

ECE 510 OCE

BDDs and Their Applications

Lecture 14. Covering Problems and Two-Level Sum-of-Products Minimization

May 10, 2000
Alan Mishchenko

Overview

- Covering problems
 - Unate and binate covering problems
 - Branch-and-bound procedure
 - Explicit vs. implicit reduction of the covering table
- Two-level sum-of-products minimization
 - Irredundant cover computation (Minato/Morreale)
 - Explicit Quine-McCluskey procedure
 - Implicit Quine-McCluskey procedure based on formulas with quantifiers
 - Implicit minimization procedure based on transposing functions (Coudert/Madre)

Unate Covering Problem (UCP)

- Given is the covering table F , each cell of which is filled with either 0 or 1. A subset of columns $S = \{s_j\}$ is called a **solution** of UCP, if for every row there exists a column in S having 1 on the intersection.
- If the cost function is defined, $\min[\sum_j(w_j \cdot s_j)]$, then the **minimum solution** of the covering problem is a solution, for which the cost function is minimum
- UCP is solved using **branch-and-bound algorithm**. The efficiency of this algorithm, to a large degree, depends on the quality of **lower bound** and **upper bound** used to prune the search space

May 11, 2000

ECE 510 OCE: BDDs and Their Applications

3

Example of UCP

x_1	x_2	x_3	x_4	x_5	x_6
1	-	1	-	-	-
-	1	-	1	-	1
-	-	1	1	1	-
-	1	-	-	-	1
1	-	-	-	-	1
-	-	1	1	1	-

- Solution 1:
 $\{x_1, x_2, x_5\}$
- Solution 2:
 $\{x_3, x_6\}$

May 11, 2000

ECE 510 OCE: BDDs and Their Applications

4

Branch-and-Bound Algorithm

```
BCP ( F, U, currentSol ) {
  (F, currentSol) = REDUCE(F, currentSol);
  if ( terminalCase(F) {
    if ( COST( currentSol < U )
      { U = COST( currentSol ); return ( currentSol ) }
    else return ("No solution");
  L = LOWER_BOUND( F, currentSol );
  if ( L ≥ U ) return ("No solution");
  xi = CHOOSE_VAR( F );
  S1 = BCP( Fxi, U, currentSol ∪ {xi} );
  if ( COST(S1) = L ) return S1;
  S2 = BCP( Fxi', U, currentSol );
  return BEST_SOLUTION( S1, S2 ); }
```

May 11, 2000

ECE 510 OCE: BDDs and Their Applications

5

Reduction of Covering Table

```
reduce( R,C,U ) {
  do { CollapseColumns( C );
    RemoveDominatedColumns( R,C );
    Solution = Solution ∪ EssentialColumns( R,C );
    if ( |Sol| ≥ U ) return ∅;
    CollapseRows(R);
    RemoveDominatedRows( R,C );
  } while ( either R or C has changed );
  return ( R,C ); }
```

May 11, 2000

ECE 510 OCE: BDDs and Their Applications

6

Binare Covering Problem (BCP)

- Given the covering table F and the cost function $\min[\sum_j (w_j x_j)]$, BCP is formulated as follows
- Find a subset of columns $S = \{s_i\}$ of minimum cost such that for every row R_j either (1) $\exists_j: (F_{ij}=1), s_j \in S$ or (2) $\exists_j: (F_{ij}=0), s_j \notin S$
- BCP arises in incompletely-specified FSM state minimization, technology mapping, decomposition of functions with DCs, BDD minimization, etc.
- BCP is reduced to the problem of finding the shortest path on the BDD representing the product of all the constraints

May 11, 2000

ECE 510 OCE: BDDs and Their Applications

7

Example of BCP

x_1	x_2	x_3	x_4	x_5	x_6
1	-	1	-	-	-
-	1	-	1	-	1
-	-	0	1	1	-
-	-	-	-	-	0
0	-	-	-	-	0
-	-	1	0	1	-

- Solution 1:
 $\{x_1, x_2\}$
- Solution 2:
 $\{x_3, x_4\}$

May 11, 2000

ECE 510 OCE: BDDs and Their Applications

8

Reduction of UCP/BCP to Satisfiability

- Create the product of constraints for each column (see previous example)
$$F(x) = (x_1 + x_3) \& (x_2 + x_4 + x_6) \& (x'_3 + x_4 + x_5) \& x'_6$$
$$\& (x'_1 + x'_6) \& (x_3 + x'_4 + x_5)$$
- Find the shortest path in the BDD for $F(x)$

May 11, 2000

ECE 510 OCE: BDDs and Their Applications

9

Implicit Formulation of UCP/BCP

- If there are many columns (>100), the satisfiability approach does not work well
- The alternative method is:
 - (1) logarithmically encode columns and rows using two sets of variables c and r ,
 - (2) represent the covering table as a relation $U(c,r)$,
 - (3) apply the branch-and-bound procedure with implicit covering table reduction

May 11, 2000

ECE 510 OCE: BDDs and Their Applications

10

Example of Implicit Reduction Step: Finding/Collapsing Duplicated Columns

Duplicated columns can be computed as

$$\text{dup_col}(c_1, c_2) = \forall r [R(r) \Rightarrow (1(r, c_1) \leftrightarrow 1(r, c_2))]$$

Duplicated columns can be collapsed by

$$C(c_1) = C(c_1) \& \exists c_2 [C(c_2) \& (c_2 < c_1) \& \text{dup_col}(c_1, c_2)],$$

where $(c < c')$ is the relation which is true for c and c' if the binary code of c is less than that of c'

$$(x < y) = x'_1 \& y_1 \mid x'_1 \& y'_1 \& x'_2 \& y_2 \mid x'_1 \& y'_1 \& x'_2 \& y'_2 \& x'_3 \& y_3 \mid \dots$$

Two-Level SOP Minimization

- **Approximate minimization**
 - Irredundant cover computation (Minato/Morreale algorithm)
- **Exact minimization**
 - Explicit Quine-McCluskey procedure
 - Implicit Quine-McCluskey procedure based on formulas with quantifiers
 - Minimization procedure based on transposing functions (Coudert/Madre algorithm)

Implicit Cube Representation

- To represent cubes of n -variable function $F(x_1, x_2, \dots, x_n)$, two sets of n vars are used:
signature variables $S = (s_1, s_2, \dots, s_n)$
and polarity variables $P = (p_1, p_2, \dots, p_n)$
 s_i variable is true iff i -th variable is present in the cube; p_i variable is true iff i -th variable enters this cube in positive polarity
- For example, cube $x_2x_3'x_4$ is represented by the pair $[(0111), (-101)]$

May 11, 2000

ECE 510 OCE: BDDs and Their Applications

13

Observations

- All cube sets in two-level minimization algorithms are represented using characteristic functions depending on signature and polarity variables
- Graph representations of cube sets using Zero-Suppressed BDDs have 2-5 times less nodes than classical BDDs

May 11, 2000

ECE 510 OCE: BDDs and Their Applications

14

Irredundant Cover Computation

- The main idea of irredundant cover computation is to divide the cover into three part
 - the one that has literal x_k in negative polarity
 - the one that has literal x_k in positive polarity
 - the one that does not have literal x_k at all
- These three sets are computed independently by calling irredundant cover recursively for problems of smaller size. Finally, the resulting cover is assembled from the parts

May 11, 2000

ECE 510 OCE: BDDs and Their Applications

15

Complete Source Code for IrrCover()

```
bdd IrrCover( bdd F1, bdd F12, int k )
{
  if ( F1 == bddfalses ) return bddfalses;
  if ( F12 == bddtrue ) return AllS0VarsZeros[k];
  // check cache for results
  bdd F1_low = bdd_restrict( F1, !P0Vars[k] );
  bdd F1_high = bdd_restrict( F1, P0Vars[k] );
  bdd F12_low = bdd_restrict( F12, !P0Vars[k] );
  bdd F12_high = bdd_restrict( F12, P0Vars[k] );
  bdd g0 = F1_low & F12_high;
  bdd P0 = IrrCover( g0, F12_low, k+1 );
  bdd g1 = F1_high & F12_low;
  bdd P1 = IrrCover( g1, F12_high, k+1 );
  bdd c0 = bdd_exist( P0, S0Vars );
  bdd c1 = bdd_exist( P1, S0Vars );
  bdd h1 = ( F1_low & !c0 ) | ( F1_high & !c1 );
  bdd h12 = F12_low & F12_high;
  bdd P = IrrCover( h1, h12, k+1 );
  bdd Cover = (!S0Vars[k] & P) | (S0Vars[k] & !P0Vars[k] & P0) | (S0Vars[k] & P0Vars[k] & P1);
  // insert the result into cache
  return Cover;
}
```

May 11, 2000

ECE 510 OCE: BDDs and Their Applications

16

Coudert/Madre Minimization Procedure

- Recursively build implicit representations of all primes and all minterms
- Iteratively transform cube sets by applying transposing functions; remove essential cubes after each iteration
- Solve the resulting cyclic core (if any) using explicit branch-and-bound techniques

May 11, 2000

ECE 510 OCE: BDDs and Their Applications

17

Source Code for **BuildPrimeCubeSet()**

```
bdd BuildPrimeCubeSet( bdd F, int k )
{
  if ( F == bddfals )      return bddfals;
  if ( F == bddtrue )     return g_AllSVarsZeros[k];
  // check cache for results
  bdd F_low = bdd_restrict( F, !g_XVars[k] );
  bdd F_high = bdd_restrict( F, g_XVars[k] );
  bdd P = BuildPrimeCubeSet( F_low & F_high, k+1 );
  bdd P0 = BuildPrimeCubeSet( F_low, k+1 ) - P;
  bdd P1 = BuildPrimeCubeSet( F_high, k+1 ) - P;
  bdd Setn = !g_SVars[k] & P;
  bdd Set0 = g_SVars[k] & !g_PVars[k] & P0;
  bdd Set1 = g_SVars[k] & g_PVars[k] & P1;
  bdd Result = Setn | Set0 | Set1;
  // insert the result into cache
  return Result;
}
```

May 11, 2000

ECE 510 OCE: BDDs and Their Applications

18