Symbolic CTL Model Checking

Crystal Chang Din

Precise Modeling and Analysis group (PMA),
University of Oslo

INF9140 - Specification and Verification of Parallel Systems
Outline

1. Computation Tree Logic (CTL)

2. CTL Model Checking Algorithm

3. Symbolic CTL Model Checking
Recall of LTL

A linear logic over an infinite sequence of states.
Transition System and Computation Tree

Assume $TS$ is finite, and has no terminal states, we can unfold it at a chosen state into an infinite computation tree.

CTL introduces two path quantifiers:

$\forall$: all computations that start in a state (implicitly expressed in LTL)

$\exists$: some computations that start in a state (cannot be expressed in LTL)
Some Basic CTL Formulae

\[ \exists \diamond \text{black} \]
\[ \exists \square \text{black} \]
\[ \forall \diamond \text{black} \]
\[ \forall \square \text{black} \]
\[ \exists (\text{gray} \cup \text{black}) \]
\[ \forall (\text{gray} \cup \text{black}) \]

Figure 6.2: Visualization of semantics of some basic CTL formulae.
Syntax of CTL

State formulae:

\[ \Phi ::= \text{true} \mid \alpha \mid \neg \Phi \mid \Phi_1 \land \Phi_2 \mid \forall \varphi \mid \exists \varphi \]

where \( \alpha \in \text{AP} \) (atomic proposition)

Path formulae:

\[ \varphi ::= \Box \Phi \mid \Phi_1 U \Phi_2 \]

Four possible ways to turn path formulae into state formulae:

\[ \forall \Box \forall U \exists \Box \exists U \]
Examples of CTL formulae

- Illegal CTL formulae
  - $\exists (x = 1 \land y > 4)$
  - $\exists \bigcirc (true U (x = 1))$

- Legal CTL formulae
  - $\exists \bigcirc (x = 1 \land y > 4)$
  - $\exists \bigcirc \forall (true U (x = 1))$
Derived Temporal Modalities

eventually
  ▶ \( \exists \Diamond \Phi \equiv \exists (true \mathcal{U} \Phi) \)
  \( \Phi \) holds potentially
  ▶ \( \forall \Diamond \Phi \equiv \forall (true \mathcal{U} \Phi) \)
  \( \Phi \) is inevitable

always
  ▶ \( \exists \Box \Phi \equiv \neg \forall \Diamond \neg \Phi \) \( (\neq \exists \Diamond \neg \Phi) \) potentially always \( \Phi \)
  ▶ \( \forall \Box \Phi \equiv \neg \exists \Diamond \neg \Phi \) \( (\neq \forall \Diamond \neg \Phi) \) invariantly \( \Phi \)
Satisfaction Relation for CTL

Let transition system $TS = (S, Act, →, l, AP, L)$, state $s \in S$, $\alpha$ be an atomic proposition, $\Phi_1, \Phi_2$ be CTL state formulae, and $\varphi$ be a CTL path formulae.

The satisfaction relation $\models$ is defined for state formulae by

- $s \models \alpha$ iff $\alpha \in L(s)$
- $s \models \neg \Phi$ iff not $s \models \Phi$
- $s \models \Phi_1 \land \Phi_2$ iff $(s \models \Phi_1)$ and $(s \models \Phi_2)$
- $s \models \exists \varphi$ iff $\pi \models \varphi$ for some $\pi \in \text{Path}(s)$
- $s \models \forall \varphi$ iff $\pi \models \varphi$ for all $\pi \in \text{Path}(s)$

For path $\pi$, the satisfaction relation $\models$ for path formulae is defined by

- $\pi \models \Diamond \Phi$ iff $\pi[1] \models \Phi$
- $\pi \models \Phi_1 U \Phi_2$ iff $\exists j \geq 0. (\pi[j] \models \Phi_2 \land (\forall 0 \leq k < j. \pi[k] \models \Phi_1))$
Expressiveness of CTL vs. LTL

Intuitively, the LTL formula $\Diamond \Box \alpha$ is equivalent to the CTL formula $\forall \Diamond \forall \Box \alpha$. However, $s_0 \models \Diamond \Box \alpha$ and $s_0 \not\models \forall \Diamond \forall \Box \alpha$ are shown in the following case.

1. $\Diamond \Box \alpha$: 
   $\alpha$ will eventually forever hold from some state

2. $\forall \Diamond \forall \Box \alpha$: 
   On any computation, eventually some state, $s$ say, is reached such that $s \models \forall \Box \alpha$
Satisfaction Set

Given a transition system $TS$, the satisfaction set $Sat(\Phi)$ for CTL-state formula $\Phi$ is defined by:

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

For example:

$$Sat(\forall \square a) = \{ s_3 \}$$
CTL Semantics for Transition System

\[ TS \models \Phi \iff \forall s_0 \in I . s_0 \models \Phi \]

in other words,

\[ TS \models \Phi \iff I \subseteq Sat(\Phi) \]
Outline

1. Computation Tree Logic (CTL)
2. CTL Model Checking Algorithm
3. Symbolic CTL Model Checking
CTL in Existential Normal Form (ENF)

\[
\Phi ::= \text{true} \mid \alpha \mid \neg \Phi \mid \Phi_1 \land \Phi_2 \mid \exists \Box \Phi \mid \exists (\Phi_1 U \Phi_2) \mid \forall \Box \Phi
\]

Other CTL formulae can be translated into equivalent ENF formulae:

\[
\begin{align*}
\exists \Diamond \Phi & \equiv \exists (\text{true} U \Phi) \\
\forall \Diamond \Phi & \equiv \neg \exists \Diamond \neg \Phi \\
\forall (\Phi_1 U \Phi_2) & \equiv \neg \exists (\neg \Phi_2 U (\neg \Phi_1 \land \neg \Phi_2)) \land \neg \exists \Box \neg \Phi_2 \\
\forall \Box \Phi & \equiv \neg \exists \Box \neg \Phi \\
\forall \Box \Phi & \equiv \neg \exists \Diamond \neg \Phi = \neg \exists (\text{true} U \neg \Phi)
\end{align*}
\]

CTL in ENF has the same expressiveness as CTL

We will use ENF of CTL in the model checking algorithm.
Overview of the Algorithm

To verify for a given transition system $TS$ and CTL formula $\Phi$ whether $TS \models \Phi$, the procedure is:

1. recursive computation of the sets $Sat(\Psi)$ for all subformulae $\Psi$ of $\Phi$

2. checking whether $I \subseteq Sat(\Phi)$
Computation of the Satisfaction Sets

Algorithm 14 Computation of the satisfaction sets

Input: finite transition system $TS$ with state set $S$ and CTL formula $\Phi$ in ENF
Output: $Sat(\Phi) = \{ s \in S \mid s \models \Phi \}$

(* recursive computation of the sets $Sat(\Psi)$ for all subformulae $\Psi$ of $\Phi$ *)

switch($\Phi$):

- true
- $a$
- $\Phi_1 \land \Phi_2$
- $\neg \Psi$
- $\exists \circ \Psi$
- $\exists (\Phi_1 \cup \Phi_2)$

end switch
$\text{Sat}(\exists \square \Phi)$

$T_0 = \text{Sat}(\Phi)$ and $T_{i+1} = T_i \cap \{s \in \text{Sat}(\Phi) \mid \text{Post}(s) \cap T_i \neq \emptyset\}$

until $T_{i+1} = T_i$ (reach the greatest fix point)

i.e. $\text{Sat}(\exists \square b) = \{s_0, s_2, s_4\}$
Sat(∃(Φ₁ ∪ Φ₂))

T₀ = Sat(Φ₂) and T_{i+1} = T_i ∪ \{s ∈ Sat(Φ₁) | Post(s) ∩ T_i ≠ ∅\}

until T_{i+1} = T_i (reach the smallest fix point)

i.e. Sat(∃(true ∪ (a = c) ∧ (a ≠ b))) = \{s_4, s_5, s_6, s_7\}
Time Complexity

Let $TS$ be a finite transition system with $N$ states and $K$ transitions. $\Psi$ a subformula of $\Phi$. The time complexity of calculating $\text{Sat}(\Psi)$ is

$$O(N+K)$$

Also, there are $|\Phi|$ subformulae of $\Phi$. The total time complexity of calculating $\text{Sat}(\Phi)$ is therefore

$$O((N+K) \cdot |\Phi|)$$
Outline

1. Computation Tree Logic (CTL)

2. CTL Model Checking Algorithm

3. Symbolic CTL Model Checking
Why Symbolic?

- **Original version of CTL model checking**
  - single state and single transition at a time
  - record all the predecessors and successors in each state
  - iterative computation: union and intersection of sets
  - state explosion problem in large transition systems

- **Symbolic version**
  - sets of states and sets of transitions at a time
  - binary encoding of states
  - one boolean function for each satisfaction set
  - one boolean function for all the transitions
  - iterative computation: conjunction and disjunction of a sequence of bits
  - very efficient
Encoding States

Binary encoding \((enc)\) of states, as vectors of \(n\) bits:

\[
enc : S \rightarrow \{0, 1\}^n
\]

For example:
8 states \((s_0, s_1, \ldots, s_7)\) can be encoded with 3 bits
\(s_0: 000\)
\(s_1: 001\)
\[ \ldots \]
\(s_7: 111\)
Two Boolean Functions ($X_T$ and $\triangle$)

- $X_T$: to encode set of states $T \subseteq S$ (i.e. $T=\text{Sat}(\alpha)$):
  \[ X_T : \{0,1\}^n \rightarrow \{0,1\} \quad \text{s.t.} \quad X_T(s)=1 \quad \text{iff} \quad s \in T \]

- $\triangle$: to encode set of transitions $\rightarrow \subseteq S \times S$:
  \[ \triangle : \{0,1\}^{2n} \rightarrow \{0,1\} \quad \text{s.t.} \quad \triangle(s,s')=1 \quad \text{iff} \quad s \rightarrow s' \]
\textit{Sat}(∃□b) \textit{ in Symbolic Version}

binary decision tree (BDT)

8 states \((s_0, s_1, \ldots, s_7)\) can be encoded with 3 bits \((z_1z_2z_3)\)

\[X_T(z_1, z_2, z_3) \rightarrow \{0,1\}, \ T=Sat(b)\]  
i.e. \[X_T(s_2) = X_T(0,1,0) = 1\]

\[\triangle(z_1, z_2, z_3, z'_1, z'_2, z'_3) \rightarrow \{0,1\} \]  
i.e. \[\triangle(s_1, s_2) = \triangle(0,0,1,0,1,0) = 0\]
Symbolic Computation

\[ T_0 = \text{Sat}(\Phi_2) \text{ and } T_{i+1} = T_i \cup \{ s \in \text{Sat}(\Phi_1) | \text{Post}(s) \cap T_i \neq \emptyset \} \]

Algorithm 20 Symbolic computation of Sat(∃(C ∪ B))

\[
f_0(\overline{x}) := \chi_C(\overline{x}); \\
j := 0; \\
\text{repeat} \\
\quad f_{j+1}(\overline{x}) := f_j(\overline{x}) \lor (\chi_C(\overline{x}) \land \exists \overline{x}'.(\Delta(\overline{x}, \overline{x}') \land f_j(\overline{x}'))); \\
\quad j := j + 1 \\
\text{until } f_j(\overline{x}) = f_{j-1}(\overline{x}); \\
\text{return } f_j(\overline{x}).
\]

Algorithm 21 Symbolic computation of Sat(∃□B)

\[
f_0(\overline{x}) := \chi_B(\overline{x}); \\
j := 0; \\
\text{repeat} \\
\quad f_{j+1}(\overline{x}) := f_j(\overline{x}) \land \exists \overline{x}'.(\Delta(\overline{x}, \overline{x}') \land f_j(\overline{x}')); \\
\quad j := j + 1 \\
\text{until } f_j(\overline{x}) = f_{j-1}(\overline{x}); \\
\text{return } f_j(\overline{x}).
\]

continue the example from last slide: \( f_0 : 00010111 \rightarrow f_1 : 00010101 \rightarrow f_2 : 00010101 \)
From BDT to BDD

binary decision diagram (BDD): a reduced version of BDT