Systems and Software Verification

Chapter 4. Symbolic Model Checking

Lecturer: JUNBEOM YOO
jbyoo@konkuk.ac.kr
http://dslab.konkuk.ac.kr

Ver. 2.0

4. Symbolic Model Checking

- Symbolic model checking
  - Any model checking method attempting to represent symbolically states and transitions
  - A particular symbolic method in which BDDs are used to represent the state variables
    - BDD : Binary Decision Diagram

- Motivation:
  - State explosion is the main problem for CTL or PLTL model checking
  - State explosion occurs whenever we represent explicitly all states of automaton we use

  - Represent very large sets of states concisely, as if they were in bulk.

- Organization of chapter 4
  - Symbolic Computation of State Sets
  - Binary Decision Diagrams (BDD)
  - Representing Automata by BDDs
  - BDD-based Model Checking

4.1 Symbolic Computation of State Sets

- Iterative computation of $Sat(\phi)$
  - $A = \langle Q, T, \ldots \rangle$
  - $Pre(S)$: immediate predecessors of the states belonging to S in Q
  - $Sat(\phi)$: set of states of A which satisfy ϕ
  - ψ is the sub-formulas of ϕ

  - $Sat(\neg\psi) = Q \setminus Sat(\psi)$
  - $Sat(\psi \wedge \psi') = Sat(\psi) \cap Sat(\psi')$
  - $Sat(EX\ \psi) = Pre(Sat(\psi))$
  - $Sat(AX\ \psi) = Q \setminus Pre(Q \setminus Sat(\psi))$
  - $Sat(EF\ \psi) = Pre^*(Sat(\psi))$
  - … (others are defined in a similar way)

```
/* ==== Computation of Pre*(S) ==== */
X := S;
Y := { };
while (Y != X) {
   Y := X;
   X := X ∨ Pre(X);
}
return X;
```

  - The algorithms in Section 3.1 is an particular implementation of $Sat(\phi)$

  - Hence, $Sat(\phi)$ is an <u>explicit representation</u> of the state sets

- Which symbolic representations to use ?
  - We have to access the following primitives:
    1. A symbolic representation of $Sat(P)$ for each proposition $P \in Prop$,
    2. An algorithm to compute a symbolic representation of $Pre(S)$ from a symbolic representation of S,
    3. Algorithms to compute the complement, the union, and the intersection of the symbolic representations of the sets,
    4. An algorithm to tell whether two symbolic representations represent the same set.

- Which logic for symbolic model checking?
  - Logics based on state formulas
  - CTL is the best.
  - Mu-calculus on tree is possible.

- Systems with infinitely many states
  - Symbolic approach naturally extends to infinite systems.
  - New difficulties:
    1. Much trickier to come up with symbolic representations
    2. Iterative computation $Sat(\phi)$ is no longer guaranteed to terminate.

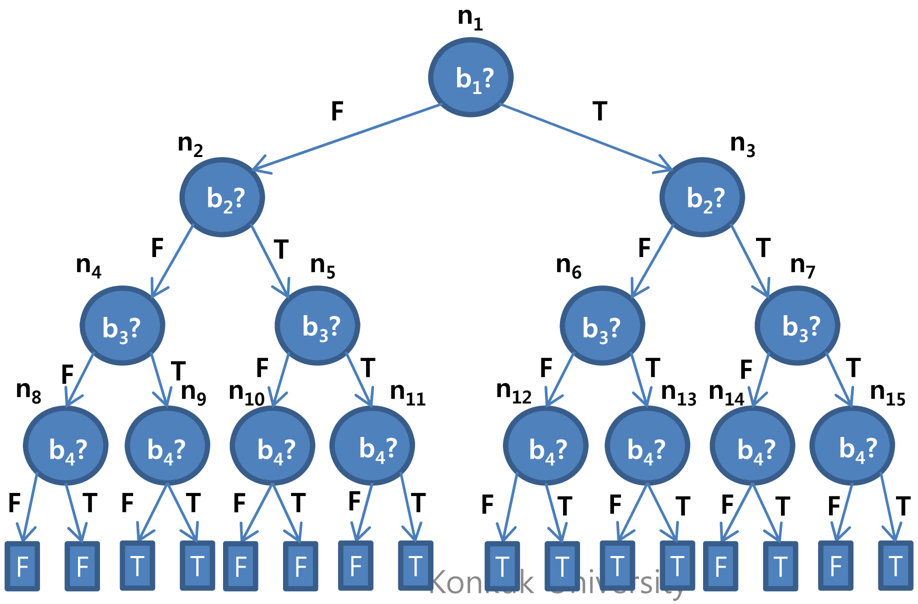
4.2 Binary Decision Diagram (BDD)

- BDD
  - A particular data structure very commonly used for representing states sets symbolically
  - Proposed in 1980s ~ early in 1990s

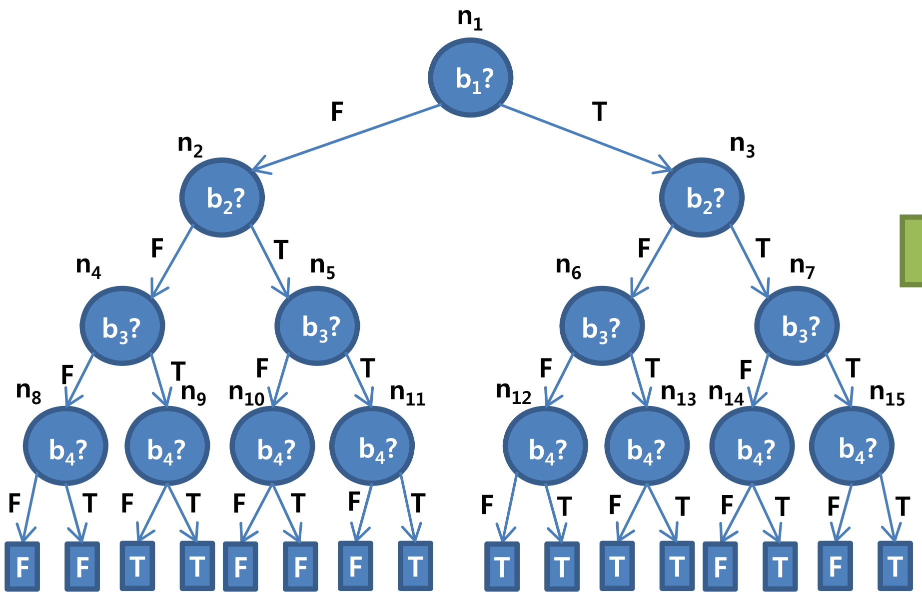
  - Make possible the verification of the system which cannot represent explicitly.

  - Advantages:
    1. Efficiency
    2. Simplicity
    3. Easy Adaptation
    4. Generality

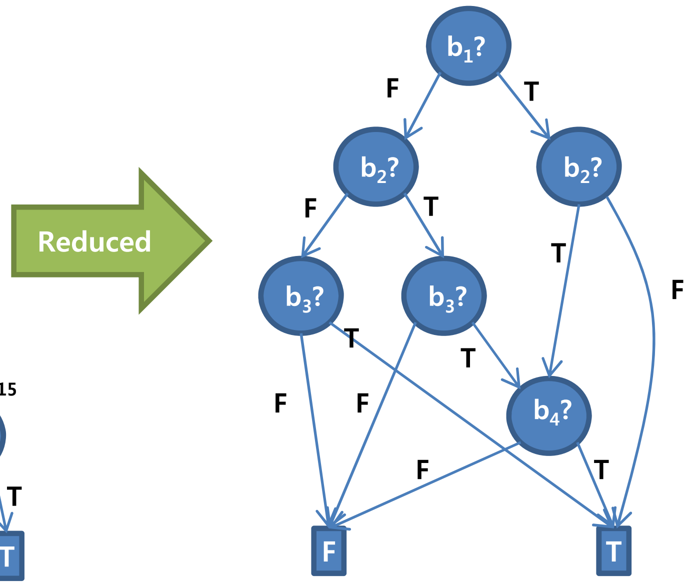
- BDD structure
  - Example
    - Consider n boolean variables $x_1, x_2, \dots, x_n$ associated with a tuple $< b_1, b_2, \dots, b_n >$
    - Suppose n = 4,
    - The set S of our interest is the set such that $(b_1 \vee b_3) \wedge (b_2 \Rightarrow b_4)$ is true.
    - We have several ways to represent the set:
      - S = {<F,F,T,F>, <F,F,T,T> , … >
      - S = $(b_1 \vee b_2) \wedge (b_3 \Rightarrow b_4)$
      - S = $(b_1 \wedge \neg b_2) \vee (b_1 \wedge b_4) \vee (b_3 \wedge \neg b_2) \vee (b_3 \wedge b_4)$ ← DNF
      - …
      - Decision Tree ← Our choice.



- Decision tree reduction
  - A <u>BDD</u> is a reduced decision tree.
  - Reduction rules:
    1. Identical sub-trees are identified and shared. (n_8 and n_{10})
       → leads to a directed acyclic graph (dag)
    2. Superfluous internal nodes are deleted. (n_7)
  - Advantages:
    1. Space saving
    2. Canonicity



Decision tree



BDD

- Canonicity of BDDs
  - BDDs canonically represent sets of boolean tuples. (fundamental property of BDDs)
  - If the order of the variable x_i is fixed, then there exists a unique BDD for each set S.

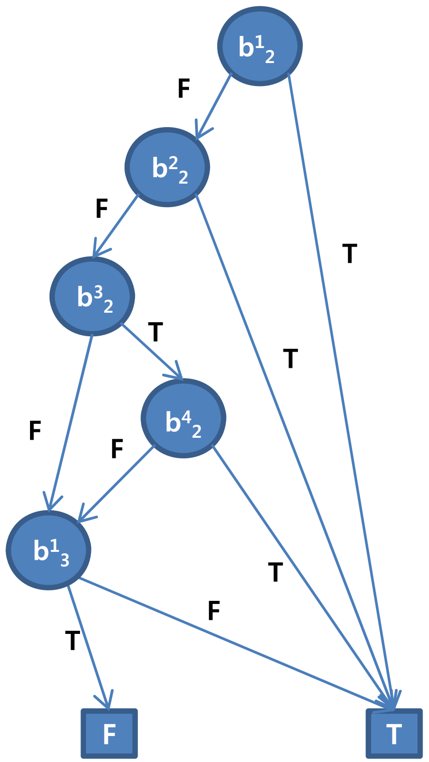
  - Properties of BDDs
    1. We can test the equivalence of two BDDs in constant time.
    2. We can tell whether a BDD represents the empty set simply by verifying whether it is reduced to a unique leaf F.

- Operations on BDDs
  - All boolean operations
    1. Emptiness test
    2. Comparison
    3. Complementation
    4. Intersection
    5. Union and other binary boolean operations
    6. Projection and abstractions
  - Complexity : linear or quadratic (for each operation)
    → the same state explosion problems still exist.

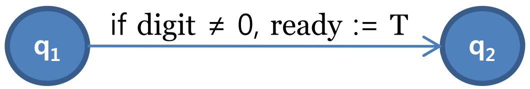
4.3 Representing Automata by BDDs

- Before applying BDDs to symbolic model checking, we need to restate
  - Representing the states by BDDs
  - Representing transitions by BDDs

- Representing the states by BDDs
  - Consider an automaton A with
    - $Q = \{q_0, \ldots, q_6\}$ → $b^1{}_1, b^2{}_1, b^3{}_1$
    - var digit:0..9 → $b^1{}_2, b^2{}_2, b^3{}_2, b^4{}_2$
    - var ready:bool → $b^1{}_3$
    - $< b^1{}_1, b^2{}_1, b^3{}_1, b^1{}_2, b^2{}_2, b^3{}_2, b^4{}_2, b^1{}_3 >$
    - $<$ F, T, T, T, F, F, F, F $> = <q_3$, 8, F $>$
  - Let's represent Sat(ready ⇒ (digit > 2))
    - States $<q, k, b>$ such that if b = T and k > 2
    - ready ⇒ (digit > 2) ≡ ¬ ready ∨ (digit > 2)



- Representing transitions by BDDs
  - The same idea is applied.
  - $<q_3, 8, F> \rightarrow <q_5, 0, F>$: $<$ F, T, T, T, F, F, F, F, T, F, T, F, F, F, F, F $>$
  - For example,



  - $(<q, k, b>, <q', k', b'>)$
    $\rightarrow q = q_1, k \neq 0, q' = q_2, k' = k, b' = T$

    $\rightarrow (\neg b^1_1 \wedge \neg b^2_1 \wedge b^3_1)$
    $\wedge (b^1_2 \vee b^2_2 \vee b^3_2 \vee b^4_2)$
    $\wedge (\neg b'^1_1 \wedge \neg b'^2_1 \wedge b'^3_1)$
    $\wedge (b'^1_2 \Leftrightarrow b^1_2 \wedge b'^2_2 \Leftrightarrow b^2_2 \wedge b'^3_2 \Leftrightarrow b^3_2 \wedge b'^4_2 \; b^4_2)$
    $\wedge b'^1_3$

4.4 BDD-based Model Checking

- BDDs can serve as an instance of symbolic model checking scheme
  - Provide compact representations for the sets of states in an automata
  - Support the basic sets of operations
  - Computation of $Pre(S)$ in section 4.1 is very simple

- Implementation
  - SMV (chapter 12)
  - Efficiency of BDDs depends on
    - B_T representing the transition relation T (as containing pairs of states)
    - Choice of ordering for the boolean variables
  - Very easy to explode exponentially

- Perspective
  - Widely used from early 1990s
  - Current work on model checking
    - Aiming at applying BDD technology to solve more verification problems (ex. program equivalence)
    - Aiming at extending the limits inherent to BDD-based model checking
  - Widely used throughout the VLSI design industry