FORMAL METHODS

Lecture V: CTL Model Checking

Alessandro Artale

Faculty of Computer Science – Free University of Bolzano
Room 2.03

artale@inf.unibz.it http://www.inf.unibz.it/~artale/

Some material (text, figures) displayed in these slides is courtesy of:
Summary of Lecture V

- CTL Model Checking: General Ideas.
- CTL Model Checking: The Labeling Algorithm.
- Labeling Algorithm in Details.
- CTL Model Checking: Theoretical Issues.
CTL Model Checking

CTL Model Checking is a formal verification technique s.t.

• The system is represented as a Kripke Model $\mathcal{KM}$:

• The property is expressed as a CTL formula $\varphi$, e.g.:

$$\square P (p \Rightarrow P \diamond q)$$

• The algorithm checks whether all the initial states, $s_0$, of the Kripke model satisfy the formula $(\mathcal{KM}, s_0 \models \varphi)$. 
The algorithm proceeds along two macro-steps:

1. Construct the set of states where the formula holds:

   \[ [[\varphi]] := \{ s \in S : K_M , s \models \varphi \} \]

   (\([[[\varphi]]\) is called the denotation of \(\varphi\));

2. Then compare the denotation with the set of initial states:

   \[ I \subseteq [[[\varphi]]] ? \]
To compute $[[\varphi]]$ proceed “bottom-up” on the structure of the formula, computing $[[\varphi_i]]$ for each subformula $\varphi_i$ of $\varphi$.

For example, to compute $[[\Box P (p \Rightarrow \Box P \Diamond q)]]$ we need to compute:

- $[[q]]$,
- $[[\Box \Diamond q]]$,
- $[[p]]$,
- $[[p \Rightarrow \Box \Diamond q]]$,
- $[[\Box (p \Rightarrow \Box \Diamond q)]]$
To compute each $[[\varphi_i]]$ for generic subformulas:

- Handle boolean operators by standard set operations;
- Handle temporal operators $\Box \bigcirc$, $\Diamond \bigcirc$ by computing pre-images;
- Handle temporal operators $\Box \Box$, $\Diamond \Box$, $\Box \Diamond$, $\Diamond \Diamond$, $\Box \bigcup$, $\Diamond \bigcup$, by applying fixpoint operators.
Summary

- CTL Model Checking: General Ideas.
- CTL Model Checking: The Labeling Algorithm.
- Labeling Algorithm in Details.
- CTL Model Checking: Theoretical Issues.
The Labeling Algorithm: General Idea

The **Labeling Algorithm**:

- **Input**: Kripke Model and a CTL formula;
- **Output**: set of states satisfying the formula.

**Main Idea**: Label the states of the Kripke Model with the subformulas of $\varphi$ satisfied there.
The Labeling Algorithm: An Example

\[ \square \Diamond q \equiv (q \lor \square \bigcirc (\square \Diamond q)) \]

\[ [\square \Diamond q] \text{ can be computed as the union of:} \]
- \[ [q] = \{2\} \]
- \[ [q \lor \square \bigcirc q] = \{2\} \cup \{1\} = \{1, 2\} \]
- \[ [q \lor \square \bigcirc (q \lor \square \bigcirc q)] = \{2\} \cup \{1\} = \{1, 2\} \text{ (fixpoint).} \]
The Labeling Algorithm: An Example (Cont.)

"p"

1

p

q

2

4

"AF q"

"p -> AF q"

1

p

q

3

4

2

1

p

q

3

p

p

"p \rightarrow \text{AF } q"

"AG(p \rightarrow \text{AF } q)"

\[ p \rightarrow \text{AF } q \]

\[ \square P \phi \equiv (\phi \land \square P \Box P \phi) \]

\[ [\square P \phi] \text{ can be computed as the intersection of:} \]

- \[ [[\phi]] = \{1, 2, 4\} \]
- \[ [[\phi \land \square P \Box P \phi]] = \{1, 2, 4\} \cap \{1, 3\} = \{1\} \]
- \[ [[\phi \land \square P \Box P \phi]] = \{1, 2, 4\} \cap \{\} = \{\} \text{ (fixpoint)} \]
The Labeling Algorithm: An Example (Cont.)

- The set of states where the formula holds is empty, thus:
  - The initial state does not satisfy the property;
  - $\mathcal{M} \notmodels P \Box (p \Rightarrow P \Diamond q)$.

- **Counterexample:** A lazo-shaped path: $1, 2, \{3, 4\}^\omega$ (satisfying $\Diamond \Diamond (p \land \Diamond P \neg q)$)
Summary

- CTL Model Checking: General Ideas.
- CTL Model Checking: The Labeling Algorithm.
- Labeling Algorithm in Details.
- CTL Model Checking: Theoretical Issues.
The Labeling Algorithm: General Schema

Assume \( \phi \) written in terms of \( \neg, \land, \bigcirc, \bigtriangledown, \square \) – minimal set of CTL operators.

The Labeling algorithm takes a CTL formula and a Kripke Model as input and returns the set of states satisfying the formula (i.e., the denotation of \( \phi \)):

1. For every \( \phi_i \in \text{Sub}(\phi) \), find \( [\phi_i] \);
2. Compute \( [\phi] \) starting from \( [\phi_i] \);
3. Check if \( I \subseteq [\phi] \).

Subformulas \( \text{Sub}(\phi) \) of \( \phi \) are checked bottom-up.

To compute each \( [\phi_i] \): if the main operator of \( \phi_i \) is a

- **Boolean Operator**: apply standard set operations;
- **Temporal Operator**: apply recursive rules until a fixpoint is reached.
Let $\mathcal{KM} = \langle S, I, R, L, \Sigma \rangle$ be a Kripke Model.

\[
\begin{align*}
[[false]] &= \{\} \\
[[true]] &= S \\
[[p]] &= \{s \mid p \in L(s)\} \\
[[\neg \varphi_1]] &= S \setminus [[\varphi_1]] \\
[[\varphi_1 \land \varphi_2]] &= [[\varphi_1]] \cap [[\varphi_2]]
\end{align*}
\]
Denotation of Formulas: The $\Diamond \bigcirc \varphi$ Case

- $[[\Diamond \bigcirc \varphi]] = \{s \in S \mid \exists s'. \langle s, s' \rangle \in R \text{ and } s' \in [[\varphi]]\}$
- $[[\Diamond \bigcirc \varphi]]$ is said to be the Pre-image of $[[\varphi]]$ ($\text{PRE}([[\varphi]])$).
- Key step of every CTL M.C. operation.
From the semantics of the $\square$ temporal operator:

$$\square \phi \equiv \phi \land \bigcirc (\square \phi)$$

Then, the following equivalence holds:

$$\Diamond \square \phi \equiv \phi \land \Diamond \bigcirc (\Diamond \square \phi)$$

To compute $[\Diamond \square \phi]$ we can apply the following recursive definition:

$$[[\Diamond \square \phi]] = [[\phi]] \cap PRE([[\Diamond \square \phi]])$$
We can compute \( X := \lbrack \diamondsuit \Box \varphi \rbrack \) inductively as follows:

\[
\begin{align*}
X_1 &:= \lbrack \varphi \rbrack \\
X_2 &:= X_1 \cap \text{PRE}(X_1) \\
\vdots \\
X_{j+1} &:= X_j \cap \text{PRE}(X_j)
\end{align*}
\]

When \( X_n = X_{n+1} \) we reach a fixpoint and we stop.

**Termination.** Since \( X_{j+1} \subseteq X_j \) for every \( j \geq 0 \), thus a fixed point always exists (Knaster-Tarski’s theorem).
Denotation of Formulas: The ♦ □ Case

From the semantics of the □ temporal operator:

\[ \varphi \sqcup \psi \equiv \psi \lor (\varphi \land \bigcirc (\varphi \sqcup \psi)) \]

Then, the following equivalence holds:

\[ \Diamond^P (\varphi \sqcup \psi) \equiv \psi \lor (\varphi \land \Diamond^P \Diamond^P (\varphi \sqcup \psi)) \]

To compute \[ [\Diamond^P (\varphi \sqcup \psi)] \] we can apply the following recursive definition:

\[ [\Diamond^P (\varphi \sqcup \psi)] = [\psi] \cup ([\varphi] \cap \text{PRE}([\Diamond^P (\varphi \sqcup \psi)])) \]
We can compute $X := [\Diamond (\varphi \cup \psi)]$ inductively as follows:

- $X_1 := [\psi]$
- $X_2 := X_1 \cup ([\varphi] \cap \text{PRE}(X_1))$
- $\ldots$
- $X_{j+1} := X_j \cup ([\varphi] \cap \text{PRE}(X_j))$

When $X_n = X_{n+1}$ we reach a fixpoint and we stop.

**Termination.** Since $X_{j+1} \supseteq X_j$ for every $j \geq 0$, thus a fixed point always exists (Knaster-Tarski’s theorem).
We assume the Kripke Model to be a global variable:

\textbf{Function Label}(\varphi) \{ \\
\textbf{case} \ varphi \ \textbf{of} \\
\quad \text{true:} \quad \text{return} \ S; \\
\quad \text{false:} \quad \text{return} \ \{\}; \\
\quad \text{an} \ \text{atom} \ p: \quad \text{return} \ \{s \in S \mid p \in L(s)\}; \\
\quad \neg \varphi_1: \quad \text{return} \ S\setminus\text{Label}(\varphi_1); \\
\quad \varphi_1 \land \varphi_2: \quad \text{return} \ \text{Label}(\varphi_1) \cap \text{Label}(\varphi_2); \\
\quad \Diamond \varphi_1: \quad \text{return} \ \text{PRE}(\text{Label}(\varphi_1)); \\
\quad \Diamond (\varphi_1 \cup \varphi_2): \quad \text{return} \ \text{Label\_EU}(\text{Label}(\varphi_1),\text{Label}(\varphi_2)); \\
\quad \Diamond \Box \varphi_1: \quad \text{return} \ \text{Label\_EG}(\text{Label}(\varphi_1)); \\
\textbf{end case} \\
\}
\[ [P \bigcirc \varphi] = \text{PRE}([\varphi]) = \{ s \in S \mid \exists s'. \langle s, s' \rangle \in R \text{ and } s' \in [\varphi] \} \]

**Function** \( \text{PRE}([\varphi]) \) \{

\begin{align*}
    & \text{var } X; \\
    & X := \{\}; \\
    & \text{for each } s' \in [\varphi] \text{ do} \\
    & \quad \text{for each } s \in S \text{ do} \\
    & \quad \quad \text{if } \langle s, s' \rangle \in R \text{ then} \\
    & \quad \quad \quad X := X \cup \{s\}; \\
    & \quad \text{return } X
\end{align*}
\}
\[ \left[ \begin{array}{l} \Box \varphi \end{array} \right] = \left[ \varphi \right] \cap \text{PRE} \left( \left[ \begin{array}{l} \Box \varphi \end{array} \right] \right) \]

**FUNCTION** \texttt{LABEL\_EG}([\varphi])\{ 
    \texttt{var X, OLD-X;}
    X := [\varphi];
    OLD-X := \emptyset;
    \texttt{while X} \neq OLD-X 
    \texttt{begin}
        OLD-X := X;
        X := X \cap \text{PRE}(X)
    \texttt{end}
    \texttt{return X}
\}
\[[\diamond (\varphi \cup \psi)] = [\psi] \cup ([\varphi] \cap \text{PRE}([\diamond (\varphi \cup \psi)]))\]

**FUNCTION** \text{LABEL\_EU}([\varphi],[\psi]){

\textbf{var} \ X, \text{OLD\_X};
\ X := [\psi];
\ OLD\_X := S;
\ \textbf{while} \ X \neq \text{OLD\_X}
\ \textbf{begin}
\quad \text{OLD\_X} := X;
\quad X := X \cup ([\varphi] \cap \text{PRE}(X))
\ \textbf{end}
\ \textbf{return} \ X

}
Summary

- CTL Model Checking: General Ideas.
- CTL Model Checking: The Labeling Algorithm.
- Labeling Algorithm in Details.
- CTL Model Checking: Theoretical Issues.
The Labeling algorithm works recursively on the structure $\varphi$.

For most of the logical constructors the algorithm does the correct things according to the semantics of CTL.

- To prove that the algorithm is Correct and Terminating we need to prove the correctness and termination of both $\lozenge P \Box$ and $\lozenge P \setminus$ operators.
Definition. Let $S$ be a set and $F$ a function, $F : 2^S \rightarrow 2^S$, then:

1. $F$ is monotone iff $X \subseteq Y$ then $F(X) \subseteq F(Y)$;

2. A subset $X$ of $S$ is called a fixpoint of $F$ iff $F(X) = X$;

3. $X$ is a least fixpoint (LFP) of $F$, written $\mu X . F(X)$, iff, for every other fixpoint $Y$ of $F$, $X \subseteq Y$

4. $X$ is a greatest fixpoint (GFP) of $F$, written $\nu X . F(X)$, iff, for every other fixpoint $Y$ of $F$, $Y \subseteq X$

Example. Let $S = \{s_0, s_1\}$ and $F(X) = X \cup \{s_0\}$. 
Knaster-Tarski Theorem

Notation: $F^i(X)$ means applying $F$ $i$-times, i.e., $F(F(\ldots F(X)\ldots))$.

Theorem[Knaster-Tarski]. Let $S$ be a finite set with $n + 1$ elements. If $F : 2^S \rightarrow 2^S$ is a monotone function then:

1. $\mu X. F(X) \equiv F^{n+1}(\emptyset)$;
2. $\nu X. F(X) \equiv F^{n+1}(S)$.

Proof. (See the textbook “Logic in CS” pg.241)
The function $\text{LABEL}_EG$ computes:

$$[[\lozenge \Box \varphi]] = [[\varphi]] \cap \text{PRE}([[\lozenge \Box \varphi]])$$

applying the semantic equivalence:

$$\lozenge \Box \varphi \equiv \varphi \land \lozenge \lozenge (\lozenge \Box \varphi)$$

Thus, $[[\lozenge \Box \varphi]]$ is the fixpoint of the function:

$$F(X) = [[\varphi]] \cap \text{PRE}(X)$$
Theorem. Let $F(X) = \lceil \varphi \rceil \cap \text{PRE}(X)$, and let $S$ have $n + 1$ elements. Then:

1. $F$ is monotone;

2. $\lceil \diamond \Box \varphi \rceil$ is the greatest fixpoint of $F$.

Proof. (See the textbook “Logic in CS” pg.242)
Correctness and Termination: ♦ $\cup$ Case

The function $\textit{LABEL}_{EU}$ computes:

$$\llbracket ♦ (\varphi \cup ψ) \rrbracket = \llbracket ψ \rrbracket \cup (\llbracket ϕ \rrbracket \cap \text{PRE}(\llbracket ♦ (\varphi \cup ψ) \rrbracket))$$

applying the semantic equivalence:

$$♦ (\varphi \cup ψ) \equiv ψ \lor (\varphi \land ♦ \circ ♦ (\varphi \cup ψ))$$

Thus, $\llbracket ♦ (\varphi \cup ψ) \rrbracket$ is the \textit{fixpoint} of the function:

$$F(X) = \llbracket ψ \rrbracket \cup (\llbracket ϕ \rrbracket \cap \text{PRE}(X))$$
**Theorem.** Let $F(X) = \llbracket \psi \rrbracket \cup (\llbracket \varphi \rrbracket \cap \text{PRE}(X))$, and let $S$ have $n + 1$ elements. Then:

1. $F$ is monotone;
2. $\llbracket \diamond (\varphi \cup \psi) \rrbracket$ is the least fixpoint of $F$.

**Proof.** (See the textbook “Logic in CS” pg.243)
Summary of Lecture V

- CTL Model Checking: General Ideas.
- CTL Model Checking: The Labeling Algorithm.
- Labeling Algorithm in Details.
- CTL Model Checking: Theoretical Issues.