

ECE 510 OCE
BDDs and Their Applications

**Lecture 11. FSM Equivalence Checking
and FSM State Minimization**

May 2, 2000
Alan Mishchenko

Overview

- Equivalence for FSMs and FSM states
- Product machine (PM)
- Solving the problem of FSM equivalence
 - Derive transition and output relation of the PM
 - Perform reachability on the PM and verify property Output = 1
 - Generate an error trace if the equivalence check has failed
- Solving the problem of FSM state minimization
 - Derive transition and output relation of the PM
 - Perform reachability on the PM and derive the state equivalence relation
 - Transform the initial FSM's transition and output relations
- Compatible projection operator

May 2, 2000 ECE 510 OCE: BDDs and Their Applications 2

FSM Equivalence

- Definition. Two state machines are equivalent, if starting from their reset states, for any sequence of input vectors, they produce identical sequences of output vectors

May 2, 2000 ECE 510 OCE: BDDs and Their Applications 3

FSM State Equivalence

- Definition. Two states s_1 and s_2 of an FSM are equivalent, if for any sequence of input vectors, the FSM starting from state s_1 produces the same sequence of output vectors as the FSM starting from state s_2

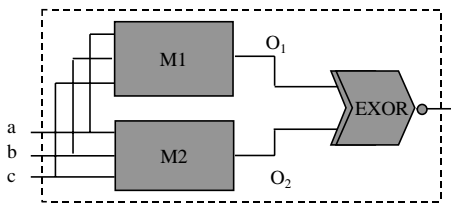
May 2, 2000

ECE 510 OCE: BDDs and Their Applications

4

Product Machine (PM)

- Given FSM $\{ I, O, S, \delta, \lambda \}$, with k inputs, n states, m output, the product machine is $\{ I, \{0,1\}, S \times S, \delta^2, \lambda^2 \}$ (the product machine has k inputs, $2n$ states, 1 output)



May 2, 2000

ECE 510 OCE: BDDs and Their Applications

5

Deriving Representation of PM

- Given transition relations and output functions of component machines $T_1(i, s_1, n_1), \{\lambda_k^1(i, s_1)\}$ and $T_2(i, s_2, n_2), \{\lambda_k^2(i, s_2)\}$, those of the PM can be computed as follows:

$$T_{PM}(i, s, n) = T_1(i, s_1, n_1) \& T_2(i, s_2, n_2)$$

$$\lambda_{PM}(i, s) = \prod_k [\lambda_k^1(i, s_1) = \lambda_k^2(i, s_2)],$$

where s_1 and s_2 are the sets of the current state variables for the component machines, and s is the union of these sets (similar for n)

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

6

FSM Equivalence Checking

- Find the transition relations and output functions of M_1 and M_2 . Find the transition relation and output function of the PM
- Perform reachability for the PM, while checking its output
- If the output of the product machine is 1 for all reachable states, M_1 and M_2 are equivalent; otherwise, generate an error trace
(It is possible to define equivalence relative to any *subset* of inputs and outputs of the FSM)

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

7

Equivalence Checking Formulas

- Property expresses equivalence of M_1 and M_2 in states s_1 and s_2 which constitute state s of PM
 $P(s) = \forall_i [\lambda_{PM}(i,s)]$
- Machines M_1 and M_2 are equivalent iff
 $\forall_s [A_R(s) \Rightarrow P(s)] = 1$
where $A_R(s)$ is the set of reachable states of the PM and $P(s)$ is the property that expresses equivalence of M_1 and M_2 in the product state s
- Alternatively, M_1 and M_2 are *not* equivalent iff
 $\exists_s [A_R(s) \& (P(s))'] = 0$

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

8

Reachability Analysis Procedure

```
bool VerifyPropertyUsingReachabilityAnalysis( FSM* pM, bdd Property )
{
  bdd initState = FindBddCube( 0, pM->NBits, CSVars, 0 );
  bdd Reached = initState, From = initState, New[MAXITERNUM];
  int NIter = 0;
  do {
    bdd To = bdd_appex(pM->TransRel, From, bddop_and, AllCSVars);
    To = bdd_replace( To, pNS4CS );
    New[ NIter ] = To - Reached;
    bdd Check = ( New[ NIter ] >> Property );
    if ( Check != bddtrue ) return false;
    From = New[ NIter ];
    Reached = Reached | New[ NIter ];
  } while ( New[ NIter++ ] != bddfals );
  return true;
}
```

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

9

FSM State Minimization

- Find the transition relations and the output functions of M . Find the transition relation and output function of the PM created by two identical instances of M
- Compute the state equivalence relation, describing the sets of all equivalent state pairs
- Compute the equivalence class characterization relation, by selecting a representative state from each class of equivalence states
- Compute the transformed transition relation and the transformed output relation

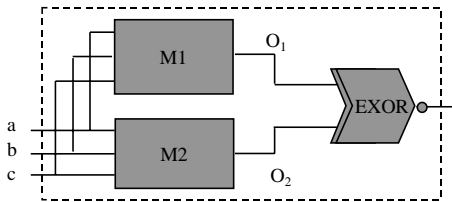
May 2, 2000

ECE 510 OCE: BDDs and Their Applications

10

Product Machine (PM)

- Given FSM $\{ I, O, S, \delta, \lambda \}$, with k inputs, n states, m output, the product machine is $\{ I, \{0,1\}, S \times S, \delta^2, \lambda^2 \}$ (the product machine has k inputs, $2n$ states, 1 output)



May 2, 2000

ECE 510 OCE: BDDs and Their Applications

11

Equivalence/Distinguishability Relations

- State equivalence relation is a boolean function $E(s_1, s_2)$, which is true for codes s_1 and s_2 iff the corresponding states are equivalent
- State distinguishability relation is a boolean function $D(s_1, s_2)$, which is true for codes s_1 and s_2 iff the corresponding states are *not* equivalent

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

12

Properties of Equivalence Relation

- Equivalence relation is reflexive, symmetric, and transitive
- Suppose the equivalence classes are $\{(00,01), (11), (10)\}$

	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	0	0	1	0
10	0	0	0	1

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

13

Computing Equivalence Relation

- $E(s_1, s_2)$ can be computed using the following procedure (iterated until $E_j(s) = E_{j+1}(s)$)
- $$E_0(s) = \forall_i [\lambda_{PM}(i,s)]$$
- $$E_{j+1}(s) = E_j(s) \ \& \ \forall_i \exists_n [T(i,s,n) \ \& \ E_j(n)]$$
- where $E_j(n) = R(s \rightarrow n)[E_j(s)]$ and $R(s \rightarrow n)$ is the variable replacement operator

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

14

Computing Distinguishability Relation

- $D(s_1, s_2)$ can be computed using the following procedure (iterated until $D_j(s) = D_{j+1}(s)$)
- $$D_0(s) = \exists_i [\lambda_{PM}(i,s)']$$
- $$D_{j+1}(s) = D_j(s) \ + \ \exists_i \exists_n [T(i,s,n) \ \& \ D_j(n)]$$
- where $D_j(n) = R(s \rightarrow n)[D_j(s)]$ and $R(s \rightarrow n)$ is the variable replacement operator

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

15

Deriving $E(s_1, s_2)$ from $D(s_1, s_2)$

- $A_R(s_1)$ is the set of reachable states
 $A_R(s_1) = \exists_{s_2} [E(s_1, s_2)]$ or
 $A_R(s_1) = \exists_{s_2} [D(s_1, s_2)]$
- The equivalence relation is derived as follows
 $E(s_1, s_2) = [D(s_1, s_2)]' \& A_R(s_1) \& A_R(s_2)$
- Similarly, for the distinguishability relation
 $D(s_1, s_2) = [E(s_1, s_2)]' \& A_R(s_1) \& A_R(s_2)$

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

16

Equivalence Class Characterization Relation

- Equivalence class characterization relation ξ selects exactly one representative from each equivalence class of states defined by $E(s_1, s_2)$
- $\xi(s_1, s_2)$ is a boolean function that is one for the codes s_1 and s_2 iff the state corresponding to s_1 represents the state corresponding to s_2 .
 $\xi(s_1, s_2) = \text{CProjection}(E(s_1, s_2), x_0)$,
where x_0 is the reset state expressed using variables s_1 .

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

17

Reduced State Set, Transition and Output Relations of State-Minimum FSM

- The state set of the minimum-state FSM
 $Q(s_1) = \exists_{s_2} [\xi(s_1, s_2)]$
- Transition relation of the state-minimum FSM
 $T_{\min}(i, s_1, n_1) = \exists_{s_2, n_2} [T(i, s_2, n_2) \& \xi(s_1, s_2) \& \xi(n_1, n_2)]$
- Output relation of the state-minimum FSM
 $O_{\min}(i, s_1, o) = \exists_{s_2} [O(i, s_2, o) \& \xi(s_1, s_2)]$

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

18

Compatible Projection Operator

- Given an equivalence relation $E(x_1, x_2) : \{0,1\}^m \times \{0,1\}^m \rightarrow \{0,1\}$, the compatible projection is a boolean function $F(x_1, x_2) = \{ (x_1, x_2) \mid (x_1, x_2) \in E, x_2 = \text{SEL}(x_1) \}$, where $\text{SEL}(x_1)$ is a selection function that uniquely selects one representative from each equivalence class

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

19

Pseudocode of Compatible Projection

```
function CProjection( E,  $\alpha$  )
  if (  $\alpha = 1$  ) return E;
  if ( E = 0 ) return 0;
  if ( E = 1 ) return  $\alpha$ ;
   $y_1$  is the top variable in  $\alpha$ ;
  if (  $\alpha_{y_1} = 0$  )       $\alpha_1 = x_1'$ ;
  else /*if (  $\alpha_{y_1} = 0$  )*/  $\alpha_1 = x_1$ ;
   $\gamma = \exists_{x_1} E_{\alpha_1}$ ;
  return  $\alpha_1$  & CProjection(  $E_{\alpha_1}$ ,  $\alpha_{\alpha_1}$  ) +
     $\gamma \alpha_1' & \text{CProjection}( E_{\alpha_1'}, \alpha_{\alpha_1'} )$ ;
```

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

20

Complete Source Code for CProjection

```
bdd CProjection( bdd F, bdd Ref )
{
  assert( Ref != bddfals );
  if ( Ref == bddtrue ) return F;
  if ( F == bddfals ) return bddfals;
  if ( F == bddtrue ) return Ref;

  // check cache for ready-made results
  bdd NextRef, Literal;
  int CurVar = bdd_var( Ref );
  if ( bdd_low( Ref ) == bddfals )
  { // the top var is positive
    NextRef = bdd_high( Ref );
    Literal = bdd_ithvar( CurVar );
  }
  else if ( bdd_high( Ref ) == bddfals )
  { // the top var is negative
    NextRef = bdd_low( Ref );
    Literal = !bdd_ithvar( CurVar );
  }
  else // Ref is not a cube!
    assert( 0 );

  // cofactors of F with respect to this literal
  bdd PosCofF = bdd_restrict( F, Literal );
  bdd NegCofF = bdd_restrict( F, !Literal );

  // the domain where projection does exist
  bdd Domain = bdd_exists( PosCofF, AllAVars0 );

  bdd PosPart = CProjection( PosCofF, NextRef );
  bdd NegPart = !Domain &
    CProjection( NegCofF, NextRef );

  bdd Result = bdd_ite( Literal, PosPart, NegPart );

  // insert the result into cache
  return Result;
}
```

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

21

Homework: A Study of Random FSMs

- Generate transition relations of random FSMs with N states and K transitions in each state
- Perform reachability analysis using the generated transition relations and determine the number of reachable states and the number of iterations in the FSM traversal
- Assume, $N = 10000$, $K = \{1, 2, \dots, 10\}$. Draw a graph visualizing the number of reachable states, the number of iterations, and the time needed to complete the reachability analysis as a function of K .

May 2, 2000

ECE 510 OCE: BDDs and Their Applications

22
