COMP424/524-06A
Topics in Software Engineering
Part I – Model Checking Algorithms
16. Model Checking CTL
Robi Malik

Marking Algorithm (1)

Marking for propositions
• Given by Kripke-Structure

Marking for propositional formulas
1. Apply the algorithm to all subformulas
2. Evaluate the connectives on each state, e.g.,
   all states not marked \( \varphi \) are marked \( \neg \varphi \).

Checking CTL Formulas

Given:
• Kripke-Structure \( K \)
• CTL Formula \( \varphi \)

Task:
• Identify the states of \( K \) where \( \varphi \) is true.

Example: \( \neg (req \Rightarrow g_f) \)

The Marking Algorithm

Idea:
Given a CTL formula, e.g.,

\[
AG (L_2 \Rightarrow AF (\neg L_1 \land \neg L_2))
\]

mark each state of \( K \) with all subformulas that are satisfied in that state.

Start with the innermost subformulas.

Marking Algorithm (2)

Marking for \( EX \varphi \)
1. Apply marking algorithm to \( \varphi \).
2. Mark states \( EX \varphi \), if they have at least one successor marked \( \varphi \).

Marking for \( AX \varphi \)
1. Apply marking algorithm to \( \varphi \).
2. Mark states \( AX \varphi \), if all their successors are marked \( \varphi \).
Marking Algorithm (3)

Marking for $EF \varphi$
1. Apply marking algorithm to $\varphi$.
2. Mark states $EF \varphi$, if they are marked $\varphi$.
3. Mark states $EF \varphi$, if they have at least one successor already marked $EF \varphi$.
4. Repeat step 3.

Marking Algorithm (4)

Marking for $A F \varphi$
1. Apply marking algorithm to $\varphi$.
2. Mark states $AF \varphi$, if they are marked $\varphi$.
3. Mark states $AF \varphi$, if all their successors are already marked $AF \varphi$.
4. Repeat step 3.

Example: $A (g_m U req)$

Example: $req \Rightarrow AF g_s$

Other Operators?

Use equivalences …

- $AX \varphi \equiv \neg EX \neg \varphi$
- $AG \varphi \equiv \neg EF \neg \varphi$
- $EG \varphi \equiv \neg AF \neg \varphi$
- $AF \varphi \equiv A (true U \varphi)$
- $EF \varphi \equiv E (true U \varphi)$

… or try direct algorithms.
Marking Algorithm (5)

**Marking for AG ϕ**
1. Apply marking algorithm to ϕ.
2. Mark states AG ϕ, if they are marked ϕ.
3. **Unmark** states AG ϕ, if they have a successor that is *not* marked AG ϕ.
4. Repeat step 3.

**Example:** AG (¬ (req ⇒ g))

<table>
<thead>
<tr>
<th>State</th>
<th>req</th>
<th>g</th>
</tr>
</thead>
<tbody>
<tr>
<td>r0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

And Similarly ...

Note: 
\[ AG \varphi \equiv \varphi \land AX AG \varphi \]

To solve this equation calculate ...

\[ X_0 = \text{true} \]
\[ X_1 = \varphi \land AX X_0 \]
\[ X_2 = \varphi \land AX X_1 \]
... 
\[ X_n = \varphi \land AX X_n \]

**Fixed Point Characterisations**

\[
\begin{align*}
AG \varphi &= \varphi \land AX AG \varphi \\
AF \varphi &= \varphi \lor AF \varphi \\
EG \varphi &= \varphi \lor EX EG \varphi \\
EF \varphi &= \varphi \lor EF \varphi \\
A(\varphi U \psi) &= \psi \lor (\varphi \land AX A(\varphi U \psi)) \\
E(\varphi U \psi) &= \psi \lor (\varphi \land EX E(\varphi U \psi)) 
\end{align*}
\]

Another Way of Doing It

Note: 
\[ EF \varphi \equiv \varphi \lor EX EF \varphi \]

To solve this equation calculate ...

\[ X_0 = \text{false} \]
\[ X_1 = \varphi \lor EX X_0 \]
\[ X_2 = \varphi \lor EX X_1 \]
... 
\[ X_n = \varphi \lor EX X_n \]

Reading

Bérard et. al.: Chapter 3 – Model Checking