Overview

1. Branching-Time Logic
2. CTL Syntax
3. CTL Semantics
4. CTL Equivalence
5. Expressiveness of LTL versus CTL
6. CTL* and CTL+
7. Summary

Linear Time Versus Branching Time

transition system

\[ T = (S, Act, \rightarrow, S_0, AP, L) \]

abstraction from actions

state graph
+ labeling

linear-time view
state sequences
↓ traces

projection on \( AP \)

branching-time view
states & branches
↓ computation tree
Linear Time Versus Branching Time

- **Linear Temporal Logic (LTL)** is interpreted over infinite sequences of traces.
- **Computation Tree Logic (CTL)** is interpreted over infinite trees of 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.

Unfolding a Transition System

Overview

1. Branching-Time Logic
2. CTL Syntax
3. CTL Semantics
4. CTL Equivalence
5. Expressiveness of LTL versus CTL
6. $\text{CTL}^*$ and $\text{CTL}^+$
7. Summary
Computation Tree Logic

Edmund M. Clarke, Jr. (1945–)
E. Allen Emerson (1954–)

Definition: Syntax Computation Tree Logic

- CTL state-formulas with $a \in AP$ obey the grammar:
  $$\Phi ::= true \mid a \mid \Phi_1 \land \Phi_2 \mid \neg\Phi \mid \exists \phi \mid \forall \phi$$
- and $\phi$ is a path-formula formed by the grammar:
  $$\phi ::= \bigcirc \Phi \mid \Phi_1 U \Phi_2.$$  

Examples

$$\forall \bigcirc \exists \bigcirc a$$ and $$\exists (\forall \bigcirc a) U b$$ are CTL formulas.

Intuition

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

Derived CTL Operators

- potentially $\phi$: $\exists \bigcirc \phi = \exists (true U \phi)$
- inevitably $\phi$: $\forall \bigcirc \phi = \forall (true U \phi)$
- potentially always $\phi$: $\exists \square \phi := \neg \forall \bigcirc \neg \phi$
- invariantly $\phi$: $\forall \square \phi = \neg \exists \bigcirc \neg \phi$
- weak until: $\exists(\phi W \psi) = \neg \forall((\phi \land \neg \psi) U (\neg \phi \land \neg \psi))$
- invariantly until: $\forall(\phi W \psi) = \neg \exists((\phi \land \neg \psi) U (\neg \phi \land \neg \psi))$

The Boolean connectives are derived as usual
Intuitive CTL Semantics

Overview

CTL Syntax

CTL Semantics

Example CTL Formulas

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 $\Phi$ 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:

- $s \models a$ iff $a \in L(s)$
- $s \models \neg \Phi$ iff $(s \not\models \Phi)$
- $s \models \Phi \land \Psi$ iff $(s \models \Phi) \text{ and } (s \models \Psi)$
- $s \models \exists \varphi$ iff there exists $\pi \in \text{Paths}(s)$. $\pi \models \varphi$
- $s \models \forall \varphi$ iff for all $\pi \in \text{Paths}(s)$. $\pi \models \varphi$

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

--

Transition System Semantics

- For CTL-state-formula $\Phi$, the satisfaction set $\text{Sat}(\Phi)$ is defined by:
  
  $$\text{Sat}(\Phi) = \{ s \in S \mid s \models \Phi \}$$

- $TS$ satisfies CTL-formula $\Phi$ iff $\Phi$ holds in all its initial states:
  
  $$TS \models \Phi \text{ if and only if } \forall s_0 \in I. s_0 \models \Phi$$

- 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 $s_0 \not\models \Box \exists \Phi$

CTL Semantics (2)

**Definition: Satisfaction relation for CTL path-formulas**

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

- $\pi \models \Box \Phi$ iff $\pi[1] \models \Phi$
- $\pi \models \Phi \lor \Psi$ iff $(\exists j \geq 0. \pi[j] \models \Psi \text{ and } (\forall 0 \leq i < j. \pi[i] \models \Phi))$

where $\pi[j]$ denotes the state $s_j$ in the path $\pi = s_0 s_1 s_2 \ldots$.

**Semantics of $\Box$-Operator**

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

---

**Postulates**

$$\text{Post}(s) \cap \text{Sat}(\Phi) \neq \emptyset$$

$$\text{Post}(s) \subseteq \text{Sat}(\Phi)$$
Example

\[
T \not\vDash \forall \square \forall \diamond \text{start}
\]

\[
T \models \forall \square \forall \diamond \text{start} \iff \forall \diamond (\text{start} \lor \text{delivered})
\]

\[
\text{Sat}(\forall \diamond \text{start}) = \{\text{start}, \text{delivered}\}
\]

\[
\text{Sat}(\phi) = \emptyset
\]

Example

(1) Does \( TS \vDash \exists \diamond \forall \square \neg a \)?
(2) Does \( TS \vDash \forall \square \exists \diamond \neg a \)?

Infinitely Often

\[
s \models \forall \square \forall \diamond \phi \iff \forall \pi \in \text{Paths}(s) \text{ an } \phi\text{-state is visited infinitely often.}
\]

Proof.

Overview

1. Branching-Time Logic
2. CTL Syntax
3. CTL Semantics
4. CTL Equivalence
5. Expressiveness of LTL versus CTL
6. CTL* and CTL+
7. Summary
**CTL Equivalence**

**Definition: CTL equivalence**

CTL-formulas $\Phi$ and $\Psi$ (both over $AP$) are **equivalent**:

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

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

Equivalently,

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

---

**Duality**

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

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

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

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

$\forall (\Phi \lor \Psi) \equiv \neg \exists ((\Diamond \Phi \land \neg \Psi) \lor (\neg \Diamond \Phi \land \neg \Psi))$

---

**Distributive Laws**

$$\forall (\Phi \land \Psi) \equiv \forall \Phi \land \forall \Psi$$

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

But: $\exists \Diamond (\Phi \land \Psi) \neq \exists \Diamond \Phi \land \exists \Diamond \Psi$

$$\forall \Diamond (\Phi \lor \Psi) \neq \forall \Diamond \Phi \lor \forall \Diamond \Psi$$

---

$s \models \forall \Diamond (a \lor b)$ since for all $\pi \in Paths(s)$, $\pi \models \Diamond (a \lor b)$. But: $s(s'') \models \Diamond a$ and $s(s'') \models \Diamond b$. Thus: $s \nmodels \forall \Diamond a$. A similar reasoning applied to path $s(s'')$ yields $s \nmodels \forall \Diamond a$. Thus, $s \nmodels \forall \Diamond a \lor \forall \Diamond b$. 

---

**Duality $\bigcirc$ and $\Box$**

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

**correct.**

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

**wrong, e.g.,**

$s_0 \nmodels \exists \exists \Box a$

$s_0 \models \exists \Box a$

$s_0 \models \exists \Box a$

$s_0 \models \exists \Box a$
Expansion Laws

Recall in LTL: \( \varphi U \psi \equiv \psi \lor (\varphi \land O(\varphi U \psi)) \)

**CTL expansion laws**

For any CTL-formula \( \Phi \) and \( \Psi \):

- \( \forall (\Phi U \Psi) \equiv \Psi \lor (\Phi \land \forall O \forall (\Phi U \Psi)) \)
- \( \forall \square \Phi \equiv \Phi \land \forall O \forall \square \Phi \)
- \( \exists (\Phi U \Psi) \equiv \Psi \lor (\Phi \land \exists O \exists (\Phi U \Psi)) \)
- \( \exists \diamond \Phi \equiv \Phi \land \exists O \exists \diamond \Phi \)
- \( \exists \square \Phi \equiv \Phi \land \exists O \exists \square \Phi \)

Overview

1. Branching-Time Logic
2. CTL Syntax
3. CTL Semantics
4. CTL Equivalence
5. Expressiveness of LTL versus CTL
6. CTL* and CTL+
7. Summary

Equivalence of CTL and LTL Formulas

**Definition: equivalence of LTL and CTL formulas**

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

\[ TS \models \Phi \text{ if and only if } TS \not\models \varphi. \]

Examples

From CTL to LTL

Let \( \Phi \) be a CTL-formula, and \( \varphi \) the LTL-formula obtained by eliminating all path quantifiers in \( \Phi \). Then:

- either \( \Phi \equiv \varphi \) or there is no LTL-formula equivalent to \( \Phi \).
LTL and CTL are Incomparable

- Some LTL-formulas cannot be expressed in CTL, e.g.,
  - $\Diamond \Box a$
  - $\Diamond (a \land \Diamond a)$
  There does not exist an equivalent CTL formula

- Some CTL-formulas cannot be expressed in LTL, e.g.,
  - $\forall \Diamond \forall \Box a$
  - $\forall \Diamond (a \land \forall \Box a)$, and
  - $\forall \Box \exists \Diamond a$
  There does not exist an equivalent LTL formula

From CTL To LTL (1)

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

Proof.

We show that $\forall \Diamond (a \land \forall \Box a) \not\equiv \Diamond (a \land \Diamond a)$.

Path $s_0 \not\models \Diamond (a \land \Diamond a)$ but $s_0 \not\models \forall \Diamond (a \land \forall \Box a)$.

From CTL To LTL (2)

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

Proof.

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

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 $s^\omega$ violates it.
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 $\overline{TS}_n$.
- Such that $TS_n \not\models \Diamond \Box a$ and $\overline{TS}_n \models \Diamond \Box a$ (*), and
- for any $\forall$CTL-formula $\Phi$ with $|\Phi| \leq n : TS_n \models \Phi$ iff $\overline{TS}_n \models \Phi$ (**)
- 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 $\overline{TS}_n \models \Phi$
  - but this contradicts (**): $TS_n \models \Phi$ if and only if $\overline{TS}_n \models \Phi$

Proof

LTL Versus CTL
Overview

- Branching-Time Logic
- CTL Syntax
- CTL Semantics
- CTL Equivalence
- Expressiveness of LTL versus CTL
- CTL* and CTL+  

Summary

Syntax of CTL*

**Definition: Syntax CTL**

- **CTL** state-formulas with $a \in AP$ obey the grammar:
  \[
  \Phi ::= \text{true} \mid a \mid \Phi \land \Phi \mid \neg \Phi \mid \exists \varphi
  \]

- and $\varphi$ is a **CTL** path-formula formed by the grammar:
  \[
  \varphi ::= \Phi \mid \varphi \land \varphi \mid \neg \varphi \mid \bigcirc \varphi \mid \varphi_1 \lor \varphi_2
  \]

where $\Phi$ is a **CTL** state-formula, and $\varphi$, $\varphi_1$ and $\varphi_2$ 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 $a \in L(s)$
- $s \models \Phi \land \Psi$ 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 $\pi \in \text{Paths}(s)$

- $\pi \models \Phi$ iff $\pi[0] \models \Phi$
- $\pi \models \varphi_1 \land \varphi_2$ iff $\pi \models \varphi_1$ and $\pi \models \varphi_2$
- $\pi \models \neg \varphi$ iff $\pi \not\models \varphi$
- $\pi \models \bigcirc \varphi$ iff $\pi[1..] \models \varphi$
- $\pi \models \varphi_1 \lor \varphi_2$ iff $\exists j \geq 0. (\pi[j..] \models \varphi_2 \land (\forall 0 \leq k < j. \pi[k..] \models \varphi_1))$

CTL* Transition System Semantics

- For **CTL***-state-formula $\Phi$, the satisfaction set $\text{Sat}(\Phi)$ is defined by:
  \[ \text{Sat}(\Phi) = \{ s \in S \mid s \models \Phi \} \]

- $TS$ satisfies **CTL***-formula $\Phi$ iff $\Phi$ holds in all its initial states:
  \[ TS \models \Phi \text{ if and only if } \forall s_0 \in I. s_0 \models \Phi \]

This is exactly as for **CTL**.
Embedding LTL

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

$$s \models LTL \varphi \iff s \models CTL^* \forall \varphi$$

In particular:

$$TS \models LTL \varphi \iff TS \models CTL^* \forall \varphi$$

CTL* Is More Expressive Than LTL And CTL

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

$$\Phi = (\forall \diamond a) \lor (\forall \exists \lozenge b)$$

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:
  $Φ ::= \text{true} \mid a \mid Φ_1 \land Φ_2 \mid \neg Φ \mid \exists φ \mid ∀ φ$

- and $φ$ is a CTL+ path-formula formed by the grammar:
  $φ ::= φ_1 \land φ_2 \mid \neg φ \mid ◯ Φ \mid Φ_1 U Φ_2$

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:

\[ \exists (\Diamond a \land \Diamond b) \equiv \exists (a \land \exists \Diamond b) \land \exists (b \land \exists \Diamond a) \]

Rules for transforming CTL+ formulas into equivalent CTL ones:

- $\exists (\neg (Φ_1 U Φ_2)) \equiv \exists ((Φ_1 \land \neg Φ_2) U (\neg Φ_1 \land \neg Φ_2)) \lor \exists \Box \neg Φ_2$
- $\exists (\Box Φ_1 \land \Box Φ_2) \equiv \exists (Φ_1 \land \Box Φ_2)$
- $\exists (\Box Φ \land (Φ_1 U Φ_2)) \equiv (Φ_2 \land \exists \Box Φ) \lor (Φ_1 \land \exists \Box (Φ \land \exists Φ_1 U Φ_2))$
- $\exists ((Φ_1 U Φ_2) \land (Ψ_1 U Ψ_2)) \equiv (Φ_1 \land Ψ_1) U (Φ_2 \land \exists Ψ_1 U Ψ_2)) \lor$
  $\exists ((Φ_1 \land Ψ_1) U (Ψ_2 \land (Φ_1 U Φ_2)))$

From CTL+ To CTL

\[ \exists ((a U b) \land (c U d)) \equiv \exists ((a \land c) U (b \land d)) \lor \exists ((c \land a) U (d \land a U b)) \]

CTL+ and CTL∗

Summary

1. Branching-Time Logic
2. CTL Syntax
3. CTL Semantics
4. CTL Equivalence
5. Expressiveness of LTL versus CTL
6. CTL* and CTL+
7. 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 $\Phi$ is equivalent to:
- the LTL-formula obtained by removing all path quantifiers from $\Phi$, 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.