Explicit Model Checking

Hao Zheng
Dept. of Computer Science & Eng.
Univ. of South Florida
Definition of Model Checking

• Given a $M$, and a specification $f$,

• Model checking problem
  – Find all states $s$ of $M$ such that $M, s |= f$.
  – And check that $I \subseteq \{s\}$.

• Efficient model checking algorithm:
CTL Model Checking

• Given a $M$ and a CTL formula $f$, $M \models f$?
  – $F$ holds on every initial state of $M$.
• Algorithms needed only for $\text{EX}$, $\text{EG}$, $\text{EU}$.
• Labeling procedure:
  – From atomic formulas, label states that hold the formula.
  – Label $s$ with $\neg f$ if $s$ is not labeled with $f$.
  – Label $s$ with $f \land g$ if $s$ is labeled with $f$ and $g$.
  – Label $s$ with $f \lor g$ if $s$ is labeled with $f$ or $g$.
  – Label $s$ with $\text{EX} f$ if there exists a $s \rightarrow s'$ such that $s'$ is labeled with $f$. 
CTL Model Checking (cont'd)

- Labeling procedure:
  - Label $s$ with $\mathbf{E}[f \mathbf{U} g]$ as follows:
    - Label all $s'$ with $\mathbf{E}[f \mathbf{U} g]$ that is already labeled with $g$;
    - Repeat: label $s$ with $\mathbf{E}[f \mathbf{U} g]$ if $s \rightarrow s'$ and $s$ is labeled with $f$.
  - Label $s$ with $\mathbf{E} \mathbf{G} f$:
    - Delete states where $f$ does not hold;
    - Find maximal strongly connected components (SCC);
    - Label every state in SCC with $\mathbf{E} \mathbf{G} f$;
    - Label other state $s$ with $\mathbf{E} \mathbf{G} f$ if $s \rightarrow s'$ and $s'$ is labeled with $f$. 
Pseudo Code

function SAT_{EX}( M, f ) begin
    f = true : return S;
    f = true : return ∅;
    f = atomic : return \{ s ∈ S \mid f ∈ L(s) \};
    f = ¬g : return S − SAT( g );
    f = g ∧ h : return SAT( g ) \cap SAT( f );
    f = g ∨ h : return SAT( g ) \cup SAT( h );
    f = g → h : return SAT( ¬g ∨ h );
    f = AX g : return SAT( ¬EX ¬g );
    f = EX g : return SAT( ¬EX ¬g );
    f = AG g : return SAT( ¬EF ¬g );
    f = EG g : return SAT_{EG}( g );
    f = AF g : return SAT( ¬EG ¬g );
    f = EF g : return SAT_{EU}( true, g );
    f = A(g U h): return SAT( ¬(E[¬h U (¬g ∧ ¬h)] \lor EG h));
    f = E[g U h] : return SAT_{EU}( g, h );
end;
function $\text{SAT}_{EX}(M, f)$
    begin
        $X = \text{SAT}(f)$;
        $Y = \text{pre}\exists( X )$;
        return $Y$;
    end;

$\text{pre}\exists( X ) = \{ s \in S \mid \exists s', \ s \rightarrow s' \text{ and } s' \in X \}$
Pseudo Code (cont'd)

function $\text{SAT}_{\text{EU}}(M, f, g)$
begin
    $X = \text{SAT}(g)$;
    $Y = \emptyset$;
    $Z = \text{SAT}(f)$;
    repeat until $X == Y$ begin
        $Y = X$;
        $X = X \cup \{ s | s \in \text{pre}_{\exists}(X) \text{ and } s \in Z \}$;
    return $Y$;
end;
function $\text{SAT}_{\text{EG}}(M, f)$
begin
$X = \text{SAT}(f);$  
$Y = Y \cup \text{SCC}(X);$  
$Z = \emptyset;$  
repeat until $Y == Z$ begin
  $Z = Y;$  
  $Y = Y \cup \{ s \mid s \in \text{pre}\exists( X ) \text{ and } s \in X \};$
return $Y;$
end;
An Example: Mutual Exclusion

\[ f = E[\neg c_2 \cup c_1] \]
An Example: Mutual Exclusion

\[ f = E[\neg c_2 U c_1] \]
An Example: Mutual Exclusion

\[ f = E[\neg c_2 \cup c_1] \]
An Example: Mutual Exclusion

\[ f = E[\neg c_2 U c_1] \]
CTL Model Checking with Fairness

• Models may contain paths impossible in reality.
  – Due to the limitation of modeling methods.
  – Ex.: A mutex can stay in critical section forever.
  – Would lead to wrong verification results.

• Fairness are CTL formulas that hold infinitely often on all paths.
  – Those that do not satisfy fairness are removed.

• Fairness example:
  – Mutex is not in its critical section $\text{AG AF } \neg (state=\text{critical})$. 
An Example: Mutual Exclusion

\[ f = \mathcal{A}[t_1 \rightarrow \mathcal{A}F \, c_1] \]
How to Verify with Fairness

- Find all SCCs in a model.
- Given a fairness formula $f$, remove any SCC where $f$ does not hold in any state.
- Any state in the restricted (fair) model that can reach one of remaining SCCs has a fair path from it.
Concepts of Fix-Points

- Given a set $S$ and a function $F: P(S) \rightarrow P(S)$, $F$ is a monotonic if $F(X) \subseteq F(Y)$ for $X \subseteq Y \subseteq S$.
- A fix point of a monotonic function $F$ is $X \subseteq S$ such that $X = F(X)$.
- Ex.: $S=\{s_0, s_1\}$, and $F(Y) = Y \cup \{s_0\}$.
  - $Y=\emptyset$, $F(Y) = \{s_0\}$, $F(F(Y)) = \{s_0\}$.
  - $Y= \{s_1\}$, $F(Y) = \{s_0, s_1\}$, $F(F(Y)) = \{s_0, s_1\}$.
- Ex.: $S=\{s_0, s_1\}$, and $G(Y) = \text{if } Y=\{s_0\} \text{ then } \{s_1\} \text{ else } \{s_0\}$.
  - Any fix point of $G(Y)$?
Concepts of Fix Points (cont'd)

- $X_G$ is the greatest fix point of $F$ if $X_G \supseteq X$ for all other fix points $X$ of $F$.

- $X_L$ is the least fix point of $F$ if $X_L \subseteq X$ for all other fix points $X$ of $F$. 
Concepts of Fix-Points (cont'd)

- $F^i(X) = F(...F(X)...)$

- Fixpoint theorem:
  - Let $S$ be a set, and $|S| = n$. If $F: P(S) \rightarrow P(S)$ is a monotonic function, then there exist $i$ and $j$ such $F^i(\emptyset)$ is the least fix point of $F$, and $F^j(S)$ is the greatest fix point of $F$.

- Every monotonic function has a least and a greatest fix points.

- It gives methods to calculate the fix points.
Fix-point Representation of CTL

• Let $[f]$ be the set of states satisfying $f$.
• $[\text{EX}f] = \text{pre}_\exists[f]$
• $[\text{EG}f] = [f] \cap [\text{EX EG}f]$
  – $\text{EG}f = f \land \text{EX Eg}f$
  – $[\text{EG}f]$ is the greatest fix point of $F(X) = [f] \cap \text{pre}_\exists[X]$.

• $[\text{E}(f \cup g)] = [g] \cup [\text{EX E}(f \cup g)]$
  – $\text{E}(f \cup g) = g \lor \text{EX E}(f \cup g)$
  – $[\text{E}(f \cup g)]$ is the least fix point of $F(X) = [g] \cup \text{pre}_\exists[X]$. 
LTL Model Checking

- Given a model $M$ and an LTL formula $f$
  - All traces of $M$ must satisfy $f$.
  - If a trace of $M$ does not satisfy $f$.
    - Countere-xample is generated

- LTL model checking: $\Sigma_M \subseteq \Sigma_f$
  - $\Sigma_M$ is the set of traces of $M$
  - $\Sigma_f$ is the set of traces that satisfy $f$
  - Note: a trace is a sequence of states.

- Equivalently $\Sigma_M \cap \Sigma_{\neg f} = \emptyset$
LTL Model Checking (cont'd)

- Construct an automaton accepting all traces for \( f, A_f \).
- Compose \( M \) and \( A_f \), \( M \parallel A_f \)
- If \( M \parallel A_f \) has no paths, then \( M \models \neg f \).
- Otherwise, the path in \( M \parallel A_f \) is the counter-example.
An Example

\[ M \models \neg(a \lor b) \ ? \text{ or } M \not\models (a \lor b) \ ? \]
Automaton for \((a \cup b)\)
$M \parallel (a \cup b)$
Constructing the Automaton

- Re-write LTL formulas in one of sufficient sets: \{U, X\}.
- Create closure \( C(f) \) of a formula \( f \):
  - Including all sub-formulas and their complements
  - Ex.: \( C(a \ U \ b) = \{ a, b, \neg a, \neg b, a \ U \ b, \neg(a \ U \ b) \} \).
- \( s \in S \) of \( A_f \) are \( P(C(f)) \) satisfying the following conditions:
  - Let \( g \in C(f) \), \( g \in s \) and \( \neg g \in s \) cannot hold at the same time.
  - If \( g_1 \lor g_2 \in C(f) \), \( g_1 \lor g_2 \in s \) if \( g_1 \in s \) or \( g_2 \in s \).
  - Other boolean combinations are handled similarly.
  - If \( g_1 \ U \ g_2 \in s \), then \( g_2 \in s \) or \( g_1 \in s \).
  - If \( \neg(g_1 \ U \ g_2) \in s \), then \( \neg g_2 \in s \).
Constructing the Automaton (cont'd)

• Initial states are those labeled with $f$.

• $(s, s') \in T$ of $A_f$ are constructed as follows:
  - If $Xg \in s$, then $g \in s'$.
  - If $\neg Xg \in s$, then $\neg g \in s'$.
  - If $g_1 \cup g_2 \in s$ and $g_2 \notin s$, then $g_1 \cup g_2 \in s'$.
  - If $\neg (g_1 \cup g_2) \in s$ and $g_1 \in s$, then $\neg (g_1 \cup g_2) \in s'$.

• The last two rules are due to the following equiv.:
  - $g_1 \cup g_2 = g_2 \lor (g_1 \land X(g_1 \cup g_2))$
  - $\neg (g_1 \cup g_2) = \neg g_2 \land (\neg g_1 \lor X \neg (g_1 \cup g_2))$
Automaton Construction: An Example

\[ f = Xa \quad \text{and} \quad C(f) = \{ a, \neg a, Xa, \neg Xa \} \]
Acceptance Conditions

• Used to define the eventuality condition of $a \mathbin{U} b$.
  – Not all states are accepting states.
  – Every state is an accepting state in $A_{xa}$.

• Accepting condition for $a \mathbin{U} b$:
  – Every path in $A_{a \mathbin{U} b}$ has infinitely many states satisfying
    $\neg(a \mathbin{U} b) \lor b$
  – This requires that $b$ must happen eventually for $a \mathbin{U} b$ to be accepted.
  – In other words, states in $A_{a \mathbin{U} b}$ labelled with either $\neg(a \mathbin{U} b)$
    or $b$ are accepting states.
Alternative Construction of $A_{a U b}$

- $C(a U b) = \{ a, \neg a, b, \neg b \}$
- States: $ab, \neg ab, a\neg b, \neg a\neg b$
- Transition $s \rightarrow s'$ exists if
  - $a \in L(s)$, and $a \in L(s')$ or $b \in L(s')$.
  - $b \in L(s)$, and don't-care for $L(s')$
$M \models A_a U b$ ?
An Exercise

\[ M \models \neg Fp \]
An Exercise (cont'd)

\[ M \models \neg Fp \Rightarrow M \models \neg( \text{TRUE} \cup p ) \]

\[ f = \text{TRUE} \cup p \]

\[ C(f) = \{ p, \neg p, f, \neg f \} \]
CTL Model Checking for LTL

• LTL model checking: if \((M \parallel A \neg f)\) is empty, then \(f\) holds on \(M\).

• If \(\neg f\) is regarded as fairness constraints, then \(f\) can be checked as follow:
  
  \(- \quad M_F \models EG \text{ true} \quad \text{holds, then} \quad f \quad \text{holds on} \quad M.\)
Reading List


