Lecture 1: Symbolic Model Checking with BDDs

Edmund M. Clarke, Jr.
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
Temporal Logic Model Checking

**Specification Language:** A propositional temporal logic.

**Verification Procedure:** Exhaustive search of the state space of the concurrent system to determine truth of specification.


Why Model Checking?

Advantages:

- No proofs!!
- Fast
- Counterexamples
- No problem with partial specifications
- Logics can easily express many concurrency properties

Main Disadvantage: State Explosion Problem

- Too many processes
- In digital hardware terms: too many latches

Much progress recently!!
Temporal Logic

State Transition Graph or Kripke Model

Infinite Computation Tree

(Unwind State Graph to obtain Infinite Tree)
Formulas are constructed from path quantifiers and temporal operators:

1. Path quantifier:
   - A—“for every path”
   - E—“there exists a path”

2. Temporal Operator:
   - Xp—p holds next time.
   - Fp—p holds sometime in the future
   - Gp—p holds globally in the future
   - pUq—p holds until q holds
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

- **EF**(Started ∧ ¬Ready): it is possible to get to a state where Started holds but Ready does not hold.
- **AG**(Req ⇒ AF Ack): if a Request occurs, then it will be eventually Acknowledged.
- **AG(AF DeviceEnabled)**: DeviceEnabled holds infinitely often on every computation path.
- **AG(EF Restart)**: from any state it is possible to get to the Restart state.
Model Checking Problem

Let $M$ be the state–transition graph obtained from the concurrent system.

Let $f$ be the specification expressed in temporal logic.

Find all states $s$ of $M$ such that

$$M, s \models f$$

and check if initial states are among these.

Efficient model checking algorithms exist for CTL.

Explicit Traversal

Preprocessor

Model Checker
(EMC)

State Transition Graph
$10^4$ to $10^5$ states

CTL formulas

True or Counterexample
Symbolic Model Checking

Method used by most “industrial strength” model checkers:

- uses boolean encoding for state machine and sets of states.

- can handle much larger designs – hundreds of state variables.

- BDDs traditionally used to represent boolean functions.
Ken McMillan implemented a version of the CTL model checking algorithm using Binary Decision Diagrams in 1987.

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

BDDs enabled handling much larger concurrent systems. (usually, an order of magnitude increase in hardware latches!)

Fixpoint Algorithms

\[ \text{EF} p = p \lor \text{EX} \text{EF} p \]
Key properties of $\text{EF} p$:

1. $\text{EF} p = p \lor \text{EX} \text{EF} p$
2. $U = p \lor \text{EX} U$ implies $\text{EF} p \subseteq U$

We write $\text{EF} p = \text{Lfp} U.p \lor \text{EX} U$.

How to compute $\text{EF} p$:

\[
\begin{align*}
U_0 &= \text{False} \\
U_1 &= p \lor \text{EX} U_0 \\
U_2 &= p \lor \text{EX} U_1 \\
U_3 &= p \lor \text{EX} U_2 \\
&\vdots
\end{align*}
\]
$M, s_0 \models \text{EF } p$?

$U_0 = \emptyset$
$M, s_0 \models \textbf{EF} \ p?$

$U_1 = p \lor \textbf{EX} \ U_0$
$M, s_0 \models \textbf{EF } p$?

$U_2 = p \lor \textbf{EX } U_1$
$M, s_0 \models \text{EF } p$?

$U_3 = p \lor \text{EX } U_2$
Ordered Binary Decision Trees and Diagrams

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

\[ f(a_1, a_2, b_1, b_2) = (a_1 \leftrightarrow b_1) \land (a_2 \leftrightarrow b_2), \]

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.

If we use the ordering $a_1 < b_1 < a_2 < b_2$ for the comparator function, we obtain the OBDD below:
Variable Ordering Problem

The size of an OBDD depends critically on the variable ordering.

If we use the ordering $a_1 < a_2 < b_1 < b_2$ for the comparator function, we get the OBDD below:
For an \( n \)-bit comparator:

- if we use the ordering \( a_1 < b_1 < \ldots < a_n < b_n \), the number of vertices will be \( 3n + 2 \).
- if we use the ordering \( a_1 < \ldots < a_n < b_1 \ldots < b_n \), the number of vertices is \( 3 \cdot 2^n - 1 \).

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

An example is the middle output (\( n^{th} \) output) of a combinational circuit to multiply two \( n \) bit integers.
Logical operations on OBDD’s

- **Logical negation:** $\neg f(a, b, c, d)$
  
  Replace each leaf by its negation

- **Logical conjunction:** $f(a, b, c, d) \land g(a, b, c, d)$
  
  – Use Shannon’s expansion as follows,
  
  $$f \cdot g = \bar{a} \cdot (f|\bar{a} \cdot g|\bar{a}) + a \cdot (f|a \cdot g|a)$$

  to break problem into **two subproblems**. Solve subproblems recursively.

  – Always combine isomorphic subtrees and eliminate redundant nodes.

  – Hash table stores previously computed subproblems

  – Number of subproblems bounded by $|f| \cdot |g|$.
• **Boolean quantification:** \( \exists a : f(a, b, c, d) \)

  – By definition,

\[
\exists a : f = \bar{f}_{\bar{a}} \lor f_{a}
\]

  – \( f(a, b, c, d)_{\bar{a}} \): replace all \( a \) nodes by left sub-tree.

  – \( f(a, b, c, d)_{a} \): replace all \( a \) nodes by right sub-tree.

Using the above operations, we can build up OBDD’s for complex boolean functions from simpler ones.
Symbolic Model Checking Algorithm

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:

$$T(v_1, \ldots, v_n, v'_1, \ldots, v'_n)$$

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 transition relations symbolically:

Boolean formula for transition relation:

\[ (a \land \neg b \land a' \land b') \lor (a \land b \land a' \land b') \lor (a \land b \land a' \land \neg b') \]

Now, represent as an OBDD!
Consider $f = \mathbf{EX} p$.

Now, introduce state variables and transition relation:

$$f(\vec{v}) = \exists \vec{v}' [T(\vec{u}, \vec{v}') \land p(\vec{v}')]$$

Compute OBDD for relational product on right side of formula.
Symbolic Model Checking (cont.)

How to evaluate fixpoint formulas using OBDDs:

\[ \text{EF} p = \text{Lfp } U. p \lor \text{EX } U \]

Introduce state variables:

\[ \text{EF} p = \text{Lfp } U. p(\bar{v}) \lor \exists \bar{v}' \left[ T(\bar{v}, \bar{v}') \land U(\bar{v}') \right] \]

Now, compute the sequence

\[ U_0(\bar{v}), U_1(\bar{v}), U_2(\bar{v}), \ldots \]

until convergence.

Convergence can be detected since the sets of states \( U_i(\bar{v}) \) are represented as OBDDs.
Notable Examples

The following examples illustrate the power of model checking to handle industrial size problems.

**They come from many sources, not just my research group.**

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.

In 1996 researchers at Bell Labs offered to check some properties of the design. The design was almost finished, so no errors were expected.

Within five hours, six properties were specified and five were verified, using the FormalCheck verifier.

The sixth property failed, uncovering a bug that would have reduced throughput or caused lost transmissions.

The error was corrected in a few minutes and formally verified.
Notable Examples–PowerPC 620 Microprocessor

- Richard Raimi and Jim Lear at Somerset used Motorola’s Verdict model checker to debug a hardware laboratory failure.
- Initial silicon of PowerPC 620 microprocessor crashed during boot of an operating system.
- With run time in seconds, Verdict produced example of BIU deadlock causing the failure.
- Paper on this published at 1997 IEEE International Test Conference.
Future Research Directions

Additional work needed on classical model checking:

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