Learning outcomes

By the end of this lecture, you should:

- Be familiar with the branching model of time
- Be familiar with CTL syntax and semantics
- Understand CTL semantic equivalence, and why it is important
- Be familiar with important CTL equivalences
- Be familiar with Existential Normal Form

Branching model of time

CTL’s conception of time:

- At each moment in time exactly potentially multiple futures
- Time “branches” into multiple futures at each state
- Quantify over possible futures

CTL therefore describes “state properties” of systems

CTL formulae describe states in transition system
A note on models

Note: by changing model of time, not changed underlying model

CTL models are based on right-serial transition systems, same as LTL

Changing conception of time:

- Affects properties that can be expressed by formulae
- Affects what CTL formulae describe (states, not paths)

CTL syntax

**Atomic propositions**

Like in LTL, we fix a set $AP$ of atomic propositions

We continue to use $p$, $q$, $r$, and so on to range over $AP$

**CTL state and path formulae**

Define **state formulae** with the following grammar:

$$
\Phi, \Psi, \Xi ::= T \mid \bot \mid p \\
\quad ::= \neg \Phi \\
\quad ::= \Phi \land \Psi \mid \Phi \lor \Psi \mid \Phi \Rightarrow \Psi \\
\quad ::= \forall \phi \mid \exists \phi
$$

and **path formulae** with the following grammar:

$$
\phi, \psi, \xi ::= \Box \Phi \mid \Diamond \Phi \mid \Phi \text{ UNTIL } \Psi
$$

In semantics of CTL:

- Path formulae are evaluated relative to a path
- State formulae are evaluated relative to a state
Intuitive explanation of CTL formulae

First line (of state formula grammar):

\[ \top \mid \bot \mid p \]

\( \top, \bot, \) and \( p \) for \( p \) atomic are all primitive CTL state formulae

- \( \top \) is the **logical truth** constant (or “true”),
- \( \bot \) is the **logical falsity** constant (or “false”),
- \( p \) is the embedding of **atomic propositions** into CTL formulae

The last should now be familiar too!

Intuitive explanation of CTL formulae

Third line (of state formula grammar):

\[ \Phi \land \Psi \mid \Phi \lor \Psi \mid \Phi \Rightarrow \Psi \]

If \( \Phi \) and \( \Psi \) are CTL state formulae, then so are \( \Phi \land \Psi, \Phi \lor \Psi, \Phi \Rightarrow \Psi \)

- \( \Phi \land \Psi \) is **conjunction** (or “\( \Phi \) and \( \Psi \)”)
- \( \Phi \lor \Psi \) is **disjunction** (or “\( \Phi \) or \( \Psi \)”)
- \( \Phi \Rightarrow \Psi \) is **implication** (or “if \( \Phi \) then \( \Psi \)”, or “\( \psi \) whenever \( \phi \)”)

Intuitive explanation of CTL formulae

Last line (of state formula grammar):

\[ \forall \phi \mid \exists \phi \]

If \( \phi \) and \( \psi \) are CTL path formulae, then so are \( \forall \phi, \exists \phi \)

- \( \forall \phi \) is “\( \phi \) along every path that starts here”
- \( \exists \phi \) is “\( \phi \) along at least one path that starts here”, or “there exists a path where \( \phi \) holds”

Specific to CTL!
Intuitive explanation of CTL formulae

Path formula grammar:

\[ \bigcirc \Phi \mid \square \Phi \mid \diamond \Phi \mid \Phi \text{ UNTIL } \Psi \]

If \( \Phi \) and \( \Psi \) are CTL state formulae, then \( \bigcirc \Phi \), \( \square \Phi \), \( \diamond \Phi \), and \( \Phi \text{ UNTIL } \Psi \) are CTL path formulae

- \( \square \Phi \) is “henceforth \( \Phi \)”, or “from now, always \( \Phi \)”
- \( \diamond \Phi \) is “at some future point \( \Phi \)”
- \( \bigcirc \Phi \) is “immediately after \( \Phi \)”, or “in the next state \( \Phi \)”
- \( \Phi \text{ UNTIL } \Psi \) is “at some future point \( \Psi \), but until then \( \Phi \)”

Alternative syntax for modalities

Some collapse grammar of CTL into a single grammar of “formulae”

Less clear (to me, anyway) what is going on:

- \( \forall \) and \( \exists \) are instructions: “go off and examine paths”
- Path formulae evaluated relative to paths
- State formulae relative to states
- Grammar closer to grammar of CTL

Might also see (e.g. in “Logic in Computer Science”):

- \( A \) and \( E \) instead of \( \forall \) and \( \exists \)
- \( X, G, F, \) and \( U \) instead of \( \bigcirc, \square, \diamond, \) and \( \text{ UNTIL } \)

Operator precedence

We add parentheses freely to disambiguate

Assign precedence to reduce number of parentheses needed:

- Unary: \( \neg, \forall, \exists, \square, \diamond \), and \( \bigcirc \) bind most tightly
- After that \( \text{ UNTIL } \)
- After that \( \vee \) and \( \wedge \)
- Finally \( \Rightarrow \) binds least tightly

Grammar above enforces path formula be “covered” by quantifier

Impossible to construct \( \square \forall \phi \) or \( \exists \phi \text{ UNTIL } \Psi \)

Effect is to have

\[ \forall \square \phi \quad \exists \square \phi \quad \forall \diamond \phi \quad \exists \diamond \phi \]

\[ \forall \bigcirc \phi \quad \exists \bigcirc \phi \quad \forall (\phi \text{ UNTIL } \Psi) \quad \exists (\phi \text{ UNTIL } \Psi) \]

\( \forall \square, \exists \bigcirc \), and so on, are “derived modalities”
Precedence examples

So:

\[ \Phi \Rightarrow \forall \psi \] means \[ \Phi \Rightarrow (\forall (\psi)) \]

\[ \Phi \Rightarrow \psi \lor \exists \psi \] means \[ \Phi \Rightarrow (\Phi \lor (\exists (\psi))) \]

\[ \forall \diamond \phi \lor \exists \psi \Rightarrow \psi \text{ UNTIL } \Xi \] means \[ ((\forall (\phi)) \lor \exists) \Rightarrow (\psi \text{ UNTIL } \Xi) \]

and so on...

Example CTL formulae

Suppose \textit{started} and \textit{ready} are atomic propositions, then:

\[ \exists \diamond (\textit{started} \land \neg \textit{ready}) \]

can be read as:

\textit{it is possible to get to a state where “started” holds but “ready” does not}

Example CTL formulae

Suppose \textit{deadlock} is an atomic proposition, then:

\[ \forall \forall \diamond \forall \textit{deadlock} \]

can be read as:

\textit{the system will always progress to a state where it is henceforth permanently “deadlocked”}
Suppose \( \text{floor2}, \text{floor5}, \text{direction} \_\text{up}, \text{and button} \_\text{pressed} \_\text{5} \) are atomic propositions, then:

\[
\forall \square (\text{floor2} \land \text{direction} \_\text{up} \land \text{button} \_\text{pressed} \_\text{5}) \Rightarrow \forall (\text{direction} \_\text{up} \text{ UNTIL floor5})
\]

can be read as:

*A lift on the second floor travelling upwards will always continue to travel upwards until reaching level 5 whenever it contains passengers wishing to reach that floor.*
Infinite paths of states

Fix a CTL model $\mathcal{M} = (S, S_0, \rightarrow, \mathcal{L})$

Write $\text{Paths}(s)$ for set of infinite paths of $S$ starting at $s$

Write $\pi[i]$ for $i^{th}$ state of $\pi$ (“indexing”)

Write $\pi^i$ for suffix of $\pi$ starting position $i$

Satisfaction at a state

Suppose $\mathcal{M}$ is a model, $s$ is a state in $\mathcal{M}$, and $\Phi$ is a state formula

Define the satisfaction relation $s \models \Phi$ recursively by:

\[
\begin{align*}
    s \models \top & \quad \text{always} \\
    s \models \bot & \quad \text{never} \\
    s \models p & \quad \text{iff } p \in \mathcal{L}(s) \\
    s \models \neg \Phi & \quad \text{iff not } s \models \Phi
\end{align*}
\]

\[
\begin{align*}
    s \models \Phi \lor \Psi & \quad \text{iff } s \models \Phi \text{ or } s \models \Psi \\
    s \models \Phi \land \Psi & \quad \text{iff } s \models \Phi \text{ and } s \models \Psi \\
    s \models \Phi \Rightarrow \Psi & \quad \text{iff not } s \models \Phi \text{ or if } s \models \Phi \text{ and } s \models \Psi
\end{align*}
\]

\[
\begin{align*}
    s \models \forall \phi & \quad \text{iff } \pi \models \phi \text{ for every } \pi \in \text{Paths}(s) \\
    s \models \exists \phi & \quad \text{iff } \pi \models \phi \text{ for some } \pi \in \text{Paths}(s)
\end{align*}
\]

$\pi \models \phi$ is the evaluation of path formula $\phi$ relative to a path $\pi$
Satisfaction along a path

Suppose $\mathcal{M}$ is a model, $\pi$ is a path in $\mathcal{M}$, and $\phi$ is a path formula. Define the satisfaction relation $\pi \models \phi$ by:

- $\pi \models \Diamond \Phi$ iff $\pi[1] \models \Phi$
- $\pi \models \Box \Phi$ iff $\pi[i] \models \Phi$ for all $i$
- $\pi \models \Diamond \Phi$ iff $\pi[i] \models \Phi$ for some $i$
- $\pi \models \Phi$ UNTIL $\Psi$ iff $\pi[i] \models \Psi$ for some $i$ and $\pi[j] \models \Phi$ for all $j < i$

Notes on satisfaction relations

Two relations are mutually recursive—mutually recursive grammar

Satisfaction relation for path formulae similar to LTL relation

BUT:

- In LTL modality $\Diamond \phi$ uses all suffixes of path $\pi$
- In CTL modality $\Box \Phi$ uses all indexes of path $\pi$
- Similar for other modalities

Tip: imagine types of $\pi^i$, $\pi[i]$ and satisfaction relations

Examples

CTL model as a picture:

```
\begin{tikzpicture}
  \node (s0) at (0,0) {$s_0: \{a, b, c\}$};
  \node (s1) at (1,1) {$s_1: \{b\}$};
  \node (s2) at (2,1) {$s_2: \{c\}$};
  \node (s3) at (2,0) {$s_3: \{c\}$};
  \draw[->] (s0) -- (s1);
  \draw[->] (s0) -- (s2);
  \draw[->] (s1) -- (s2);
  \draw[->] (s2) -- (s3);
\end{tikzpicture}
```

We have $s_0 \models a \land b \land c$
Examples

We have $s_0 \models \forall (b \text{ UNTIL } c)$

We have $s_1 \models \forall c$

We have $s_1 \models \forall \forall c$

We have $s_1 \models \exists \diamond a$
Examples

\[ s_0: \{a, b, c\} \rightarrow s_1: \{b\} \rightarrow s_2: \{c\} \rightarrow s_3: \{c\} \rightarrow s_0: \{a, b, c\} \]

We have \( s_2 \models \exists c \)

Satisfaction in model

Write \( M \models \Phi \) when \( s \models \Phi \) for all states \( s \) in \( M \)
Read \( M \models \Phi \) as “model \( M \) satisfies \( \Phi \)”
Holds whenever all states of \( M \) satisfy \( \Phi \)

Semantic equivalence

Say \( \Phi \) and \( \Psi \) are semantically equivalent \((\Phi \equiv \Psi)\) when:

\[ M \models \Phi \text{ if and only if } M \models \Psi \text{ for all models } M \]

Intuitively \( \Phi \equiv \Psi \) asserts that:

- \( \Phi \) and \( \Psi \) have same “semantic content”
- Safe to replace \( \Phi \) with \( \Psi \) (and vice versa) in any context
- Quantifying over \( M \) means can’t distinguish models
Properties of semantic equivalence

Semantic equivalence:

- Is reflexive ($\phi \equiv \phi$)
- Is symmetric ($\phi \equiv \psi$ implies $\psi \equiv \phi$)
- Is transitive ($\phi \equiv \psi$ and $\psi \equiv \xi$ implies $\phi \equiv \xi$)

Also is congruent with structure of formulae

Example: $\phi_1 \equiv \phi_2$ implies $\neg \phi_1 \equiv \neg \phi_2$ and $\exists \phi_1 \equiv \exists \phi_2$

Important semantic equivalences

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

$$\Phi \Rightarrow \Psi \equiv \neg \Phi \lor \Psi$$

$$\Phi \lor \Psi \equiv \neg (\neg \Phi \land \neg \Psi)$$

$$\forall \diamond \Phi \equiv \neg \exists (\top \text{ UNTIL } \neg \Phi)$$

$$\forall \square \Phi \equiv \neg \exists (\top \text{ UNTIL } \neg \Phi)$$

$$\forall (\Phi \text{ UNTIL } \Psi) \equiv \neg \exists (\neg \Psi \text{ UNTIL } (\neg \Phi \land \neg \Psi)) \land \neg \exists \neg \Psi$$

$$\exists \diamond \Phi \equiv \exists (\top \text{ UNTIL } \Phi)$$

Example proof

Task: show $\Phi \lor \Psi \equiv \neg (\neg \Phi \land \neg \Psi)$

Fix arbitrary model $\mathcal{M}$ and state $s$ in $\mathcal{M}$

Need to show $s \models \Phi \lor \Psi$ if and only if $s \models \neg (\neg \Phi \land \neg \Psi)$
One direction

Assume $s \models \Phi \lor \Psi$
Then $s \models \Phi$ or $s \models \Psi$
Assume without loss of generality $s \models \Phi$
Then not $s \models \neg \Phi$
Hence not $s \models \neg \Phi \land \neg \Psi$
Therefore $s \models \neg (\neg \Phi \land \neg \Psi)$, as required

Existential Normal Form

Define formulae in Existential Normal Form (ENF) by:

$$\Phi, \Psi ::= T \mid p$$
$$\quad ::= \Phi \land \Psi \mid \neg \Phi$$
$$\quad ::= \exists \bigcirc \Phi \mid \exists (\Phi \text{ UNTIL } \Psi) \mid \exists \square \Phi$$

Theorem:

*Every state formula has an equivalent ENF formula*

Proof: by structural induction, using previous semantic equivalences and congruences

Note proof is constructive: describes an algorithm

Summary

- CTL uses a branching model of time
- CTL state formulae express “state properties” of systems
- CTL semantics with respect to states in model
- Equivalence when formulae have same “semantic content”
- Can use equivalences to rewrite a formula into ENF