Hoare Logic and Model Checking

Model Checking
Lecture 10: Computation Tree Logic (CTL)

Dominic Mulligan
Based on previous slides by Alan Mycroft and Mike Gordon

Programming, Logic, and Semantics Group
University of Cambridge

Academic year 2016–2017
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
Like in LTL, we fix a set $AP$ of **atomic propositions**

We continue to use $p$, $q$, $r$, and so on to range over $AP$
Define **state formulae** with the following grammar:

\[ \Phi, \Psi, \Xi ::= \top | \bot | p \]
\[ ::= \neg \Phi \]
\[ ::= \Phi \land \Psi | \Phi \lor \Psi | \Phi \Rightarrow \Psi \]
\[ ::= \forall \phi | \exists \phi \]

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

\[ \phi, \psi, \xi ::= \circ \Phi | \Box \Phi | \Diamond \Phi | \Phi \text{ UNTIL } \Psi \]

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):

\[ \top | \bot | p \]

\( \top, \bot, \) 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!
Second line (of state formula grammar):

\[ \neg \Phi \]

If \( \Phi \) is a CTL state formula, then \( \neg \Phi \) is a CTL state formula

- \( \neg \Phi \) is **negation** of \( \phi \) (or “not \( \Phi \)”)

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 $\forall \phi$ and $\exists \phi$ are CTL state formulae

- $\forall \phi$ 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:

\[ \circ \Phi \mid \Box \Phi \mid \Diamond \Phi \mid \Phi \text{ UNTIL } \Psi \]

If \( \Phi \) and \( \Psi \) are CTL state formulae, then \( \circ \Phi \), \( \Box \Phi \), \( \Diamond \Phi \), and \( \Phi \text{ UNTIL } \Psi \) are CTL path formulae

- \( \Box \Phi \) is “henceforth \( \Phi \)”, or “from now, always \( \Phi \)”
- \( \Diamond \Phi \) is “at some future point \( \Phi \)”
- \( \circ \Phi \) is “immediately after \( \Phi \)”, or “in the next state \( \Phi \)”
- \( \Phi \text{ UNTIL } \Psi \) is “at some future point \( \Psi \), but until then \( \Phi \)”
Grammar above enforces path formula be “covered” by quantifier

Impossible to construct $\Box \forall \phi$ or $\exists \phi \text{ UNTIL } \Psi$

Effect is to have

$$\forall \Box \phi \quad \exists \Box \phi \quad \forall \Diamond \phi \quad \exists \Diamond \phi$$

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

$\forall \Box$, $\exists \bigcirc$, and so on, are “derived modalities”
Alternative syntax for modalities

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

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

• ∀ and ∃ 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 ∀ and ∃
• X, G, F, and U instead of ◯, □, ♦, and UNTIL
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 $\lor$ and $\land$
- Finally $\Rightarrow$ binds least tightly
So:
\[ \Phi \Rightarrow \forall \Box \Psi \] means \[ \Phi \Rightarrow (\forall (\Box \Psi)) \]
\[ \Phi \Rightarrow \Psi \lor \exists \square \Psi \] means \[ \Phi \Rightarrow (\Phi \lor (\exists (\square \Psi))) \]
\[ \forall \Box \Phi \lor \Xi \Rightarrow \Psi \text{ UNTIL } \Xi \] means \[ ((\forall (\Box \Phi)) \lor \Xi) \Rightarrow (\Psi \text{ UNTIL } \Xi) \]
and so on...
Example CTL formulae

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

$$\exists \Diamond (\texttt{started} \land \neg \texttt{ready})$$

can be read as:

\textit{it is possible to get to a state where \textquote{started} holds but \textquote{ready} does not}
Example CTL formulae

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

$$\forall \Box \neg (\text{started} \land \neg \text{ready})$$

can be read as:

\textit{it is not possible to get to a state where “started” holds but “ready” does not}
Suppose \texttt{deadlock} is an atomic proposition, then:

\[ \forall \lozenge \forall \Box \text{deadlock} \]

can be read as:

\textit{the system will always progress to a state where it is henceforth permanently “deadlocked”}
Suppose $\text{floor2}$, $\text{floor5}$, $\text{direction\_up}$, and $\text{button\_pressed\_5}$ are atomic propositions, then:

$$\forall \square (\text{floor2} \land \text{direction\_up} \land \text{button\_pressed\_5} \Rightarrow \forall (\text{direction\_up \ UNTIL \ floor5}))$$

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.
Semantics of CTL
Making intuition precise

Previous examples:
  • Showed examples of properties expressible in CTL,
  • Provided intuition for meaning of CTL formulae

Time to make that intuition precise...
Recall $\mathcal{M} = \langle S, S_0, \rightarrow, \mathcal{L} \rangle$, where:

- $S$ set of **states**
- $S_0 \subseteq S$ set of **initial states**
- $\rightarrow \subseteq S \times S$ (right-serial) **transition relation** on $S$
- $\mathcal{L} : S \rightarrow \mathcal{P}(AP)$ **labelling function**

“Right serial” means $\forall s \in S. \exists s' \in S. s \rightarrow s'$
Fix a CTL model $\mathcal{M} = \langle S, S_0, \rightarrow, \mathcal{L} \rangle$

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:

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

\[ s \models \Phi \lor \Psi \quad \text{iff} \quad s \models \Phi \text{ or } s \models \Psi \]
\[ s \models \Phi \land \Psi \quad \text{iff} \quad s \models \Phi \text{ and } s \models \Psi \]
\[ s \models \Phi \Rightarrow \Psi \quad \text{iff not } s \models \Phi \text{ or if } s \models \Phi \text{ and } s \models \Psi \]
Satisfaction at a state

\[ s \models \forall \phi \quad \text{iff} \quad \pi \models \phi \text{ for every } \pi \in \text{Paths}(s) \]

\[ s \models \exists \phi \quad \text{iff} \quad \pi \models \phi \text{ for some } \pi \in \text{Paths}(s) \]

\( \pi \models \phi \) 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:

\[
\begin{align*}
\pi &\models \Diamond \Phi \quad \text{iff} \quad \pi[1] \models \Phi \\
\pi &\models \Box \Phi \quad \text{iff} \quad \pi[i] \models \Phi \quad \text{for all} \quad i \\
\pi &\models \Diamond \Phi \quad \text{iff} \quad \pi[i] \models \Phi \quad \text{for some} \quad i \\
\pi &\models \Phi \text{ UNTIL } \Psi \quad \text{iff} \quad \pi[i] \models \Psi \quad \text{for some} \quad i \quad \text{and} \quad \pi[j] \models \Phi \quad \text{for all} \quad j < i
\end{align*}
\]
Two relations are mutually recursive—mutually recursive grammar

Satisfaction relation for path formulae similar to LTL relation

BUT:

- In LTL modality $\Box \phi$ 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:

\( s_0 : \{a, b, c\} \)

\( s_1 : \{b\} \)

\( s_2 : \{c\} \)

\( s_3 : \{c\} \)
Examples

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

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

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

We have $s_1 \models \exists \Diamond a$
We have $s_2 \models \exists \Box c$
Semantic equivalence
Satisfaction in model

Write $\mathcal{M} \models \Phi$ when $s \models \Phi$ for all states $s$ in $\mathcal{M}$

Read $\mathcal{M} \models \Phi$ as “model $\mathcal{M}$ satisfies $\Phi$”

Holds whenever all states of $\mathcal{M}$ 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
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 $\exists \bigcirc \phi_1 \equiv \exists \bigcirc \phi_2$
Important semantic equivalences

$\top \equiv \neg \bot$

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

$\Phi \lor \Psi \equiv \neg (\neg \Phi \land \neg \Psi)$
Important semantic equivalences

\[ \forall \bigcirc \Phi \equiv \neg \exists \bigcirc \neg \Phi \]

\[ \forall \Box \Phi \equiv \neg \exists (\top \text{ UNTIL } \neg \Phi) \]

\[ \forall \lozenge \Phi \equiv \forall (\top \text{ UNTIL } \Phi) \]

\[ \forall (\Phi \text{ UNTIL } \Psi) \equiv \neg \exists (\neg \Psi \text{ UNTIL } (\neg \Phi \land \neg \Psi)) \land \neg \exists \Box \neg \Psi \]

\[ \exists \lozenge \Phi \equiv \exists (\top \text{ UNTIL } \Phi) \]
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 $s \models \neg(\neg\Phi \land \neg\Psi)$
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
Assume $s \models \neg(\neg\Phi \land \neg\Psi)$

Then not $s \models \neg\Phi$ and $s \models \neg\Psi$

Then not (not $s \models \Phi$ and not $s \models \Psi$)

Hence either $s \models \Phi$ or $s \models \Psi$

Without loss of generality, assume $s \models \Phi$

Then $s \models \Phi \lor \Psi$, as required

Therefore $\Phi \lor \Psi \equiv \neg(\neg\Phi \land \neg\Psi)$
Define formulae in Existential Normal Form (ENF) by:

\[ \Phi, \Psi ::= \top \mid p \]
\[ ::= \Phi \land \Psi \mid \neg \Phi \]
\[ ::= \exists \circ \Phi \mid \exists (\Phi \text{ UNTIL } \Psi) \mid \exists \Box \Phi \]

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