MODEL CHECKING

1. Introduction to model checking
2. CTL model checking
3. Implementation techniques
4. LTL model checking

E. M. Clarke et al.: *Model Checking*, Chapter 4 (pp. 35–49).
E. A. Emerson: *Automated Temporal Reasoning about Reactive Systems*, Section 3 (pp. 16–18).

Building a Model

For model checking of a system a possible world model $M$ can be generated from the system description using various techniques:

- **Explicate state methods:**
  The model $M$ is generated (before model checking) using reachability analysis techniques which build a reachability graph where each reachable state of the system is explicitly represented (state explosion).

- **On-the-fly techniques**
  The model $M$ is generated using reachability analysis techniques during model checking on demand.

- **Symbolic model checking:**
  The reachable states are represented symbolically using Boolean formulas.

1. Introduction to Model Checking

Question to be solved: Is a given formula $P$ true in a model $M$?

- Model $M$: a model of the system under validation
  The model is generated from a system description/specification which is given in a modelling/design/specification language: SDL, VHDL, process algebra, finite state automata, Petri nets, SMV, PROMELA, . . .

- Formula $P$: an interesting property of the system (system requirement)
  Often given in temporal logic: CTL, LTL, CTL*, . . .

- Model checking can be fully automated.
- Models of the realistic system often very big.
- Current techniques are scaling up to real applications

Representing the State Space Symbolically

- The (global) system states are given a binary representation ($n$ state bits).
  For each state bit $i$ two atomic propositions are introduced: $v_i$ (current state) and $v'_i$ (new state).

- For each state bit $i$ a formula is given specifying the transition relation from the current state to a new state, for example,
  
  
  $$v'_i \equiv (v_i \land v_{i+1}) \lor \neg v_{i+3}.$$ 

  The conjunction of all these formulas $T(\vec{v}, \vec{v}')$ gives the transition relation of the whole system.

- The formula $T(\vec{v}, \vec{v}')$ specifies in a symbolic form all possible state transitions: the system can move, for instance, from a state $(0, \ldots, 0)$ to a state $(1, \ldots, 1)$ iff $T(\vec{v}, \vec{v}')$ true in a truth assignment where all atoms $v_i$ are false and all atoms $v'_i$ are true.
Composing the Set of Reachable States

- Now a formula \( R(\vec{v}) \) expressing symbolically the set of reachable states of the system can be formed iteratively as follows:
  1. \( R_0(\vec{v}) := I(\vec{v}) \) where the formula \( I(\vec{v}) \) specifies the possible initial states of the system.
  2. repeat for all \( i = 1, 2, \ldots \), \( R_i(\vec{v}) := \exists w (R_{i-1}(\vec{w}) \land T(\vec{w}, \vec{v})) \) until \( R_i(\vec{v}) \equiv R_{i-1}(\vec{v}) \) (are logically equivalent).

Here \( \exists w R(w) \) is a shorthand for \( R(\top) \lor R(\bot) \) where \( R(\top) \) (\( R(\bot) \)) is the formula \( R(w) \) with atom \( w \) replaced by \( \top \) (\( \bot \)).

- The formula \( R(\vec{v}) \) specifies symbolically the reachable states: for example the state is \( (0, \ldots, 0) \) reachable if \( R(\vec{v}) \) is true in a truth assignment where all atoms \( v_i \) are false.

- In this way very large state spaces can be represented very compactly: for instance, \( R(\vec{v}) = \vec{v}_1 \) represent \( 2^n - 1 \) reachable states (i.e., all states where the state bit \( v_1 \) is true).
Example: Model Checking in SMV

$ smv example.smv
-- specification AF (state1 = n1 & state2 = s2) is false
-- as demonstrated by the following execution sequence
-- loop starts here --
state 1.1:
state1 = s1
state2 = s2
state 1.2:
state1 = n1
state2 = n2
state 1.3:
state1 = s1
state2 = s2

Global and Local Model Checking

- Global model checking:
  In which states of the model $M$ is the formula $P$ true?
- Local model checking:
  Is the formula $P$ true at a given state $s_0$ of the model $M$?
- Local model checking (in conjunction with on-the-fly techniques) enable an approach where not all (reachable) states of the model need to be examined (nor even generated).
- However, global model checking is more straightforward to implement and evaluation of formulas can be done more efficiently and with smaller memory requirements.

2. Global CTL Model Checking

- Global model checking methods determine the truth value of a formula in every state of the model.
- This can be done systematically by processing all the subformulas of the given formula starting from the atomic propositions in the following way:
  1. The subformulas of the formula $P$ are ordered in a sequence
     $P_0, P_1, \ldots, P_n (= P),$
     where each subformula $P_i$ appears only after all its proper subformulas have appeared in the sequence.
- Example. The subformulas of the formula $A(PUE(Q\neg P))$ can be ordered in such a sequence, e.g., as follows:
  $P, Q, \neg P, E(Q\neg P), A(PUE(Q\neg P)).$

CTL Model Checking

2. For all $i = 0, 1, \ldots, n$ determine the truth value of $P_i$ in every state $s \in S$ in the model $M$ as follows:
   (a) If $P_i$ is an atomic proposition, its truth value is obtained directly from the model.
   (b) If $P_i$ is of the form $\neg P_j$ or $P_j \land P_i$, its truth value can be computed from the truth values of the subformulas $P_j, P_i$ (as $j, l < i$, the truth values of $P_j, P_l$ have been determined).
   (c) If $P_i$ is of the form $AX P_j$, its truth value can be computed from the truth value of $P_j$ in states $s$ such that $sR$.
- Example. Let $M = (S, R, v)$ where $S = \{ s_0, s_1 \}$,
  $R = \{ (s_0, s_0), (s_0, s_1), (s_1, s_0) \}$, $v(s_0, P) = v(s_1, Q) = true$ and $v(s_1, P) = v(s_0, Q) = false$. Now $M, s_i \models \neg(P \land Q)$ when $i \in \{ 0, 1 \}$.
  Hence, e.g., $M, s_0 \models AX \neg((P \land Q)$ which also holds in the state $s_1$. 
2. (d) If $P_i$ is of the form $A(P_j U P_i)$, its truth value can be computed using the following equivalence:

$$A(P_j U P_i) \equiv P_i \lor (P_j \land AXA(P_j U P_i))$$

i. Mark $P_i$ true in all states where $P_i$ is true.
ii. Mark $P_i$ true in a state $s$ if $M,s \models P_j$ and $M,t \models P_i$ for all states $t$ for which $sRt$ until no new such states can be found.
iii. Mark $P_i$ false in all other states.

Example. Let $M = (S,R,v)$ where $S = \{s_0,s_1,s_2,s_3\}$,
$R = \{(s_0,s_1), (s_1,s_0), (s_0,s_2), (s_2,s_3), (s_3,s_1)\}$,
$v(s_i,P) = \text{true}$ if $i \neq 3$, and $v(s_i,Q) = \text{true}$ if $i = 3$.
Hence, $M,s_3 \models A(P U Q)$ by case (i) and $M,s_2 \not\models A(P U Q)$ case (ii). For all other states $s_i$ holds $M,s_i \not\models A(P U Q)$ by case (iii).

3. Implementation Techniques

- Next it is shown how to make the evaluation of temporal operators more efficient to reach the time complexity $O(|P| \cdot (|S| + |R|))$ (such that each operator is evaluated in time $O(|S| + |R|)$).
- Operators $E(P_j U P_i)$ and $EGP_j$ are taken as the basic operators. Notice that $A(P_j U P_i) \equiv \neg E(\neg P_j U (\neg P_j \land \neg P_i)) \land \neg EG \neg P_i$.
- For formulas of the form $E(P_j U P_i)$ the evaluation is based on using the accessibility relation $R$ backwards.
- The truth value of such a formula can be evaluated in time $O(|S| + |R|)$ with the CheckEU algorithm (see the next slide).

CheckEU: Evaluating $E(P_j U P_i)$ Formulas

procedure CheckEU($P_j, P_i$)
$T := \{s \mid M,s \models P_i\}$;
for all $s \in T$, label $E(P_j U P_i)$ true in $s$;
while $T$ is not empty do
choose $s$ in $T$ and remove it from $T$;
for all $t$ such that $(t,s) \in R$ do
if $E(P_j U P_i)$ is not yet labeled true in $t$ and $M,t \models P_i$ then
label $E(P_j U P_i)$ true in $t$;
add $t$ to $T$
end if
end for
endwhile
**Strongly Connected Components**

- Evaluating formulas of the form $E G P_j$ can be made more efficient by exploiting a technique where the model is partitioned into strongly connected components (SCGs).
- A strongly connected component of a graph is a maximal subgraph $C$ where every node is reachable from every other node in the subgraph through a path in $C$.
- A component $C$ is nontrivial iff it has more than one node or it consists of a node with an edge to itself.
- Strongly connected components can be found in linear time using Tarjan’s algorithm [SIAM J. of Computing, 1(2), 146–160, 1972].

---

**CheckEG: Evaluating $E G P_j$ Formulas**

procedure CheckEG ($P_j$)  
\[ S' := \{ s \in S \mid M, s \models P_j \}; \quad R' := \{ (s,t) \in R \mid s,t \in S' \}; \]

SCC := \{ $C$ | $C$ is a nontrivial SCC of ($S', R'$) \};  
\[ T := \{ s \mid s \in C \text{ and } C \in \text{SCC} \} ; \]

for all $s \in T$, label $E G P_j$ true in $s$;  
while $T$ is not empty do  
choose $s$ in $T$ and remove it from $T$;  
for all $t$ such that $t \in S'$ and $(t,s) \in R'$ do  
if $E G P_j$ is not yet labeled true in $t$ then  
label $E G P_j$ true in $t$; add $t$ to $T$  
end if  
end for  
endwhile

---

**Evaluating $E G P_j$ Formulas**

The evaluation is based on the restriction $M' = (S', R', v')$ of the model $M$ which is obtained from $M$ by removing the states where $P_j$ is false:

- $S' := \{ s \in S \mid M, s \models P_j \}$,
- $R' := \{ (s,t) \in R \mid s,t \in S' \}$ and
- $v'(s) = v(s)$ for all $s \in S'$.

The correctness of the evaluation builds on the following connections between models $M$ and $M'$:

**Lemma.** $M, s \models E G P_j$ iff $s \in S'$ and there is a path from $s$ to a state $t$ in $M'$ such that $t$ is in a nontrivial SCC of the graph ($S', R'$).

Now the truth value of $E G P_j$ can be evaluated in time $O(|S| + |R|)$ using the CheckEG algorithm (see the next slide).

---

**4. LTL Model Checking**

- Next we present a tableau based method for determining whether there is a full path in a given model $M$ where a given LTL formula $P$ is true.
- We write $M, s \models E P$ iff there is a full path starting from the state $s$ such that $P$ is true on the full path.
- Using this method we can answer also other LTL model checking questions:

**Example.** An LTL formula $P$ is true on every path starting from the state $s$ iff $M, s \models E \neg P$. 
**The Basic Idea of the Method**

- The question whether \( M,s \models EP \) holds is answered by constructing an LTL tableau (Büchi automaton) which describes every full path in the model \( M \) starting from the state \( s \) such that the formula \( P \) is true on the path.
- Given a tableau it is easy to check whether there is such a path.
- Reminder: we consider only operators \( \neg, \land, X, U \) (and other operators are seen as shorthands: for example \( P \lor Q = \neg(\neg P \land \neg Q) \); \( FP = \top U P \); \( GP = \neg F \neg P = \neg (T U \neg P) \)).
- For building LTL tableaux we use two auxiliary concepts:
  1. The closure \( \text{CL}(P) \) of a formula \( P \).
  2. Atoms \( (s,K) \) giving the nodes in an LTL tableau.

(See the next slides)

---

**The Closure of a Formula**

- The closure \( \text{CL}(P) \) of a formula \( P \) is the smallest set of formulas containing \( P \) and satisfying the following conditions:
  1. \( \neg P_1 \in \text{CL}(P) \) iff \( P_1 \in \text{CL}(P) \)
  2. If \( P_1 \land P_2 \in \text{CL}(P) \), then \( P_1, P_2 \in \text{CL}(P) \).
  3. If \( XP_1 \in \text{CL}(P) \), then \( P_1 \in \text{CL}(P) \)
  4. If \( \neg XP_1 \in \text{CL}(P) \), then \( X\neg P_1 \in \text{CL}(P) \)
  5. If \( P_1 UP_2 \in \text{CL}(P) \), then \( P_1, P_2, X(P_1 UP_2) \in \text{CL}(P) \)

(Here double negations are eliminated and, hence, a formula \( \neg \neg Q \) is identified with the formula \( Q \).)
- The idea is that the closure \( \text{CL}(P) \) of a formula \( P \) is the set of formulas that can affect the truth value of \( P \).

---

**Example**

Building closure \( \text{CL}((\neg H)U C) \) of \( (\neg H)UC \):

\[
\begin{align*}
(\neg H)UC & \quad \neg ((\neg H)UC) \\
H & \quad \neg H \\
C & \quad \neg C \\
X((\neg H)UC) & \quad \neg X((\neg H)UC) \\
X((\neg H)UC) & \quad \neg X((\neg H)UC)
\end{align*}
\]

(Hence, the closure \( \text{CL}(P) \) is an extended set of subformulas of \( P \) where for each formula included also its negation is present.)

---

**Atoms**

Consider a model \( M = (S,R,v) \) and a LTL formula \( P \).

An atom \( A = (s_A, K_A) \) is a pair where \( s_A \in S \) and \( K_A \subseteq \text{CL}(P) \cup AP \cup \{ \top \} \) (AP is the set of all atomic propositions) such that for the set of formulas \( K_A \):

1. for every atomic proposition \( P \in AP \cup \{ \top \}, P \in K_A \) iff \( M,s_A \models P \);
2. for every \( P_1 \in \text{CL}(P) \), \( P_1 \in K_A \) iff \( \neg P_1 \notin K_A \);
3. for every \( P_1 \land P_2 \in \text{CL}(P) \), \( P_1 \land P_2 \in K_A \) iff \( P_1 \in K_A \) and \( P_2 \in K_A \);
4. for every \( \neg XP_1 \in \text{CL}(P) \), \( \neg XP_1 \in K_A \) iff \( X\neg P_1 \in K_A \);
5. for every \( P_1 UP_2 \in \text{CL}(P) \), \( P_1 UP_2 \in K_A \) iff \( P_2 \in K_A \) or \( P_1, X(P_1 UP_2) \in K_A \).

Remark. When building atoms a formula \( \neg \neg Q \) is identified with the formula \( Q \).
**Building Atoms**

All possible atoms \((s, K)\) can be constructed using the following approach:

- The collection of possible sets of formulas \(K\) can be built using a (binary) tree (atom tableau), whose root is the set of atomic propositions and their negations true in the state \(s\).
- The tree can branch for each formula \(P_1 \in \text{CL}(P)\) into two branches where the other contains \(P_1\) and the other \(\neg P_1\). (In each set \(K\) for every \(P_1 \in \text{CL}(P)\) either \(P_1 \in K\) or \(\neg P_1 \in K\).)
- Other construction rules (see the next slide) add formulas guaranteeing that atoms satisfy the required conditions.

---

**Example**

Consider a formula \((\neg H)UC\), \(AP = \{H, C\}\), a model \(M\) and a state \(s_1\) where \(v(s_1, H) = v(s_1, C) = \text{false}\). The atomic tableau can be built as follows:

\[
\begin{array}{|c|c|c|}
\hline
T, \neg H, \neg C & \neg(\neg H)UC & \neg \neg(\neg H)UC \\
\hline
\neg C & H & \neg\neg(\neg H)UC \\
\times & \neg\neg(\neg H)UC & H \\
\hline
\end{array}
\]

Now possible atoms for the state \(s_1\) are \((s_1, K_1)\) and \((s_1, K_2)\) where \(K_1 = \{T, \neg H, \neg C, (\neg H)UC, X((\neg H)UC), \neg X(\neg(\neg H)UC)\}\) and \(K_2 = \{T, \neg H, \neg C, (\neg H)UC, \neg X((\neg H)UC), X(\neg(\neg H)UC)\}\).

---

**Rules to Construct an Atom Tableau**

- \(P_1 \in \text{CL}(P)\)
- \(P_1 | \neg P_1\)
- \(P_1 \land P_2\)
- \(\neg (P_1 \land P_2)\)
- \(\neg P_1 \land \neg P_2\)
- \(P_1 \lor P_2\)
- \(\neg(P_1 \lor P_2)\)
- \(X P_1\)
- \(\neg X P_1\)
- \(P_2\)
- \(\neg(P_2\land P_1\land P_2)\)
- \(\neg(X P_1 \land P_2)\)
- \(\neg\neg(X P_1 \land P_2)\)

- A branch closes if it contains a formula and its negation.
- The set of formulas \(K\) in an open branch which is finished (no new formulas can be added using the rules above and for which for every \(P_1 \in \text{CL}(P)\), \(P_1 \in K\) or \(\neg P_1 \in K\), is a valid set of formulas in an atom \((s, K)\).

---

**LTL Tableau**

**Definition.** Given a \(M = (S, R, v)\) and a formula \(P\), the **LTL tableau** is a graph \(G = (N, E)\) where

- the set of nodes \(N\) is the set of atoms constructed from the model \(M\) and the formula \(P\) and
- for the set of edge \(E \subseteq N \times N: (A, B) \in E\) iff
  1. \((s_A, s_B) \in R\) and
  2. for every \(XP_1 \in \text{CL}(P)\), \(XP_1 \in K_A\) iff \(P_1 \in K_B\).
Example

Consider a formula \((\neg H)UC\) and a model \(M = (S, R, v)\) where

- \(S = \{s_1, s_2\}\), \(R = \{(s_1, s_2), (s_2, s_1)\}\) and
- \(v(s_1, H) = v(s_1, C) = v(s_2, H) = v(s_2, C) = false\).

Then the LTL tableau is the graph \(G = (N, E)\) where

\[N = \{(s_1, K_1), (s_2, K_1), (s_1, K_2), (s_2, K_2)\}\]

and

\[E = \{((s_1, K_1), (s_2, K_1)), ((s_2, K_1), (s_1, K_1)), ((s_1, K_2), (s_2, K_2)), ((s_2, K_2), (s_1, K_2))\}\]

where sets \(K_1, K_2\) are above

\[K_1 = \{T, \neg H, \neg C, (\neg H)UC, X((\neg H)UC), \neg X((\neg H)UC)\}\]

and

\[K_2 = \{T, \neg H, \neg C, (\neg H)UC, \neg X((\neg H)UC), \neg X((\neg H)UC)\}\].

Eventuality Sequences

Definition. An eventuality sequence is an infinite path \(\pi\) in a LTL tableau \(G\) such that if \(P_1 \cup P_2 \in K_A\) for some atom \(A\) in the path \(\pi\), then there is an atom \(B\) which is reachable from \(A\) along the path \(\pi\) such that \(P_2 \in K_B\).

Example. The path \((s_1, K_1), (s_2, K_1), (s_2, K_1), (s_2, K_1), \ldots\) is not an eventuality sequence because \((\neg H)UC \in K_1\) and \(C \notin K_1\). On the other hand, \((s_1, K_2), (s_2, K_2), (s_2, K_2), (s_2, K_2), \ldots\) is an eventuality sequence. An eventuality sequence in an LTL tableau for a model \(M\) and a formula \(P\) provides a full path in \(M\) where the formula \(P\) is true.

Lemma. Let \(M\) be a model, \(P\) an LTL formula and \(G\) the corresponding LTL tableau. Then \(M, s \models EP\) if there is an eventuality sequence \(\pi\) in \(G\) starting at an atom \((s, K)\) such that \(P \in K\). Eventuality sequences can be found efficiently from an LTL tableau using self-fulfilling strongly connected components.

Self-Fulfilling SCCs

Definition. A strongly connected component \(C\) of a LTL tableau \(G\) is called self-fulfilling iff for every atom \(A \in C\) and every formula \(P_1 \cup P_2 \in K_A\) there is an atom \(B \in C\) such that \(P_2 \in K_B\).

Lemma. An LTL tableau \(G\) has an eventuality sequence starting at an atom \((s, K)\) iff there is a path from the atom \((s, K)\) to some self-fulfilling SCC of \(G\).

Example. (Cont’d) There is no eventuality sequence from the atom \((s_1, K_1)\) because there is no path from it to a self-fulfilling SCC. Notice that \(\{(s_2, K_1)\}\) is not self-fulfilling.

There is an eventuality sequence from the atom \((s_1, K_2)\) because there is a path from it to a self-fulfilling SCC \(\{(s_2, K_2)\}\).

Properties LTL Tableaux

Theorem. Let \(M\) be a model, \(P\) an LTL formula and \(G\) the corresponding LTL tableau.

Then \(M, s \models EP\) if there is an atom \((s, K)\) in \(G\) such that \(P \in K\) and there is a path in \(G\) from the atom \((s, K)\) to some self-fulfilling SCC of \(G\).

Example. (Cont’d) The LTL tableau \(G\) has no atom \((s_1, K)\) such that \((\neg H)UC \in K\) and there is a path from it to a self-fulfilling SCC.

Hence, \(M, s_1 \not\models E((\neg H)UC)\).
**LTL Model Checking Algorithm**

The theorem above provides a basis for the following LTL model checking algorithm whose time complexity is $O((|S| + |R|) \cdot 2^{O(|P|)})$.

To determine whether $M, s \models EP$ holds:

1. Construct the LTL tableau $G$ for $M, P$.
2. Compute the strongly connected components of $G$.
3. Identify the self-fulfilling SCCs.
4. Check for all atoms $(s, K)$ in $G$, where $P \in K$, if there is a path from the atom $(s, K)$ to some self-fulfilling SCC of $G$.
5. If such a path is found, then $M, s \models EP$ holds otherwise it does not hold.

---

**Summary**

- Typical model checking tools take as input a model given using a specification language supported by the checker and a temporal logic formula and give as output a notification that the formula is true in the model or otherwise a counter example.
- CTL and LTL are among the most widely applied temporal logics in model checking.
- CTL model checking can be done in linear time w.r.t. the size of the model and the size of the temporal formula.
- LTL model checking can be done in linear time w.r.t. the size of the model but even the best known methods take exponential time in the size of the temporal formula in the worst case.

---

**Computational Complexity**

- **CTL**
  - Model checking: **P-complete**
  - $O(|M| \cdot |P|)$
- **LTL**
  - Model checking: **PSPACE-complete**
  - $O(|M| \cdot \exp(|P|))$
- **CTL**
  - Model checking: **PSPACE-complete**
  - $O(|M| \cdot \exp(|P|))$