

ECE 510 OCE BDDs and Their Applications

Lecture 9. Programming with BDDs

April 25, 2000
Alan Mishchenko

Overview

- Mapping elementary BDD variables
- Building transition/output relation from STG
- BDD operators for image computation
- Programming reachability analysis procedures
- Programming error trace generation

Mapping Elementary Bdd Variables

```
// first, map the input variables
for ( int i = 0; i < INPUTVARRANGE; i++ )
    g_InVars[i] = bdd_ithvar(i);
// next, the current state and next state variables
for ( i = 0; i < STATEVARRANGE; i++ ) {
    g_CSVars[i] = bdd_ithvar( INPUTVARRANGE + 2*i );
    g_NSVars[i] = bdd_ithvar( INPUTVARRANGE + 2*i + 1 );
}
// finally, the output variables
for ( i = 0; i < OUTPUTVARRANGE; i++ )
    g_OutVars[i] =
        bdd_ithvar( INPUTVARRANGE + 2*STATEVARRANGE + i );
```

April 25, 2000

ECE 510 OCE: BDDs and Their Applications

3

FSM Data Structure

```
typedef struct FSM_tag {
    int NInputs;    // number of inputs
    int NOutputs;  // number of outputs
    int NStates;   // number of states
    int NLines;    // number of lines in KISS table
    int NBits;     // number of bits (flip-flops)
    string StateNames[MAXSTATENUM]; // symbolic state names
    bdd TransRel; // the transition relation
    bdd OutRel;   // the output relation
} FSM;
```

April 25, 2000

ECE 510 OCE: BDDs and Their Applications

4

Building Transition/Output Relations (1)

```
// iteratively create the transition and output relation
for ( i = 0; i < pM->NLines; i++ )
{
    Input >> InBuf >> CSBuf >> NSBuf >> OutBuf;
    // check whether the length of InBuf and OutBuf are correct
    bdd InputCube = bddtrue;
    for ( int k = 0; k < pM->NInputs; k++ )
        if ( InBuf[k] == '1' )
            InputCube &= InVars[k];
        else if ( InBuf[k] == '0' )
            InputCube &= !InVars[k];
        else if ( InBuf[k] != '-' )
            Error( "The input cube contains a wrong symbol");
}
```

April 25, 2000

ECE 510 OCE: BDDs and Their Applications

5

Building Transition/Output Relations (2)

```
int CSNum = FindStateNumber( pM->StateNames, string(CSBuf));
bdd CSCube = FindBddCube( CSNum, pM->NBits, CSVars, 0 );
int NSNum = FindStateNumber( pM->StateNames, string(NSBuf));
bdd NSCube = FindBddCube( NSNum, pM->NBits, NSVars, 0 );

// compute the output cube, similar to the input cube

// add this transition to the transition and output relations
pM->TransRel |= InputCube & CSCube & NSCube;
pM->OutRel |= InputCube & CSCube & OutputCube;
}
```

April 25, 2000

ECE 510 OCE: BDDs and Their Applications

6

Function FindBddCube()

```
bdd FindBddCube( int Value, int CodeWidth, bdd Vars[], int Offset )
// returns a bdd composed of elementary bdds found in array Vars[],
// such that the number Value is encoded with CodeWidth variables
// (most significant bit is encoded with the first bdd variable)
{ bdd Result = bddtrue;
  for ( int z = 0; z < CodeWidth; z++ )
    if ( Value & ( 1 << (CodeWidth-1-z) ) )
      Result &= Vars[ Offset+z ];
    else
      Result &= !Vars[ Offset+z ];
  return Result;
}
```

April 25, 2000

ECE 510 OCE: BDDs and Their Applications

7

BDD Operators for Image Computation

- To have an existential quantifier computed for the function **F** with variables **Vars**, call function `bdd bdd_exist(bdd F, bdd Vars);`
- For efficient computation of existential quantifier and certain boolean operation, call the function `bdd bdd_appex(bdd F, bdd G, int OpCode, bdd Vars);`
- To replace variables from set **V1** to set **V2**, create a replacement pair and call the function `bdd bdd_replace(bdd F, bddPair* pReplaceV1forV2);`

April 25, 2000

ECE 510 OCE: BDDs and Their Applications

8

Defining Replacement Pair

```
int iCSVars[ STATEVARRANGE ]; // int nums of cur state vars
int iNSVars[ STATEVARRANGE ]; // int nums of the next state vars

for ( int j = 0; j < pM->NBits; j++ ) {
    iCSVars[j] = INPUTVARRANGE + 2*j;
    iNSVars[j] = INPUTVARRANGE + 2*j + 1;
}
// set up the replacement pair
bddPair *pNSforCS = bdd_newpair();
if ( bdd_setpairs( pNSforCS, iNSVars, iCSVars, pM->NBits ) )
    Error( "Cannot create the replacement pair" );
```

April 25, 2000

ECE 510 OCE: BDDs and Their Applications

9

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->TRel, From, bddop_and, AllCSVars);
        To = bdd_replace( To, pNS4CS );
        New[ NIter ] = To - Reached;
        bdd Check = bdd_apply( New[ NIter ], Property, bddop_imp );
        if ( Check != bddtrue ) return false;
        From = New[ NIter ];
        Reached = Reached | New[ NIter ];
    } while ( New[ NIter++ ] != bddfals );
    return true;
}
```

April 25, 2000

ECE 510 OCE: BDDs and Their Applications

10

Generating Error Trace

```
if ( Check != bddtrue ) {
    bdd TraceCube = bdd_satone( !Check );
    for ( int k = NIter; k >=0; k-- ) {
        unsigned int State = 0; // the state in the error cube
        for ( int p = 0; p < pM->NBits; p++ )
            if ( (TraceCube & CSVars[p]) != bddfals )
                State |= ( 1 << ( pM->NBits-1-p ) );
        cout << "The error state is: " << pM->StateNames[ State ];
        // compute the backward image of this cube
        bdd ShiftedCube = bdd_replace( TraceCube, pCS4NS );
        bdd BackImage = bdd_appex( pM->TransRel, ShiftedCube,
                                   bddop_and, AllNSVars );
        bdd Overlap = BackImage & New[ k-1 ];
        TraceCube = bdd_satone( Overlap );    }    return; }
```

April 25, 2000

ECE 510 OCE: BDDs and Their Applications

11