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).

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

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.

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,

  $$v'_i \leftrightarrow (v_i \wedge v_{i+1}) \vee \neg v_{i+3} \; .$$

  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 $i = 1, 2, \ldots$, $R_i(\vec{v}) := \exists \vec{w}(R_{i-1}(\vec{w}) \wedge T(\vec{w}, \vec{v}))$ until $R_i(\vec{v}) \equiv R_{i-1}(\vec{v})$ (are logically equivalent).
     Here $\exists w R(w)$ is a shorthand for $R(\top) \vee R(\bot)$ 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 $(0, \ldots, 0)$ is reachable iff $R(\vec{v})$ 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, $R(\vec{v}) = v_1$ represent 2^{n-1} reachable states (i.e., all states where the state bit v_1 is true).

Model Checkers

- take typically as input (i) a model given using a specification language supported by the checker and (ii) a formula given in a temporal logic supported by the checker (requirement specification);
- give as output a notification that the formula is true in (the initial states of) the model or a counter example (an execution path in the model where the formula is not true);
- and, hence, can also be used for debugging, i.e., for finding errors in system designs.

Example: SMV Model Checker

Implementation techniques

- Uses symbolic state space representation.
- Formulas representing the state space are manipulated in an efficient OBDD normal form (ordered binary decision diagrams).
- Model checking temporal logic formulas (CTL and LTL) is also done symbolically.

Example: a Model Given in the SMV Specification Language

```
MODULE main

VAR
state1: {s1, n1};
ASSIGN
init(state1) := s1;
next(state1) :=
case
  (state1 = s1) &
         (state2 = s2): n1;
  (state1 = n1) : {n1, s1};
  1: state1;
esac;
```

```
VAR
state2: {s2, n2};
ASSIGN
init(state2) := s2;
next(state2) :=
case
  (state1 = s1) &
         (state2 = s2): n2;
  (state2 = n2) : {n2, s2};
  1: state2;
esac;

SPEC
AF ((state1 = n1) &
    (state2 = s2))
```

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 is 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 $\mathbf{A}(P\mathbf{U}\mathbf{E}(Q\mathbf{U}\neg P))$ can be ordered in such a sequence, e.g., as follows:

$$P,\ Q,\ \neg P,\ \mathbf{E}(Q\mathbf{U}\neg P),\ \mathbf{A}(P\mathbf{U}\mathbf{E}(Q\mathbf{U}\neg P)).$$

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 $P_j \wedge P_l$, its truth value can be computed from the truth values of the subformulas P_j, P_l (as $j, l < i$, the truth values of P_j, P_l have been determined).
   (c) If P_i is of the form $\mathbf{AX}P_j$, its truth value can be computed from the truth value of P_j in states t such that sRt.

Example. Let $M = (S, R, v)$ where $S = \{s_0, s_1\}$, $R = \{\langle s_0, s_0\rangle, \langle s_0, s_1\rangle, \langle s_1, s_0\rangle\}$, $v(s_0, P) = v(s_1, Q) =$ true and $v(s_1, P) = v(s_0, Q) =$ false. Now $M, s_i \models \neg(P \wedge Q)$ when $i \in \{0, 1\}$.

Hence, e.g.,$M, s_0 \models \mathbf{AX}\neg(P \wedge Q)$ which also holds in the state s_1.

2. (d) If P_i is of the form $\mathbf{A}(P_j \mathbf{U} P_l)$, its truth value can be computed using the following equivalence:

$$\mathbf{A}(P_j \mathbf{U} P_l) \equiv P_l \vee (P_j \wedge \mathbf{AXA}(P_j \mathbf{U} P_l))$$

i. Mark P_i true in all states where P_l 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 $S = \{s_0, s_1, s_2, s_3\}$,
$R = \{\langle s_0, s_1\rangle, \langle s_1, s_0\rangle, \langle s_0, s_2\rangle, \langle s_2, s_3\rangle, \langle s_3, s_3\rangle\}$,
$v(s_i, P) = \text{true}$ iff $i \neq 3$, and $v(s_i, Q) = \text{true}$ iff $i = 3$.

Hence, $M, s_3 \models \mathbf{A}(P\mathbf{U}Q)$ by case (i) and $M, s_2 \models \mathbf{A}(P\mathbf{U}Q)$ case (ii). For all other states s_i holds $M, s_i \not\models \mathbf{A}(P\mathbf{U}Q)$ by case (iii).

2. (e) If P_i is of the form $\mathbf{E}(P_j \mathbf{U} P_l)$, its truth value can be computed using the following equivalence:

$$\mathbf{E}(P_j \mathbf{U} P_l) \equiv P_l \vee (P_j \wedge \mathbf{EXE}(P_j \mathbf{U} P_l))$$

i. Mark P_i true in all states where P_l is true.

ii. Mark P_i true in a state s if

$M, s \models P_j$ and $M, t \models P_i$ for some state t such that sRt,

no new such formulas can be found.

iii. mark P_i false in all other states.

- The time complexity of the algorithm is $O(|P| * |S| * (|S| + |R|))$.
- Evaluation of formulas starting with temporal operators can be made more efficient such that the time complexity is reduced to $O(|P| * (|S| + |R|))$.
- It is also possible to combine fairness constraints into global model checking.

3. Implementation Techniques

- Next it is shown how to make the evaluation of temporal operators more efficient to reach the time complexity $O(|P| * (|S| + |R|))$ (such that each operator is evaluated in time $O(|S| + |R|)$).
- Operators $\mathbf{E}(P_j \mathbf{U} P_l)$ and $\mathbf{EG}P_j$ are taken as the basic operators. Notice that

$$\mathbf{A}(P_j \mathbf{U} P_l) \equiv \neg\mathbf{E}(\neg P_l \mathbf{U} (\neg P_j \wedge \neg P_l)) \wedge \neg\mathbf{EG}\neg P_l.$$

- For formulas of the form $\mathbf{E}(P_j \mathbf{U} P_l)$ 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 $\mathbf{E}(P_j U P_l)$ Formulas

```
procedure CheckEU(P_j, P_l)
    T := {s | M,s ⊨ P_l};
    for all s ∈ T, label E(P_j U P_l) 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) ∈ R do
            if E(P_j U P_l) is not yet labeled true in t and M,t ⊨ P_j then
                label E(P_j U P_l) true in t;
                add t to T
            endif
        endfor
    endwhile
```

Strongly Connected Components

- Evaluating of formulas of the form $\mathbf{EG}P_j$ can be made more efficient by exploiting a technique where the model is partitioned into strongly connected components (SCCs).
- 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].

Evaluating $\mathbf{EG}P_j$ Formulas

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

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

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

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

☞ Now the truth value of $\mathbf{EG}P_j$ can be evaluated in time $O(|S| + |R|)$ using the CheckEG algorithm (see the next slide).

CheckEG: Evaluating $\mathbf{EG}P_j$ Formulas

procedure CheckEG(P_j)
  $S' := \{s \in S \mid \mathcal{M}, s \models P_j\}$; $R' := \{(s,t) \in R \mid s,t \in S'\}$;
  $SCC := \{C \mid C$ is a nontrivial SCC of $(S', R')\}$;
  $T := \{s \mid s \in C$ and $C \in SCC\}$;
  for all $s \in T$, label $\mathbf{EG}P_j$ 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 $\mathbf{EG}P_j$ is not yet labeled true in t then
        label $\mathbf{EG}P_j$ true in t; add t to T
      endif
    endfor
  endwhile

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 $M, s \models \mathbf{E}P$ 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 $M, s \not\models \mathbf{E}\neg P$.

The Basic Idea of the Method

- The question whether $M, s \models \mathbf{E}P$ 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 $\neg, \wedge, \mathbf{X}, \mathbf{U}$
  (and other operators are seen as shorthands: for example $P \vee Q = \neg(\neg P \wedge \neg Q)$; $\mathbf{F}P = \top \mathbf{U} P$; $\mathbf{G}P = \neg \mathbf{F} \neg P = \neg(\top \mathbf{U} \neg P)$).
- For building LTL tableaux we use two auxiliary concepts:
  1. The closure $\mathrm{CL}(P)$ of a formula P.
  2. Atoms (s, K) giving the nodes in a LTL tableau.

  (See the next slides)

The Closure of a Formula

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

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

Example

Building closure $\mathrm{CL}((\neg H)\mathbf{U}C)$ of $(\neg H)\mathbf{U}C$:

$$
\begin{array}{ll}
(\neg H)\mathbf{U}C & \neg((\neg H)\mathbf{U}C) \\
H & \neg H \\
C & \neg C \\
\mathbf{X}((\neg H)\mathbf{U}C) & \neg \mathbf{X}((\neg H)\mathbf{U}C) \\
\mathbf{X}\neg((\neg H)\mathbf{U}C) & \neg \mathbf{X}\neg((\neg H)\mathbf{U}C)
\end{array}
$$

(Hence, the closure $\mathrm{CL}(P)$ 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 $K_A \subseteq \mathrm{CL}(P) \cup \mathrm{AP} \cup \{\top\}$ (AP is the set of all atomic propositions) such that for the set of formulas K_A:

1. for every atomic proposition $P \in \mathrm{AP} \cup \{\top\}$, $P \in K_A$ iff $M, s_A \models P$;
2. for every $P_1 \in \mathrm{CL}(P)$, $P_1 \in K_A$ iff $\neg P_1 \notin K_A$;
3. for every $P_1 \wedge P_2 \in \mathrm{CL}(P)$, $P_1 \wedge P_2 \in K_A$ iff $P_1 \in K_A$ and $P_2 \in K_A$;
4. for every $\neg \mathbf{X} P_1 \in \mathrm{CL}(P)$, $\neg \mathbf{X} P_1 \in K_A$ iff $\mathbf{X} \neg P_1 \in K_A$;
5. for every $P_1 \mathbf{U} P_2 \in \mathrm{CL}(P)$, $P_1 \mathbf{U} P_2 \in K_A$ iff $P_2 \in K_A$ or $P_1, \mathbf{X}(P_1 \mathbf{U} P_2) \in K_A$.

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 $P_1 \in \mathrm{CL}(P)$ into two branches where the other contains P_1 and the other $\neg P_1$.

  (In each set K for every $P_1 \in \mathrm{CL}(P)$ 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.

Rules to Construct an Atom Tableau

$$\frac{P_1 \in \mathrm{CL}(P)}{P_1 \mid \neg P_1} \qquad \frac{P_1 \wedge P_2}{\begin{array}{c} P_1 \\ P_2 \end{array}} \qquad \frac{\neg(P_1 \wedge P_2)}{\neg P_1 \mid \neg P_2}$$

$$\frac{\mathbf{X}P_1}{\neg\mathbf{X}\neg P_1} \qquad \frac{\neg\mathbf{X}P_1}{\mathbf{X}\neg P_1} \qquad \frac{(P_1\mathbf{U}P_2)}{\begin{array}{c|c} P_2 & P_1 \\ & \mathbf{X}(P_1\mathbf{U}P_2) \end{array}} \qquad \frac{\neg(P_1\mathbf{U}P_2)}{\begin{array}{c|c} \neg P_2 & \neg P_2 \\ \neg P_1 & \neg\mathbf{X}(P_1\mathbf{U}P_2) \end{array}}$$

- 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 $P_1 \in \mathrm{CL}(P)$, $P_1 \in K$ or $\neg P_1 \in K$), is a valid set of formulas in an atom (s,K).

Example

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

$$\begin{array}{c}
\top, \neg H, \neg C \\
\begin{array}{c|c}
\begin{array}{c}
(\neg H)\mathbf{U}C \\
\begin{array}{c|c}
C & \neg H \\
\times & \mathbf{X}((\neg H)\mathbf{U}C) \\
 & \neg\mathbf{X}\neg((\neg H)\mathbf{U}C)
\end{array}
\end{array}
&
\begin{array}{c}
\neg((\neg H)\mathbf{U}C) \\
\begin{array}{c|c}
\neg C & \neg C \\
H & \neg\mathbf{X}((\neg H)\mathbf{U}C) \\
\times & \mathbf{X}\neg((\neg H)\mathbf{U}C)
\end{array}
\end{array}
\end{array}
\end{array}$$

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

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 $\mathbf{X}P_1 \in \mathrm{CL}(P)$, $\mathbf{X}P_1 \in K_A$ iff $P_1 \in K_B$.

Example

Consider a formula $(\neg H)\mathbf{U}C$ and a model $M = (S, R, v)$ where

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

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

$$N = \{\, (s_1, K_1), (s_2, K_1), (s_1, K_2), (s_2, K_2) \,\} \text{ and}$$
$$E = \{\, ((s_1, K_1), (s_2, K_1)), ((s_2, K_1), (s_2, K_1)), ((s_1, K_2), (s_2, K_2)), ((s_2, K_2), (s_2, K_2)) \,\}$$

where sets K_1, K_2 are above
$K_1 = \{\top, \neg H, \neg C, (\neg H)\mathbf{U}C, \mathbf{X}((\neg H)\mathbf{U}C), \neg\mathbf{X}\neg((\neg H)\mathbf{U}C)\}$ and
$K_2 = \{\top, \neg H, \neg C, \neg((\neg H)\mathbf{U}C), \neg\mathbf{X}((\neg H)\mathbf{U}C), \mathbf{X}\neg((\neg H)\mathbf{U}C)\}$.

Eventuality Sequences

Definition. An eventuality sequence is an infinite path π in a LTL tableau G such that if $P_1\mathbf{U}P_2 \in K_A$ for some atom A in the path π, then there is an atom B which is reachable from A along the path π 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 $(\neg H)\mathbf{U}C \in K_1$ 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 $M, s \models \mathbf{E}P$ iff there is an eventuality sequence π 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 $P_1\mathbf{U}P_2 \in K_A$ 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 $M, s \models \mathbf{E}P$ iff 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 $(\neg H)\mathbf{U}C \in K$ and there is a path from it to a self-fulfilling SCC.

Hence, $M, s_1 \not\models \mathbf{E}((\neg H)\mathbf{U}C)$.

LTL Model Checking Algorithm

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

To determine whether $M, s \models \mathbf{E}P$ 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 $M, s \models \mathbf{E}P$ holds otherwise it does not hold.

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

  $O(|M| \cdot exp(|P|))$

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.