Model Checking

Lecture #10: Computation Tree Logic

[Baier & Katoen, Chapter 6.1–6.3]

Joost-Pieter Katoen

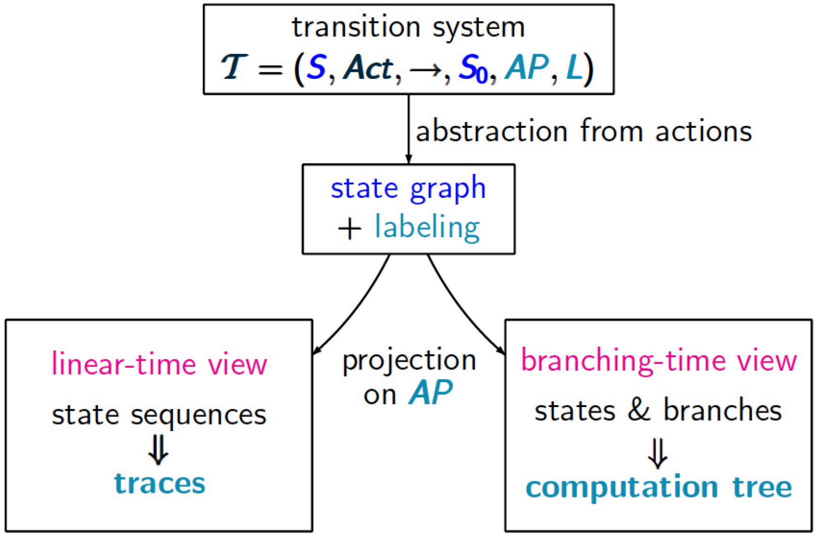
Software Modeling and Verification Group

Model Checking Course, RWTH Aachen, WiSe 2019/2020

Overview

Overview

Linear Time Versus Branching Time



Linear Time Versus Branching Time

- Linear Temporal Logic (LTL) is interpreted over $\underbrace{\text{infinite sequences}}_{\text{traces}}$
- Computation Tree Logic (CTL) is interpreted over $\underbrace{\text{infinite trees}}_{\text{computation trees}}$
- Traces are obtained from paths in a transition system.
- Computation trees are infinite trees whose nodes are labelled with sets of propositions
  - They are obtained by unfolding a transition system
  - Such tree contains several traces

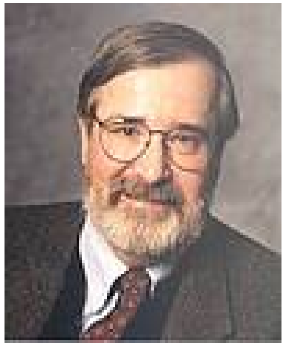
Linear Time Versus Branching Time

	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

Unfolding a Transition System

Overview

Computation Tree Logic



Edmund M. Clarke, Jr.
(1945–)

E. Allen Emerson
(1954–)

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:

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

Examples

$\forall\Box\exists\bigcirc a$ and $\exists(\forall\Box a) \, \mathsf{U} \, b$ are CTL formulas.

Intuition

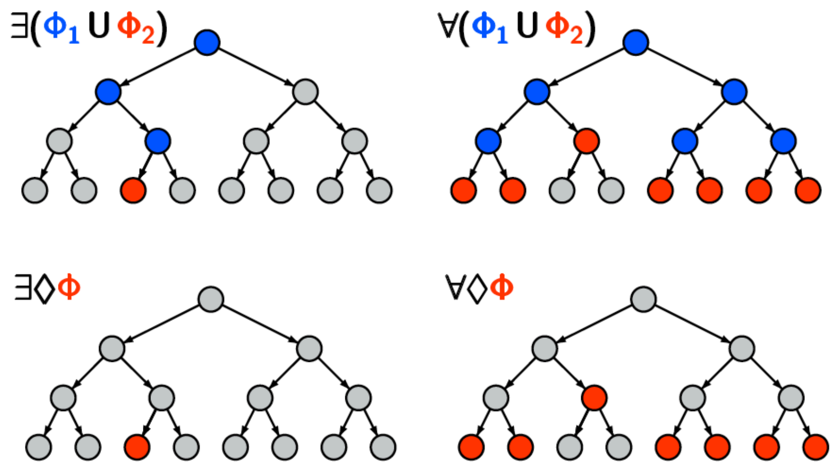
- $s \models \forall\varphi$ if all paths starting in s fulfill φ
- $s \models \exists\varphi$ if some path starting in s fulfill φ

Derived CTL Operators

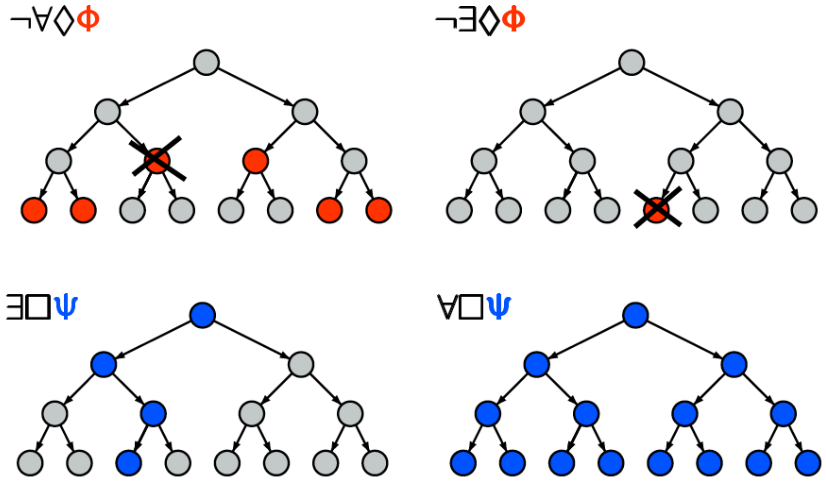
potentially Φ:	$\exists\Diamond\Phi$	$=$	$\exists(\text{true} \, \mathsf{U} \, \Phi)$
inevitably Φ:	$\forall\Diamond\Phi$	$=$	$\forall(\text{true} \, \mathsf{U} \, \Phi)$
potentially always Φ:	$\exists\Box\Phi$	$:=$	$\neg\forall\Diamond\neg\Phi$
invariantly Φ:	$\forall\Box\Phi$	$=$	$\neg\exists\Diamond\neg\Phi$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

The Boolean connectives are derived as usual

Intuitive CTL Semantics



Intuitive CTL Semantics



Example CTL Formulas

Overview

CTL Semantics

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

- Interpretation of state-formulas over states of *TS*
- Interpretation of path-formulas over paths of *TS*

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:

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

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:

$$\begin{array}{ll} \pi \models \bigcirc\Phi & \text{iff } \pi[1] \models \Phi \\ \pi \models \Phi \,\mathsf{U}\, \Psi & \text{iff } (\exists j \geq 0.\ \pi[j] \models \Psi \text{ and } (\forall\, 0 \leq i < j.\ \pi[i] \models \Phi)) \end{array}$$

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

Transition System Semantics

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

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

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

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

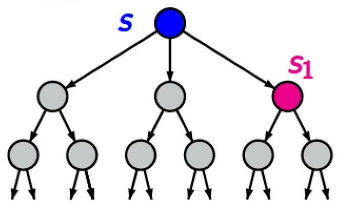
- Point of attention: $TS \not\models \Phi$ is not equivalent to $TS \not\models \neg\Phi$
  because of several initial states, e.g., $s_0 \models \exists\Box\,\Phi$ and $s_0' \not\models \exists\Box\,\Phi$

Semantics of $\bigcirc$-Operator

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

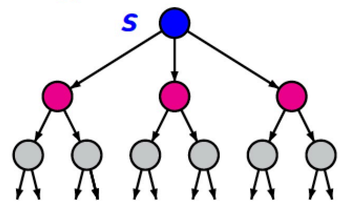
$$s \models \forall\bigcirc\Phi \quad \text{iff} \quad \forall\pi = s\, s_1\, s_2 \ldots \in Paths(s).\ \pi \models \bigcirc\Phi, \text{that is },\ s_1 \models \Phi$$

$\exists\bigcirc\Phi$



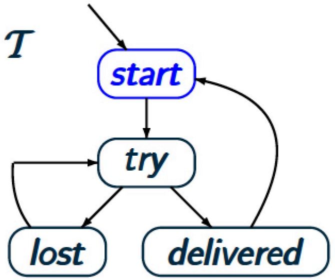
$Post(s) \cap Sat(\Phi) \neq \varnothing$

$\forall\bigcirc\Phi$



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

Example



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

CTL formula

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

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

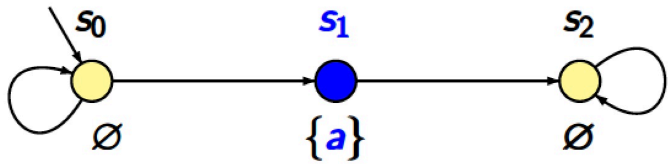
Infinitely Often

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

Proof.

□

Example



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

Overview

CTL Equivalence

Definition: CTL equivalence

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

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

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

Equivalently,

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

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

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

$s \models \forall \Diamond (a \vee b)$ since for all $\pi \in Paths(s). \pi \models \Diamond (a \vee b)$. But: $s\,(s'')^\omega \models \Diamond\, a$ and $s\,(s'')^\omega \not\models \Diamond\, b$. Thus: $s \not\models \forall \Diamond b$. A similar reasoning applied to path $s\,(s')^\omega$ yields $s \not\models \forall \Diamond\, a$ Thus, $s \not\models \forall \Diamond\, a \vee \forall \Diamond b$.

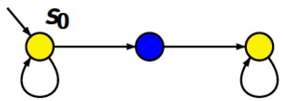
Duality $\bigcirc$ and $\Box$

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

correct.

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

wrong, e.g.,



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

$s_0 \models \exists \Box \exists \bigcirc a$

$s_0 \models \exists \bigcirc a$

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

$\Longrightarrow\; s_0 \models \exists \Box \exists \bigcirc a$

Expansion Laws

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

CTL expansion laws

For any CTL-formula Φ and Ψ:

$$\begin{aligned}
\forall(\Phi \mathsf{U} \Psi) &\equiv \Psi \vee (\Phi \wedge \forall\bigcirc \forall(\Phi \mathsf{U} \Psi)) \\
\forall\Diamond \Phi &\equiv \Phi \vee \forall\bigcirc \forall\Diamond \Phi \\
\forall\Box \Phi &\equiv \Phi \wedge \forall\bigcirc \forall\Box \Phi \\
\\
\exists(\Phi \mathsf{U} \Psi) &\equiv \Psi \vee (\Phi \wedge \exists\bigcirc \exists(\Phi \mathsf{U} \Psi)) \\
\exists\Diamond \Phi &\equiv \Phi \vee \exists\bigcirc \exists\Diamond \Phi \\
\exists\Box \Phi &\equiv \Phi \wedge \exists\bigcirc \exists\Box \Phi
\end{aligned}$$

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

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

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
  - $\forall\Box\exists\Diamond\, a$

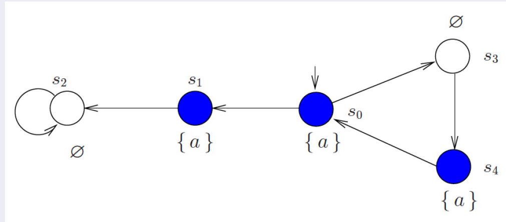
  There does not exist an equivalent LTL formula

From CTL To LTL (1)

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

Proof.

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



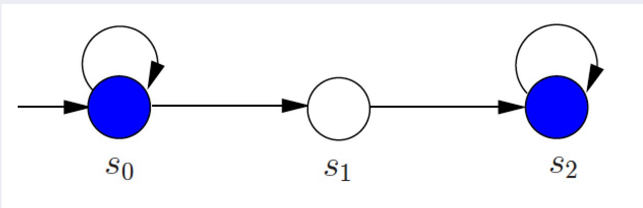
$s_0 \models \Diamond(a \wedge \bigcirc a)$ but $\underbrace{s_0 \not\models \forall\Diamond(a \wedge \forall\bigcirc a)}_{\text{path } s_0\, s_1\, (s_2)^\omega \text{ violates it}}$.

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

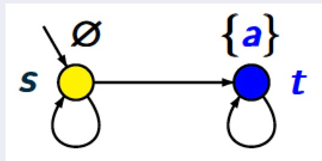
□

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

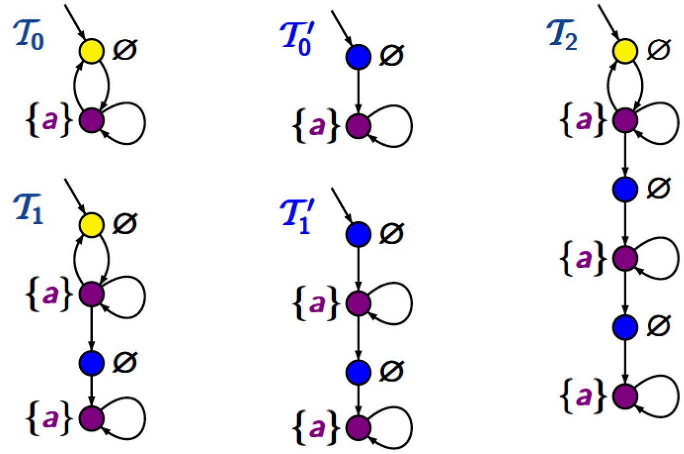
The LTL-formula $\Diamond\Box\ a$ cannot be expressed in CTL.

Proof.

- Provide two series of transition systems TS_n and $\widehat{TS}_n$
- Such that $TS_n \not\models \Diamond\Box\ a$ and $\widehat{TS}_n \models \Diamond\ \Box\ a$ (*), and
- for any $\forall$CTL-formula Φ with $|\Phi| \leq n$: $TS_n \models \Phi$ iff $\widehat{TS}_n \models \Phi$ (**)
  - proof by induction on n (omitted here)
- Assume there is a CTL-formula $\Phi \equiv \Diamond\Box\ a$ with $|\Phi| = n$
  - by (*), it follows $TS_n \not\models \Phi$ and $\widehat{TS}_n \models \Phi$
  - but this contradicts (**): $TS_n \models \Phi$ if and only if $\widehat{TS}_n \models \Phi$

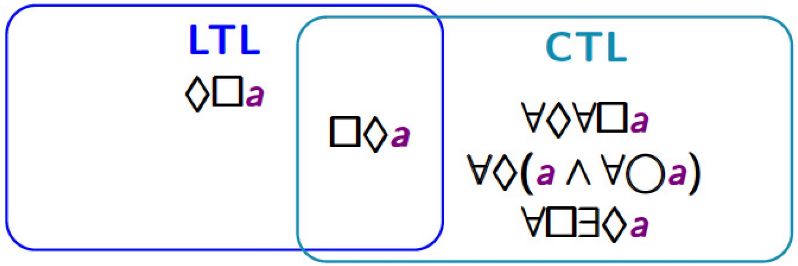
$\square$

Proof



Proof

LTL Versus CTL



Overview

Syntax of CTL*

Definition: Syntax CTL*

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

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

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

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

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 $\pi \in Paths(s)$

$\pi \models \Phi$	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	$\exists j \geq 0. (\pi[j..] \models \varphi_2 \wedge (\forall 0 \leq k < j. \pi[k..] \models \varphi_1))$

CTL* Transition System Semantics

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

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

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

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

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:

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

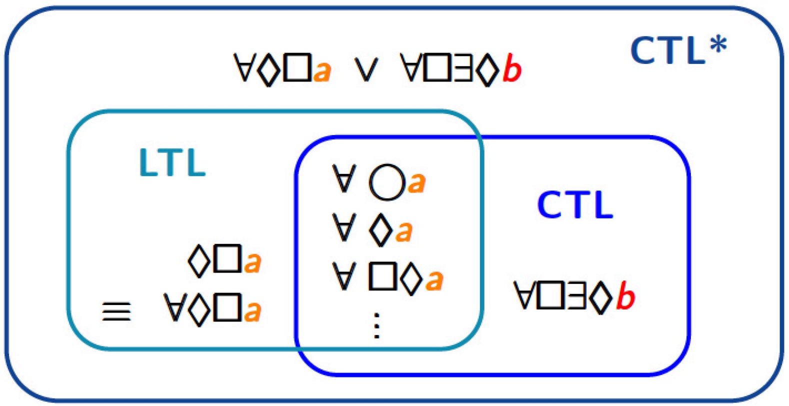
CTL* Is More Expressive Than LTL And CTL

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

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

cannot be expressed in neither LTL nor CTL.

Relating LTL, CTL, and CTL*



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

but yields formulae can be much shorter than their equivalent in 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:

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

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

CTL$^+$ Is As Expressive As CTL

For example:

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

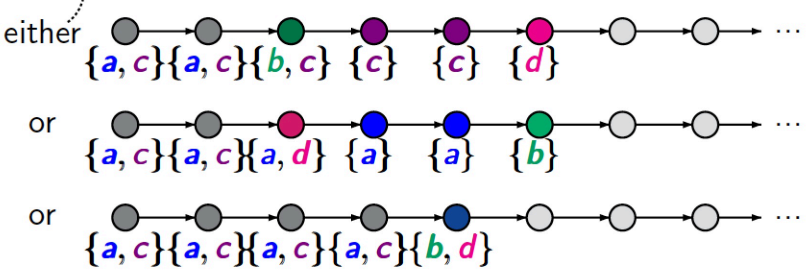
Rules for transforming CTL$^+$ formulas into equivalent CTL ones:

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

From CTL$^+$ To CTL

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

CTL$^+$ formula **CTL** formula



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
- Adding Boolean combinations of path formulas does not raise expressive power of CTL
- CTL^* is strictly more expressive than LTL and CTL

Next Lecture

Friday November 22, 14:30