Temporal Logic Model Checking

Specification Language
A propositional temporal logic.

Verification Procedure:
Exhaustive search of the state space of the concurrent system

E. M. Clarke and A. Emerson. Synthesis of synchronization skeletons for
branching time temporal logic. In Logic of Programs: Workshop, Yorktown Heights,

J. P. Quille and J. Sifakis. Specification and Verification of concurrent systems in
CESAR. In Proceedings of the Fifth International Symposium on Programming,
1981.

•
**Why Model Checking?**

**Advantages:**
- No proofs!!!
- Fast
- Counterexamples
- Noproblem with partial specifications
- Logics can easily express many concurrency properties
- Much progress recently!!

**Main Disadvantage: State explosion Problem**
- Too many processes
- Too many latches
- In digital hardware terms: too many latches

---

**Why Model Checking?**

---
Temporal Logic

State Transition Graph or Kripke Model

State Transition Graph to Obtained Infinite Tree

Infinite Computation Tree

(Unwind State Graph to Obtain Infinite Tree)
Computation Tree Logics

Formulas are constructed from path quantifiers and temporal operators:

1. Path quantifier:
   - $\forall$ — "for every path"
   - $\exists$ — "there exists a path"

2. Temporal Operator:
   - $X$ — holds next time.
   - $F$ — holds sometime in the future
   - $G$ — holds globally in the future
   - $U$ — holds until $d$ holds

$\Phi$ holds $d$ — $\Phi$ holds $d$ in the future
In CTL each temporal operator must be immediately preceded by a path quantifier.

The four most widely used CTL operators are illustrated below. Each computation tree has initial state $s_0$ as its root.
Typical CTL Formulas

• $\forall G (EF \text{ Restart})$: From any state it is possible to get to the Restart state.

- Computation Path.

• $\forall G (AF \text{ DeviceEnabled})$: DeviceEnabled holds infinitely often on every computation path.

• $\forall G (AF \text{ DeviceEnabled})$: DeviceEnabled holds infinitely often on every computation path.

• $\forall G (AF \text{ DeviceEnabled})$: DeviceEnabled holds infinitely often on every computation path.

- From any state it is possible to get back to the Ready state.

- Ready does not hold.

- Restart does not hold.

- EF (Started $\land$ Ready): It is possible to get to a state where Started holds but
Let $\mathcal{F}$ be the state-transition graph obtained from the concurrent system.

Let $\mathcal{W}$ be the specification expressed in temporal logic.

Find all states $s$ of such that $\mathcal{W}$ holds.

Efficient model checking algorithms exist for CTL.

Explicit Traversal

Preprocessor

Model Checker (EMC)

State Transition Graph

10^4 to 10^5 states

True or Counterexample

CTL formulas
Symbolic Model Checking

Method used by most "industrial strength" model checkers:

- BDDs traditionally used to represent boolean functions.
- Can handle much larger designs – hundreds of state variables.
- Uses boolean encoding for state machine and sets of states.

Symbolic Model Checking
Symbolic Model Checking with BDDs

Ken McMillan implemented a version of the CTL model checking algorithm using Binary Decision Diagrams (BDDs) in 1987. BDDs enabled handling much larger concurrent systems, usually an order of magnitude increase in hardware latch!s.

Carl Pixley independently developed a similar algorithm, as did the French researchers, Coudert and Madre.


Fixpoint Algorithms

\[ d \text{ EX EF} \land d = d \text{ EF} \]
Fixpoint Algorithms (cont.)

Key properties of $\mathcal{EF}$:

1. $\exists \mathcal{EF} \land d = \exists \mathcal{EF}$
2. $\exists \mathcal{EF} \land d = \exists \mathcal{EF}$
3. $\exists \mathcal{EF} \land d = \exists \mathcal{EF}$
4. $\exists \mathcal{EF} \land d = \exists \mathcal{EF}$

False $= 0 \mathcal{EF}$

How to compute $\mathcal{EF}$:

We write $\mathcal{EF} \cup d \land \text{Lfp} = \mathcal{EF} \cup d$.

$\cup \supset d \implies \mathcal{EF} \cup \exists \mathcal{EF} \land d = \cup$.

1. $\mathcal{EF} \land d = \mathcal{EF} \land d$
2. $\mathcal{EF} \land d = \mathcal{EF} \land d$

Key properties of $\mathcal{EF}$:
\[ \emptyset = ^0\Omega \]

\[ \mathcal{d} \mathcal{E} \models ^0s \mathcal{W} \]
$^0 \Omega \exists \land \Phi \land \downarrow d = ^1 \Omega$

$\vdash d \Phi \models ^0 s \uparrow \mathcal{M}$
\( \Omega \mathcal{E} \wedge d = \exists \Omega \)

\[
\models d \mathcal{E}I = 0s \models \mathcal{M}
\]
$\varepsilon_\Omega \mathbf{X} \varepsilon \land d = \varepsilon_\Omega$

$\mathcal{P} d \mathcal{E} \models 0s, \mathcal{W}$
Ordered Binary Decision Trees and Diagrams

Ordered Binary Decision Tree for the two-bit comparator, given by the formula:

\[ f(q_1, q_2, p_1, p_2) = ( q_1 \leftrightarrow q_2 ) \vee ( p_1 \leftrightarrow p_2 ) \]

is shown in the figure below:
An Ordered Binary Decision Diagram (OBDD) is an ordered decision tree where all isomorphic subtrees are combined, and all nodes with isomorphic children are eliminated. Given a parameter ordering, OBDD is unique up to isomorphism.

From Binary Decision Trees to Diagrams

If we use the ordering $a_2 > a_1 > b_2 > b_1$ for the comparator function, we obtain the OBDD below:

OBDD for Comparator Example
The size of an OBDD depends critically on the variable ordering. If we use the ordering $q_2 > q_1 > a_2 > a_1$ for the comparator function, we get the OBDD below:

\[
\begin{array}{c}
\text{Variable Ordering Problem}
\end{array}
\]
For an \( n \)-bit comparator:

Two \( n \) bit integers.

An example is the middle output (\( u \)th output) of a combinational circuit to multiply variables ordering.

Moreover, there are boolean functions that have exponential size OBDDs for any variable ordering.

If we use the ordering \( a_1 > a_2 > \cdots > a_q \), the number of vertices is

\[ 3n + 2 \]

If we use the ordering \( a_1 > a_2 > \cdots > a_q \), the number of vertices will be

\[ 3n - 1 \]
Logical operations on OBDD's

- Logical negation:
  $\neg f$ is constructed by replacing each leaf by its negation

- Logical conjunction:
  $f \land g$ is constructed via Shannon's expansion as follows:
  
  
  
  
  
  
  
  
  

- Always combine isomorphic subtrees and eliminate redundant nodes.

- To break problem into two subproblems, solve subproblems recursively.
  
  
  
  

- Number of subproblems bounded by $|b| \cdot |f|$.

- Hash table stores previously computed subproblems.

- Always combine isomorphic subtrees and eliminate redundant nodes.

  - Combine isomorphic subtrees.
  - Eliminate redundant nodes.

  - Combine isomorphic subtrees.
  - Eliminate redundant nodes.

-Hashtable stores previously computed subproblems.

  - Number of subproblems bounded by $|b| \cdot |f|$.

  - Always combine isomorphic subtrees and eliminate redundant nodes.

  - Combine isomorphic subtrees.
  - Eliminate redundant nodes.

Logical operations on OBDD's
Logical operations (cont.)

- Boolean quantification: \( \exists a : f(a, b, c, d) \)
  
  - By definition,
  
  \[ \exists a : f = f|_a \vee f|_{\neg a} \]

Using the above operations, we can build up OBDD's for complex boolean functions from simpler ones.
How to represent state-transition graphs with Ordered Binary Decision Diagrams:

Assume that system behavior is determined by n boolean state variables $v_1, v_2, \ldots, v_n$.

The Transition relation $T$ will be given as a boolean formula in terms of the state variables: $T(v_1, v_2, \ldots, v_n, v'_1, v'_2, \ldots, v'_n)$

where $v_1, \ldots, v_n$ represents the current state and $v'_1, \ldots, v'_n$ represents the next state.

Now convert $T$ to a OBDD!!
Symbolic Model Checking (cont.)

Representing transitions symbolically:

Boolean formula for transition relation:

\[
(q' \land p \lor q \lor r) \land \\
(q \land p \land q' \land r) \land \\
(q \land p \land q \land r')
\]

Now, represent as an OBDD!

Representing transitions symbolically:
Symbolic Model Checking (cont.)

Consider $\text{EX}$. Now, introduce state variables and transition relations:

\[
[\langle \rho \rangle d \lor \langle \rho ', \omega \rangle J \rangle, \rho E = \langle \omega \rangle f
\]

Consider $\text{OBDD}$ for relational product on right side of formula.

\[
\text{Compute OBDD for relational product on right side of formula.}
\]

Now, introduce state variables and transition relations:

\[
\text{Consider EX = f.}
\]
Convergence can be detected since the sets of states \( (\pi)^0 \cup (\pi)^1 \cup (\pi)^2 \) are represented as OBDDs.

until convergence.

\[ \vdots, (\pi)^2 \cup (\pi)^1 \cup (\pi)^0 \]

Now, compute the sequence

\[ [[(\pi) \cup (\pi, a) \cup] \land (a) \cup (a) \cup d \cdot \cup \text{d} \ldd = d \text{ E} \]

Introduce state variables:

\[ \cup \text{E} \land d \cdot \cup \text{d} \ldd = d \text{ E} \]

How to evaluate fixpoint formulas using OBDDs:

Symboilc Model Checking (cont.)
The following examples illustrate the power of model checking to handle industrial size problems. They come from many sources, not just my research group.

Notable Examples

In 1992 Clarke and his students at CMU used SMV to verify the cache coherence protocol in the IEEE Futurebus+ Standard. They constructed a precise model of the protocol and attempted to show that it satisfied a formal specification of cache coherence. They found a number of previously undetected errors in the design of the protocol. This was the first time that formal methods have been used to find errors in an IEEE Standard. Although development started in 1988, all previous attempts to validate Futurebus+ were based on informal techniques.
A High-level Data Link Controller (HDLC) was being designed at AT&T in Madrid.

Notable Examples–HDLC

- The error was corrected in a few minutes and formally verified.
- The sixth property failed, uncovering a bug that would have reduced throughput or caused lost transmissions.
- Within five hours, six properties were specified and five were verified, using the FormalCheck verifier.
- In 1996 researchers at Bell Labs offered to check some properties of the design. The design was almost finished, so no errors were expected.
- HDLC was being designed at AT&T in Madrid.
Richard Raimi and Jim Lear at Somerst used Motorola's Verdict model checker to debug a hardware laboratory failure. With run time in seconds, Verdict produced an example of BIU deadlock causing the failure. Initial silicon of PowerPC 620 microprocessor crashed during boot of an operating system. Paper on this published at 1997 IEEE International Test Conference. Notable Examples—PowerPC 620 Microprocessor
Future Research Directions

Additional work needed on classical model checking:

- Parameterized Designs.
- Symmetry, and
- Compositional Reasoning.
- Abstraction.