Lecture 16: Computation tree logic
CTL Model Checking

Huan Zhang
huan@huan-zhang.com
Class project

Midterm project presentation: 3/26 and 3/28. (Next week!)

- **5-min** presentations for each team. (5% of final grade)
- Slides due on 3/25. We will compile all slides into a single file for fast switching.
- Presentation includes problem setting, proposed methodology, and initial results.
- I will give you some feedback after your presentation (1-min)

In addition: each person should give feedback for 3 projects that interest you most on each day. (total 6 feedbacks; count towards the 5% class participation grades. Feedback will be submitted to Canvas and also shared to peers. Feedback template will be given.)
Outline

• Temporal logics
  • Computational Tree Logic (CTL)

• CTL model checking for automata
  • Setup
  • CTL syntax and semantics
  • Model checking algorithms
  • Example

• References: Model Checking, Second Edition, by Edmund M. Clarke, Jr., Orna Grumberg, Daniel Kroening, Doron Peled and Helmut Veith

• Principles of Model Checking, by Christel Baier and Joost-Pieter Katoen
Introduction to temporal logics

Temporal logics: Formal language for representing, and reasoning about, propositions qualified in terms of a sequence

Amir Pnueli received the ACM Turing Award (1996) for seminal work introducing temporal logic into computer science and for outstanding contributions to program verification.

Large follow-up literature, e.g., different temporal logics MTL, MITL, PCTL, ACTL, STL, applications in synthesis and monitoring

Slides adapted from Prof. Sayan Mitra's slides in Fall 2021
Setup: States are labeled

We have a set of **atomic propositions (AP)**

These are the properties that hold in each state, e.g., “light is green”, “has 2 tokens”, “oven is hot”

We have a **labeling function** that assigns to each state, a set of propositions that hold at that state

\[ L: Q \rightarrow 2^{AP} \]
Notations

Automata with state labels but no action labels ("Kripke structure")

\[ \mathcal{A} = \langle Q, Q_0, T, L \rangle \]

\[ AP = \{ a, b, c \} \]

\[ L(q_0) = \{ a, b \} \]
Computational tree logic (CTL)

**Unfolding** the automaton

We get a tree, representing all possible computations

A **CTL formula** allows us to specify subsets of paths in this tree
CTL quantifiers

Path quantifiers
- E: Exists some path
- A: All paths

Temporal operators
- X: Next state
- U: Until ("p U q" means "p holds until q holds")
- F: Eventually (some time in future)
- G: Globally (always)

\( \phi \): “no collision”
Invariance: \( AG\phi \)

\( \phi \): “one token”
Stabilization: \( AF\phi \)
Visualizing CTL semantics

Path quantifiers
E: Exists some path
A: All paths

Temporal operators
X: Next state
U: Until
F: Eventually
G: Globally (Always)

$q \models EF \text{red}$

$q \models EG \text{red}$

$q \models AF \text{red}$

$q \models AG \text{red}$

Slides adapted from Prof. Sayan Mitra’s slides in Fall 2021
Visualizing CTL semantics

Path quantifiers
E: Exists some path
A: All paths

Temporal operators
X: Next state
U: Until
F: Eventually
G: Globally (Always)

Example diagrams:
- \( q \models A [\text{red } U \text{ green}] \)
- \( q \models E [\text{red } U \text{ green}] \)
- \( q \models AX \text{ red} \)
- \( q \models EX \text{ red} \)

Slides adapted from Prof. Sayan Mitra’s slides in Fall 2021
CTL syntax

CTL syntax

State Formula (SF) ::= true | p | ¬f₁ | f₁ ∧ f₂ | E \( \phi \) | A \( \phi \)

Path Formula (PF) ::= X f₁ | f₁ U f₂ | G f₁ | F f₁

Examples:

everything in the previous two slides;

\( AG (p₁ \Rightarrow AF p₂) \)

Non-examples

\( AXX a \); path and state operators must alternate in CTL

Slides adapted from Prof. Sayan Mitra’s slides in Fall 2021
CTL syntax

**State Formula (SF)** ::= \( true \mid p \mid \neg f_1 \mid f_1 \land f_2 \mid E \phi \mid A \phi \)

**Path Formula (PF)** ::= \( X f_1 \mid f_1 U f_2 \mid G f_1 \mid F f_1 \)

where \( p \in AP, \ f_1, f_2 \in SF, \ \phi \in PF \)

**Depth** of formula: number of production rules used

EX \( a; \) (depth 3)

AX EX \( a; \) (depth 5)

AG AF green; (depth 5)

AF AG single token (depth 5)

Slides adapted from Prof. Sayan Mitra’s slides in Fall 2021
microwave oven state machine

1
!Start
!Close
!Heat
!Error

2
Start
!Close
!Heat
Error

3
!Start
Close
Heat
!Error

4
!Start
Close
Heat
!Error
done

5
Start
Close
!Heat
Error

6
Start
Close
!Heat
!Error

7
Start
Close
Heat
!Error

Start oven
Open door
Close door
Open door
Start oven
Warmup
Start cooking

Reset
Let’s check this: $\textbf{AG} (\text{Start} \Rightarrow \textbf{AF} \text{Heat})$

**English:** “In all states, it is true that if start holds in a state, the in some state on all future paths from that state, heat will eventually hold also”
Let’s check this: \( \textbf{AG} \) (Start => AF Heat)

\( q_1 = \textbf{AG} \) (Start => AF Heat) ?
CTL semantics

Automaton $\mathcal{A} = \langle Q, Q_0, T, L \rangle$, $q \in Q$

CTL formula $\phi$

For a state $q$, $q \models \phi$ denotes that $q$ satisfies $\phi$

For a execution $\alpha$, $\alpha \models \phi$ denotes that path (execution) $\alpha$ satisfies $\phi$
CTL semantics (cont.)

Here $\models$ is defined inductively as:

$$
q \models p \iff p \in L(q), \text{ for } p \in AP
$$

$$
q \models \neg f_1 \iff q \not\models f_1
$$

$$
q \models f_1 \land f_2 \iff q \models f_1 \land q \models f_2
$$

$$
q \models E\phi \iff \exists \alpha, \alpha.f\text{ state} = q, \alpha \models \phi
$$

$$
q \models A\phi \iff \forall \alpha, \alpha.f\text{ state} = q, \alpha \models \phi
$$

$$
\alpha \models Xf \iff \alpha[1] \models f
$$

$$
\alpha \models f_1 U f_2 \iff \exists i \geq 0, \alpha[i] \models f_2 \text{ and } \forall j < i \alpha[j] \models f_1
$$

$$
\alpha \models F f_1 \iff \exists i \geq 0, \alpha[i] \models f_1
$$

$$
\alpha \models G f_1 \iff \forall i \geq 0, \alpha[i] \models f_1
$$

Example: $q_1 \models AG (\text{Start} => \text{AF Heat})$

Automaton satisfies property: $\mathcal{A} \models f$ iff $\forall q \in Q_0, q \models f$
Universal CTL operators

$X, U, G$ can be used to derive other operators

$true \ U \ f \ \equiv \ F \ f$

$Gf \ \equiv \ \neg F(\neg f)$

All combinations can be expressed using $EX, EU, EG$

<table>
<thead>
<tr>
<th>$AXf$</th>
<th>$AGf$</th>
<th>$AFf$</th>
<th>$A[f_1Uf_2]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\neg EX(\neg f)$</td>
<td>$\neg EF(\neg f)$</td>
<td>$\neg EG(\neg f)$</td>
<td>$\neg(E[(\neg f_1)U(\neg f_1 \lor f_2)] \lor EG(\neg f_2))$</td>
</tr>
<tr>
<td>$EX$</td>
<td>$EG$</td>
<td>$EF$</td>
<td>$EU$</td>
</tr>
<tr>
<td>$EX$</td>
<td>$EG$</td>
<td>$E(true \ U \ f)$</td>
<td>$EU$</td>
</tr>
</tbody>
</table>

Slides adapted from Prof. Sayan Mitra’s slides in Fall 2021
Algorithm for deciding $\mathcal{A} \models f$

Algorithm works by structural induction on the depth of the formula

Explicit state model checking

Compute the subset $Q' \subseteq Q$ such that $\forall q \in Q'$ we have $q \models f$

If $Q_0 \subseteq Q'$ then we can conclude $\mathcal{A} \models f$
Algorithm for deciding $\mathcal{A} \models f$

Since all CTL operators can be expressed by EX, EU, EG, we just need to figure out how to check these operators.

$\mathcal{A}$

CTL: e.g., $AG (a \rightarrow AF b)$
Induction on depth of formula

Algorithm computes a function \( \text{label}: Q \rightarrow CTL(AP) \) that labels each state with a CTL formula

- Initially, \( \text{label}(q) = L(q) \) for each \( q \in Q \)

- At \( i^{th} \) iteration \( \text{label}(q) \) contains all sub-formulas of \( f \) of depth \((i - 1)\) that \( q \) satisfies

At termination \( f \in \text{label}(q) \iff q \models f \)
CheckEG($f_1, Q, T, L$)

From $\mathcal{A}$ we construct a new automaton $\mathcal{A}' = \langle Q', T', L' \rangle$ such that

$$Q' = \{ q \in Q \mid f_1 \in \text{label}(q) \}$$

$$T' = \{ (q_1, q_2) \in T \mid q_1 \in Q' \}$$

$$L' : Q' \rightarrow 2^{AP} \quad \forall q' \in Q', \ L'(q') : = L(q')$$

Claim. $q \models EGf_1$ iff

(1) $q \in Q'$

(2) $\exists \alpha \in \text{Execs}_\mathcal{A}$, with $\alpha.f\text{state} = q$ and $\alpha.l\text{state}$ is in a nontrivial Strongly Connected Components $C$ of the graph $\langle Q', T' \rangle$
Set of states in $\mathcal{A}'$ that can reach the nontrivial SCC of $\neg$Heat
**Claim.** \( \mathcal{A}, q \models EGf_1 \) iff 

1. \( q \in Q' \) and 
2. \( \exists \alpha \in \text{Execs}_{\mathcal{A}} \) with \( \alpha.f\text{state} = q \) and \( \alpha.l\text{state} \) is in a nontrivial SCC \( C \) of the graph \( \langle Q', T' \rangle \)

**Proof.** Suppose \( \mathcal{A}, q \models EGf_1 \)

Consider any execution \( \alpha \) with \( \alpha.f\text{state} = q \). Obviously, \( q \models f_1 \) and so, \( q \in Q' \).

Since \( Q \) is finite \( \alpha \) can be written as \( \alpha = \alpha_0 \alpha_1 \) where \( \alpha_0 \) is finite and every state in \( \alpha_1 \) repeats infinitely many times.

Let \( C \) be the states in \( \alpha_1 \). \( C \in Q' \).

Consider any two \( q_1 \) and \( q_2 \) states in \( C \), we observe that \( q_1 \Leftrightarrow q_2 \), and therefore \( C \) is a SCC.

Consider (1) and (2). We construct a path \( \alpha = \alpha_0 \alpha_1 \) such that \( \alpha_0.f\text{state} = q \) and \( \alpha_0 \in Q' \) and \( \alpha_1 \) visits some states infinitely often.
Let $Q' = \{q \in Q \mid f_1 \in \text{label}(q)\}$

Let $\mathbb{C}$ be the set of nontrivial SCCs of $\langle Q', T' \rangle$

$T = \bigcup_{C \in \mathbb{C}} \{q \mid q \in C\}$

for each $q \in T$

$\text{label}(q) := \text{label}(q) \cup \{EGf_1\}$

while $T \neq \emptyset$

for each $q \in T$

$T := T \setminus \{q\}$

for each $q' \in Q'$ such that $(q', q) \in T'$

if $EGf_1 \notin \text{label}(q')$ then

$\text{label}(q') := \text{label}(q') \cup \{EGf_1\}$

$T := T \cup \{q'\}$

Proposition. For any state $\text{label}(q) \ni EGf_1$ iff $q \models EGf_1$.

Proposition. Finite $Q$ therefore terminates and in $O(|Q| + |T|)$ steps.

Slides adapted from Prof. Sayan Mitra’s slides in Fall 2021
Let $S = \{q \in Q \mid f_2 \in \text{label}(q)\}$
for each $q \in S$
\[\text{label}(q) := \text{label}(q) \cup \{E[f_1Uf_2]\}\]
while $S \neq \emptyset$
for each $q' \in S$
$S := S \setminus \{q'\}$
for each $q \in T^{-1}(q')$
if $f_1 \in \text{label}(q)$ then
\[\text{Check all states whose next state is } q'\]
\[\text{This if statement will be always true for } EF f_2\]
\[\text{if } f_1 \in \text{label}(q) \text{ then}\]
\[\text{label}(q) := \text{label}(q) \cup \{E[f_1Uf_2]\}\]
$S := S \cup \{q\}$

**Proposition.** For any state $\text{label}(q) \ni E[f_1Uf_2]$ iff $q \models E[f_1Uf_2]$.

**Proposition.** Finite $Q$ therefore terminates and in $O(|Q| + |T|)$ steps.

Slides adapted from Prof. Sayan Mitra's slides in Fall 2021
Structural induction on formula

Six cases to consider based on structure of $f$

$f = p$, for some $p \in AP$, $\forall q$, $\text{label}(q) := \text{label}(q) \cup f$

$f = \neg f_1$  
if $f_1 \notin \text{label}(q)$ then $\text{label}(q) := \text{label}(q) \cup f$

$f = f_1 \wedge f_2$  
if $f_1, f_2 \in \text{label}(q)$ then $\text{label}(q) := \text{label}(q) \cup f$

$f = EXf_1$  
if $\exists q' \in Q$ such that $(q, q') \in T$ and $f_1 \in \text{label}(q')$, then $\text{label}(q) := \text{label}(q) \cup f$

$f = E[f_1 U f_2]$  
$\text{CheckEU}(f_1, f_2, Q, T, L)$

$f = EGf_1$  
$\text{CheckEG}(f_1, Q, T, L)$
Putting it all together

Explicit model checking algorithm input $\mathcal{A} \vDash f$?
Structural induction over CTL formula

\[
\begin{align*}
f &= p, & \text{for some } p \in \text{AP}, \forall q, \text{ label}(q) := \text{label}(q) \cup \{p\} \\
f &= \neg f_1 & \text{if } f_1 \notin \text{label}(q) \text{ then } \text{label}(q) := \text{label}(q) \cup f \\
f &= f_1 \land f_2 & \text{if } f_1, f_2 \in \text{label}(q) \text{ then } \text{label}(q) := \text{label}(q) \cup f \\
f &= \text{EX}f_1 & \text{if } \exists q' \in Q \text{ such that } (q, q') \in T \text{ and } f_1 \in \text{label}(q') \text{ then } \\
& \quad \text{label}(q) := \text{label}(q) \cup f \\
f &= \text{E}[f_1 \cup f_2] & \text{CheckEU}(f_1, f_2, Q, T, L) \\
f &= \text{EG}f_1 & \text{CheckEG}(f_1, Q, T, L)
\end{align*}
\]

Proposition. Overall complexity of CTL model checking $O(|f|(|Q| + |T|))$ steps.

Slides adapted from Prof. Sayan Mitra’s slides in Fall 2021
AG (Start => AF Heat)
\[ \text{EF} (\text{Start} \land \text{EG} \neg \text{Heat}) \]

\[ \neg [\text{True EU} (\text{Start} \land \text{EG} \neg \text{Heat})] \]
! EF (Start ∧ EG ! Heat)

! [True EU (Start ∧ EG ! Heat)]
! EF (Start ∧ EG ! Heat)

Start, ! Heat
EG ! Heat

Nontrivial SCC of ! Heat
\[ \text{\textit{EF}} (\text{Start} \land \text{EG} \rightarrow \text{\textit{Heat}}) \]

\[ \text{Start} \land \text{EG} \rightarrow \text{\textit{Heat}} \]

\[ \text{Set of states that can reach the nontrivial SCC of \text{\textit{Heat}}} \]
\[ \neg \text{EF} (\neg \text{Start} \land \neg \text{EG} \land \neg \text{Heat}) \]

\[ \text{Start}, \neg \text{Heat} \]

\[ \neg \text{EG} \land \neg \text{Heat} \]

\[ \text{Start} \land \neg \text{EG} \land \neg \text{Heat} \]

Start oven

Open door

Start, \neg \text{Heat}

\[ \text{EG} \land \neg \text{Heat} \]

\[ \text{Start} \land \text{EG} \land \neg \text{Heat} \]

\[ \text{Start}, \neg \text{Heat} \]

\[ \text{EG} \land \neg \text{Heat} \]

\[ \text{Start} \land \text{EG} \land \neg \text{Heat} \]

\[ \neg \text{EF} (\neg \text{Start} \land \neg \text{EG} \land \neg \text{Heat}) \]

\[ \text{Start}, \neg \text{Heat} \]

\[ \neg \text{EG} \land \neg \text{Heat} \]

\[ \text{Start} \land \neg \text{EG} \land \neg \text{Heat} \]

Open door

Close door

Start oven

Start cooking

Warmup

\[ \text{EF} (\neg \text{Start} \land \neg \text{EG} \land \text{Heat}) \]

\[ \text{Start}, \text{Heat} \]

\[ \neg \text{EG} \land \text{Heat} \]

\[ \text{Start} \land \neg \text{EG} \land \text{Heat} \]

\[ \neg \text{EF} (\neg \text{Start} \land \neg \text{EG} \land \text{Heat}) \]

\[ \text{Start}, \text{Heat} \]

\[ \neg \text{EG} \land \text{Heat} \]

\[ \text{Start} \land \neg \text{EG} \land \text{Heat} \]

Close door

Start oven

\[ \text{Start}, \text{Heat} \]

\[ \neg \text{EG} \land \text{Heat} \]

\[ \text{Start} \land \neg \text{EG} \land \text{Heat} \]

\[ \neg \text{EF} (\neg \text{Start} \land \neg \text{EG} \land \text{Heat}) \]

\[ \text{Start}, \text{Heat} \]

\[ \neg \text{EG} \land \text{Heat} \]

\[ \text{Start} \land \neg \text{EG} \land \text{Heat} \]

Open door

Close door

Start oven

Start cooking
None of the states are labeled with ! EF (Start ∨ EG ∨ Heat)