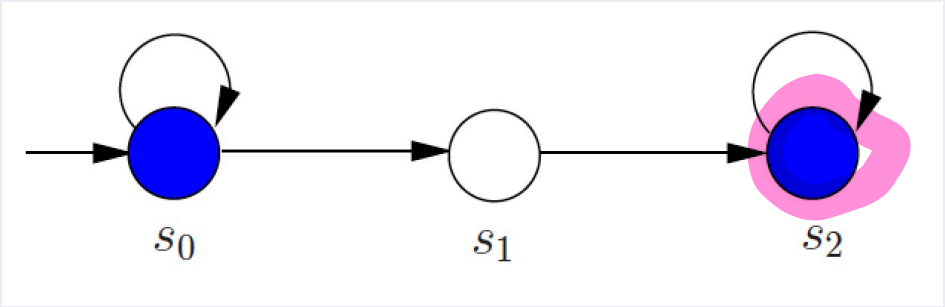
$s_0 \models \Diamond(a \wedge \bigcirc a)$ but $\underbrace{s_0 \not\models \forall\Diamond(a \wedge \forall\bigcirc a)}_{\text{path } s_0\, s_1\, (s_2)^\omega \text{ violates it}}$.

From CTL To LTL (2)

$\forall\Diamond\forall\Box a$ cannot be expressed in LTL.

Proof.

We show that: $\forall\Diamond\forall\Box a$ is not equivalent to $\Diamond\Box a$.



$s_0 \models \Diamond\Box a$ but $\underbrace{s_0 \not\models \forall\Diamond\forall\Box a}_{\text{path } s_0^\omega \text{ violates it}}$

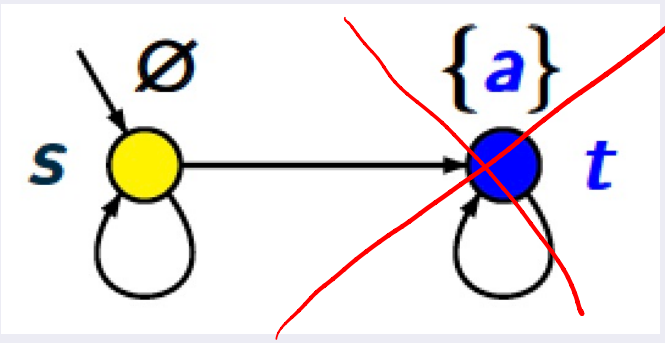
□

From CTL To LTL (3)

The CTL-formula $\forall\Box\exists\Diamond a$ cannot be expressed in LTL.

Proof.

- This is shown by contraposition: assume $\varphi \equiv \forall\Box\exists\Diamond a$; let TS:



- $TS \models \forall\Box\exists\Diamond a$, and thus—by assumption—$TS \models \varphi$
- Remove state t. Then: $Paths(TS') \subseteq Paths(TS)$, thus $TS' \models \varphi$
- But $TS' \not\models \forall\Box\exists\Diamond a$ as path $s^\omega \not\models \Box\exists\Diamond a$

□

From LTL To CTL

The LTL-formula $\Diamond\Box\, a$ cannot be expressed in CTL.

Proof.

Provide two series of transition systems TS_n and TS'_n for $n = 0, 1, 2, \ldots$ (see next slide) such that:

- $TS_n \not\models \Diamond\Box a$ and $TS'_n \models \Diamond\Box\, a$ (*), and
- For any CTL-formula Φ with $|\Phi| \leq n$: $TS_n \models \Phi$ iff $TS'_n \models \Phi$ (**)

  proof by induction on n (omitted here)

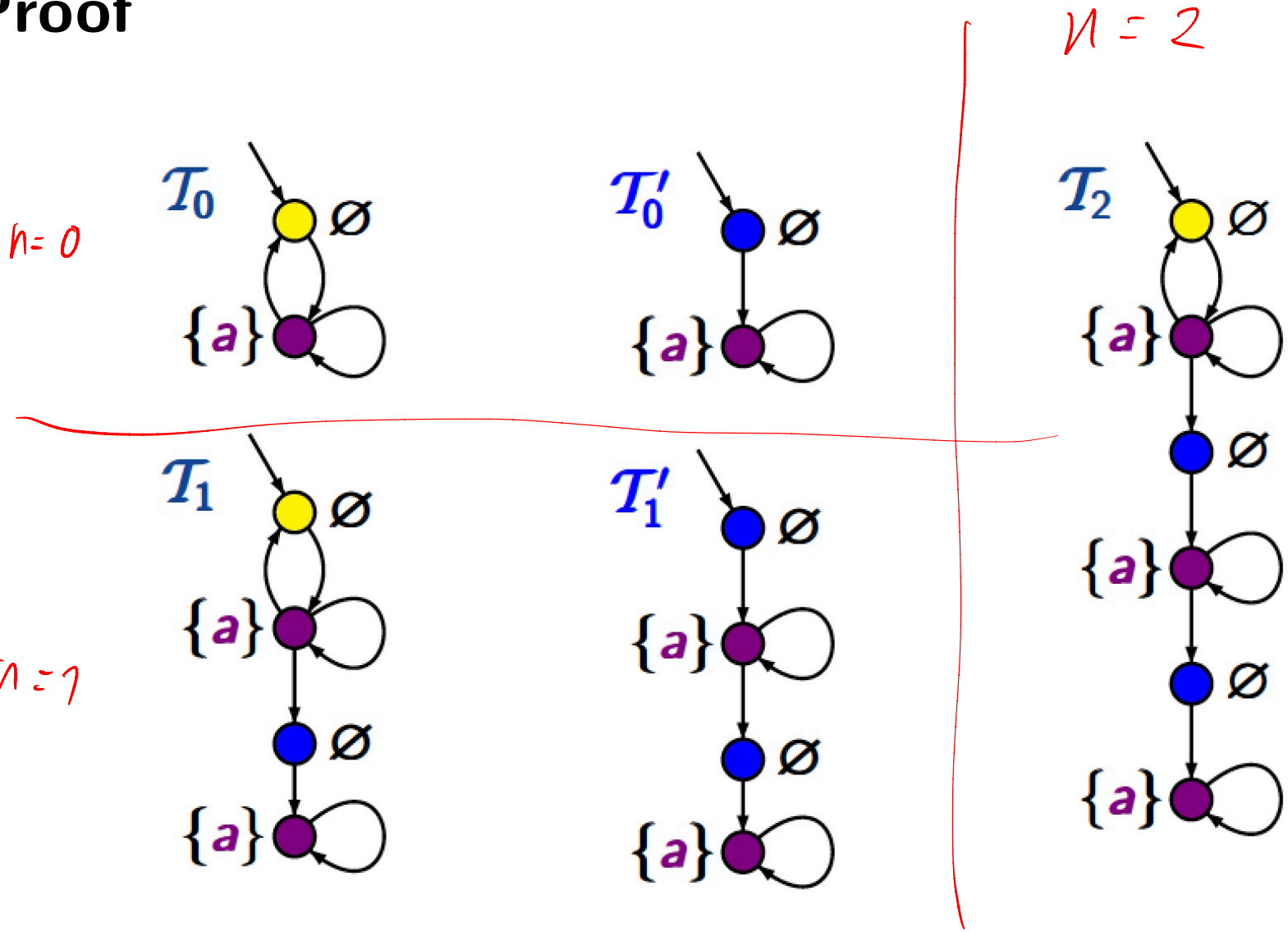
Proof by contraposition.

Assume there is a CTL-formula $\Phi \equiv \Diamond\Box a$ with $|\Phi| = n$ for some n

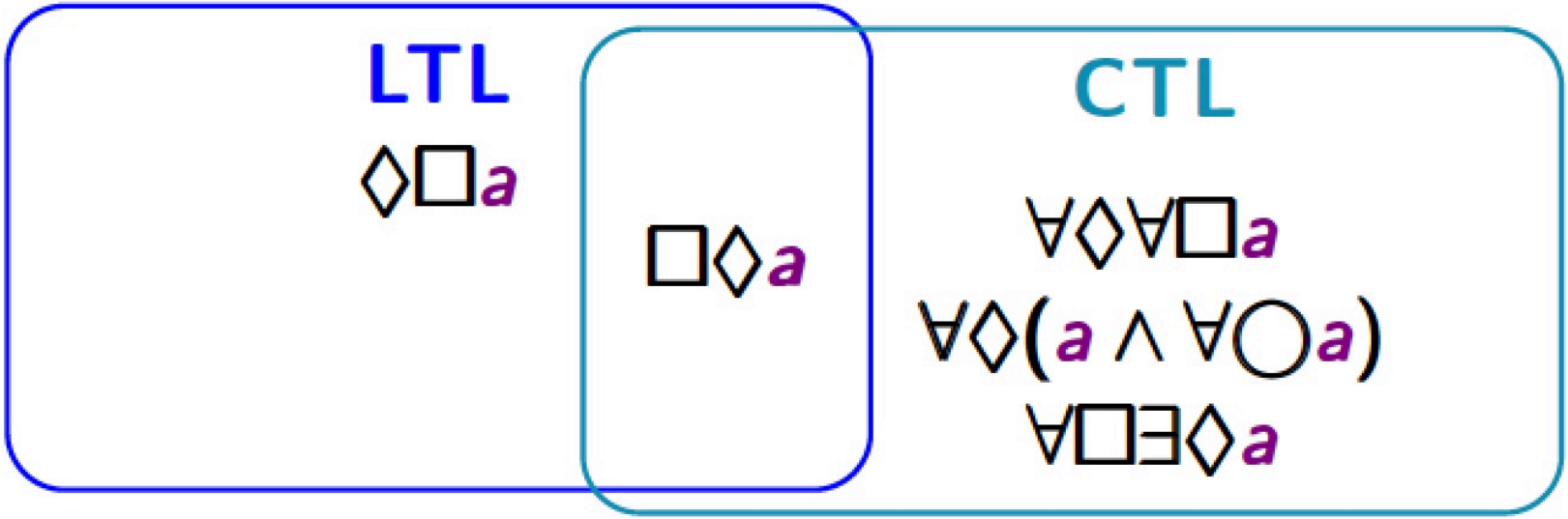
- by (*), it follows $TS_n \not\models \Phi$ and $TS'_n \models \Phi$
- but this contradicts (**): $TS_n \models \Phi$ if and only if $TS'_n \models \Phi$

□

Proof


n = 2
n = 0
n = 1
$\mathcal{T}_0$
$\mathcal{T}_0'$
$\mathcal{T}_1$
$\mathcal{T}_1'$
$\mathcal{T}_2$
∅
{a}

LTL Versus CTL



Overview

Syntax of CTL*

Definition: Syntax CTL*

- CTL* state-formulas with $a \in AP$ obey the grammar:

$$\Phi ::= \text{true} \mid a \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi \mid \exists\varphi$$

- and φ is a CTL* path-formula formed by the grammar:

$$\varphi ::= \Phi \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2$$

where Φ is a CTL* state-formula, and φ, φ_1 and φ_2 are path-formulas.

in CTL*: $\forall\varphi = \neg\exists\neg\varphi$. This does not hold in CTL.

CTL* Semantics

$s \models \text{true}$

$s \models a$ iff $a \in L(s)$

$s \models \Phi \wedge \Psi$ iff $(s \models \Phi)$ and $(s \models \Psi)$

$s \models \neg \Phi$ iff not $s \models \Phi$

$s \models \exists \varphi$ iff $\pi \models \varphi$ for some $\pi \in \mathit{Paths}(s)$

CTL* Semantics

$s \models \text{true}$

$s \models a$ iff $a \in L(s)$

$s \models \Phi \wedge \Psi$ iff $(s \models \Phi)$ and $(s \models \Psi)$

$s \models \neg \Phi$ iff not $s \models \Phi$

$s \models \exists \varphi$ iff $\pi \models \varphi$ for some $\pi \in \mathit{Paths}(s)$

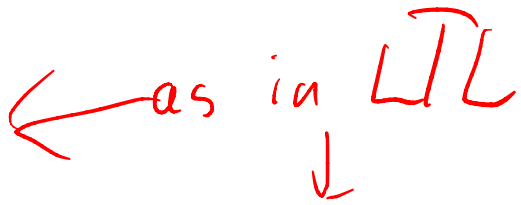
$\pi \models \Phi$ iff $\pi[0] \models \Phi$

$\pi \models \varphi_1 \wedge \varphi_2$ iff $\pi \models \varphi_1$ and $\pi \models \varphi_2$

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

$\pi \models \bigcirc \varphi$ iff $\pi[1..] \models \varphi$

$\pi \models \varphi_1 \,\mathsf{U}\, \varphi_2$ iff $\exists j \geq 0.\ (\pi[j..] \models \varphi_2 \wedge (\forall\, 0 \leq i < j.\ \pi[i..] \models \varphi_1))$



CTL* Transition System Semantics

- For CTL*-state-formula Φ, the satisfaction set $Sat(\Phi)$ is defined by:

$$Sat(\Phi) = \{ s \in S \mid s \vDash \Phi \}$$

- TS satisfies CTL*-formula Φ iff Φ holds in all its initial states:

$$TS \vDash \Phi \quad \text{if and only if} \quad \forall s_0 \in I.\, s_0 \vDash \Phi$$

This is exactly as for CTL

Embedding LTL

For LTL formula φ and TS without terminal states (both over AP) and for each $s \in S$:

$$\underbrace{s \models \varphi}_{\text{LTL semantics}} \quad \text{if and only if} \quad \underbrace{s \models \forall\varphi}_{\text{CTL}^* \text{ semantics}}$$

In particular:

$$TS \models_{LTL} \varphi \quad \text{if and only if} \quad TS \models_{CTL*} \forall\varphi$$

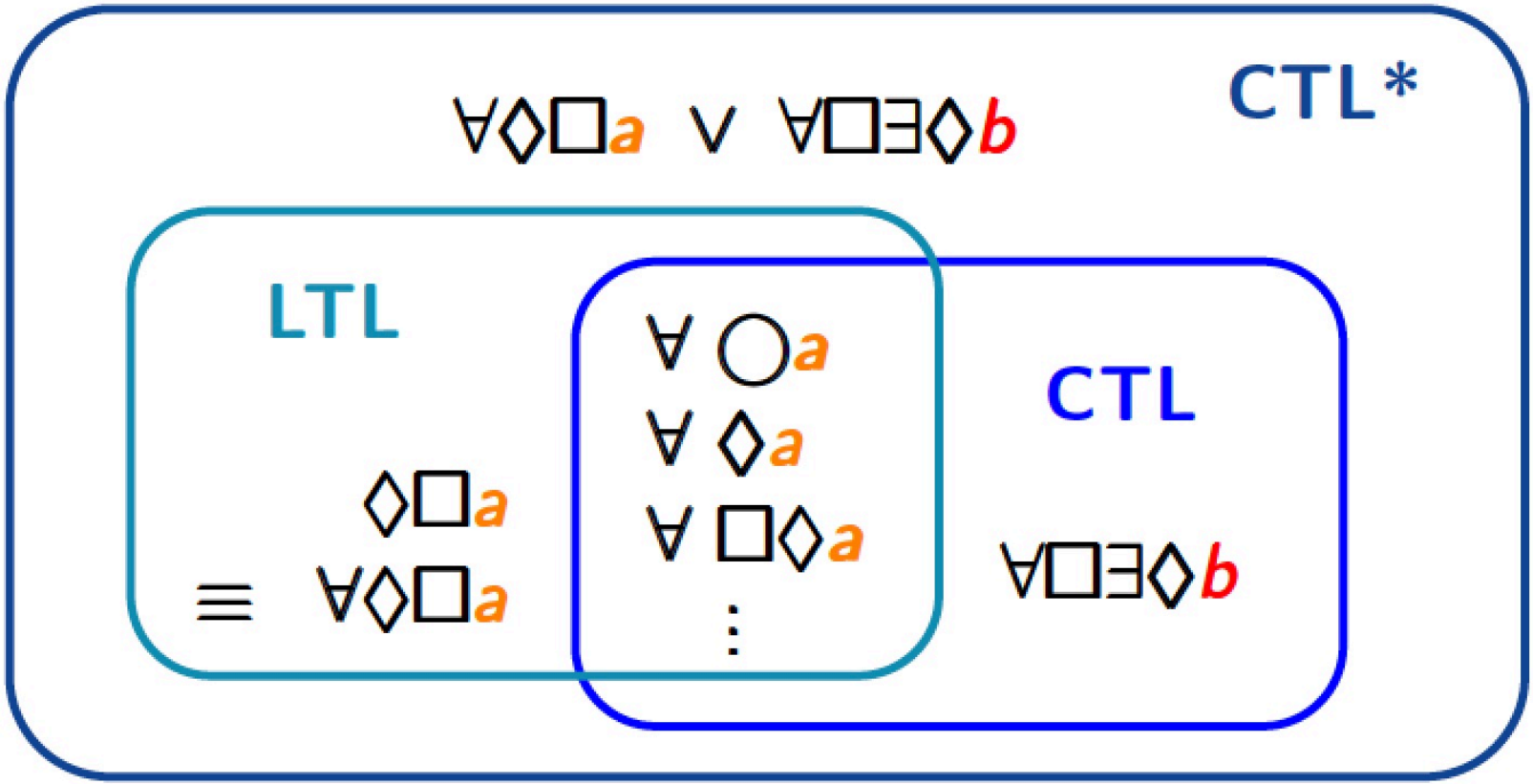
CTL* Is More Expressive Than LTL And CTL

The CTL*-formula over $AP = \{ a, b \}$:

$$\Phi \;=\; (\forall \Diamond \Box\, a) \;\vee\; (\forall \Box \exists \Diamond\, b)$$

can neither be expressed in LTL nor in CTL.

Relating LTL, CTL, and CTL*



Boolean Combinations of Path Formulas

Definition: Syntax CTL+

- CTL+ state-formulas with $a \in AP$ obey the grammar:

$$\Phi \ ::= \ \text{true} \ \Big| \ a \ \Big| \ \Phi_1 \wedge \Phi_2 \ \Big| \ \neg\Phi \ \Big| \ \exists\varphi \ \Big| \ \forall\varphi$$

- and φ is a CTL+ path-formula formed by the grammar:

$$\varphi \ ::= \ \varphi_1 \wedge \varphi_2 \ \Big| \ \neg\varphi \ \Big| \ \bigcirc\Phi \ \Big| \ \Phi_1 \,\mathsf{U}\, \Phi_2$$

as in CTL

where Φ, Φ_1 and Φ_2 are CTL+ state-formulas and φ_1 and φ_2 are CTL+ path-formulas.

Adding Boolean combinations of path formulae to CTL
does not change its expressiveness

but CTL^+ formulae can be much shorter than their equivalent in CTL

CTL+ Is As Expressive As CTL

For example:

$$\underbrace{\exists(\Diamond a \wedge \Diamond b)}_{\text{CTL}^+ \text{ formula}} \equiv \underbrace{\exists\Diamond\,(a \wedge \exists\Diamond\, b) \;\wedge\; \exists\Diamond\,(b \wedge \exists\Diamond\, a)}_{\text{CTL formula}}$$

Rules for transforming CTL+ formulas into equivalent CTL ones:

$$\begin{aligned}
\exists(\neg(\Phi_1 \,\mathsf{U}\, \Phi_2)) &\equiv \exists(\,(\Phi_1 \wedge \neg\Phi_2)\,\mathsf{U}\,(\neg\Phi_1 \wedge \neg\Phi_2)\,) \;\vee\; \exists\Box\neg\Phi_2 \\
\exists(\bigcirc\Phi_1 \wedge \bigcirc\Phi_2) &\equiv \exists\bigcirc(\Phi_1 \wedge \Phi_2) \\
\exists(\bigcirc\Phi \wedge (\Phi_1 \,\mathsf{U}\, \Phi_2)) &\equiv (\Phi_2 \wedge \exists\bigcirc\Phi) \;\vee\; (\Phi_1 \wedge \exists\bigcirc(\Phi \wedge \exists(\Phi_1 \,\mathsf{U}\, \Phi_2))) \\
\exists((\Phi_1 \,\mathsf{U}\, \Phi_2) \;\wedge\; (\Psi_1 \,\mathsf{U}\, \Psi_2)) &\equiv \exists((\Phi_1 \wedge \Psi_1)\,\mathsf{U}\,(\Phi_2 \wedge \exists(\Psi_1 \,\mathsf{U}\, \Psi_2))) \;\vee \\
&\qquad \exists((\Phi_1 \wedge \Psi_1)\,\mathsf{U}\,(\Psi_2 \wedge \exists(\Phi_1 \,\mathsf{U}\, \Phi_2))) \\
&\;\;\vdots
\end{aligned}$$

From CTL+ To CTL

$$\exists((a\,\mathbf{U}\,b)\ \wedge\ (c\,\mathbf{U}\,d)) \equiv \exists((a\wedge c)\,\mathbf{U}(b\wedge\exists(c\,\mathbf{U}\,d)))$$
$$\vee\ \exists((c\wedge a)\,\mathbf{U}(d\wedge\exists(a\,\mathbf{U}\,b)))$$

CTL+ formula

CTL formula

either $\{a,c\}\{a,c\}\{b,c\}\ \{c\}\ \{c\}\ \{d\}\ \cdots$

or $\{a,c\}\{a,c\}\{a,d\}\ \{a\}\ \{a\}\ \{b\}\ \cdots$

or $\{a,c\}\{a,c\}\{a,c\}\{a,c\}\{b,d\}\ \cdots$

Overview

Summary

- Computation tree logic (CTL) is a logic interpreted over infinite trees
- Path quantifiers in CTL alternate with temporal modalities

  ∀ ∃
- CTL and LTL have an incomparable expressive power
- A CTL-formula Φ is equivalent to:
  - the LTL-formula obtained by removing all path quantifiers from Φ, or
  - there is no equivalent LTL-formula
- Boolean combinations of path formulas do not raise CTL's expressive power
- CTL* is strictly more expressive than LTL and CTL

Next Lecture

Monday May 23, 10:30

No exercise class this Wednesday
No lecture this Thursday