For simplicity, we will refer to ROBDD simply as BDD.

A BDD is ordered (OBDD) if the variables respect a given linear order.

A binary decision diagram (BDD) is a rooted, directed acyclic graph with

Definitions

In general, it requires an exponential number of nodes.

Optimize

Identify identical subgraphs.

Remove redundant tests.

Yielding:

Uniqueness — no two distinct nodes are roots of isomorphic subgraphs.

No redundant tests — low \( (n \) high \( (n) \)

\( \text{low} (n) \text{ high} (n) \) on all paths through the graph. An OBDD is reduced (ROBDD) if for all nodes \( n \) in the graph:

One or two nodes of out-degree zero (leaves) labeled \( 0 \) or \( 1 \), and

A set of variable nodes of out-degree 2. The two outgoing edges are given by the functions low \( (u) \) and high \( (u) \).

A BDD is reduced (ROBDD) if the variables respect a given linear order.

\( \text{low} (u) \neq \text{high} (u) \) for all nodes \( u \) in the graph.

transition relation.

A key development for symbolic model checking was the development of binary decision diagrams (BDD) as an efficient representation of boolean assertions.

decision diagrams (BDD) are an efficient representation of boolean assertions.

Next, we consider the symbolic approach to model checking. Note that every assertion on a finite-domain FDS can be represented as a Boolean formula over

We start with a binary decision diagram. For example, following is a decision diagram (tree) for the formula \( \varphi = \tau x \lor \theta \).

\( \varphi = \tau x \lor \theta \).
Claim 12: For every function \( f : \text{Bool}^n \rightarrow \text{Bool} \) and variable ordering \( x_1 < x_2 < \ldots < x_n \), there exists exactly one BDD representing this function.

The complexity of BDD representation is very sensitive to the variable ordering. For example, the BDD representation of \( (x_1 = y_1) \land (x_2 = y_2) \) under the variable ordering \( x_1 < x_2 < y_1 < y_2 \) is:

\[
\begin{array}{c}
\text{Sensitivity to Variable Ordering}
\end{array}
\]

The internal representation of a BDD is shown in the diagram. The operations for initializing and manipulating BDDs are described in the text:

- \( \text{init}(T) \): Initialize the BDD
- \( \text{alloc}(T) \): Allocate a new node
- \( \text{H}(T(u)) = T^{-1}(u) \): The inverse of the BDD
- \( \text{var}(u), \text{low}(u), \text{high}(u) \): Abbreviations for variable, low, and high

The implementation of BDD packages is detailed in the text, including types, variables, and record structures.
Universal quantification can be completed dually:

\[ [t \leftarrow x] : \forall : t : x \]  

Existential quantification can be completed, using the equivalence

\[ \exists : t : x \]  

substituting \( t \) in assertion \( x \) for the result of

\[ \text{return } \text{res}(n) \]
Lecture 6: Symbolic Model Checking

A. Pnueli

Application of BDD's to Symbolic Model Checking

Let $V$ be the state variables for the FDS $D$. Taking a vocabulary $U = V \cup \{V_0\}$, we represent the state formulas $\phi$ for each $i \in C$, and the INV symbolic working variables $new$ and $old$ as BDD's over which are $\Lambda = \Lambda \lor (\Lambda)\phi : \Delta E = (\phi)\prime$.

Priming an assertion $\phi$ is performed by

$(\Lambda)\phi \lor (\Lambda')\phi : \Delta E = \phi \lor \phi'$

The existential pre-condition transformer is computed by

To check for equivalence such as $new = old$, we compute $t = new \iff ppo$. All the boolean operations used in the INV algorithm can be implemented by the following operations:

1. Negation can be computed by $t = t_1$ where $t \oplus t = 1$.
2. Applying a function $f$. All the boolean operations used in the INV algorithm can be implemented by the following operations:

By definition, $t \oplus t = 1$, where $t \oplus t = 1$. The transition relation $\Delta$ is represented as a BDD over $\Lambda$, which may be fully dependent on both $\Lambda$ and $\Lambda$. Analysis of Reactive Systems, NYU, Fall, 2009.