

ECE 510 OCE
BDDs and Their Applications

Lecture 20.
Sequential Verification

June 1, 2000
Alan Mishchenko

Overview

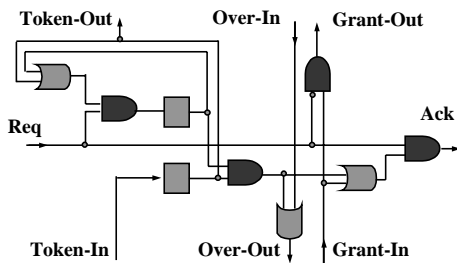
- Examples of formal verification
- Verifiable properties
- Temporal logics and CTL
- Implementation of CTL model checking
- Finite automata and L-automata
- ω -regular languages
- Formulation of verification problem using process and property automata
- Language containment algorithm

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

2

Synchronous Arbiter Cell

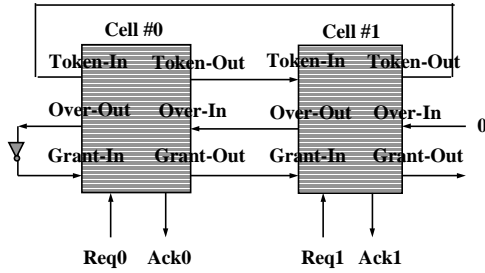


June 1, 2000

ECE 510 OCE: BDDs and Their Applications

3

Interconnection of Two Arbiter Cells



June 1, 2000

ECE 510 OCE: BDDs and Their Applications

4

Safety Property

- A safety property expresses the condition that something bad *should not* happen
- Example: mutual exclusion, "only one (out of two or more) signals can be asserted at the same time"

$$\text{AG} ((\text{Ack0}=1 \rightarrow \text{Ack1}=0) * (\text{Ack1}=1 \rightarrow \text{Ack0}=0))$$

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

5

Liveness Property

- A liveness property expresses the condition that something good *should* happen
- Example: "from every state that is reachable from the initial states, we must eventually encounter a state in which either there is no request, or there is an acknowledgement"

$$\text{AG} (\text{AF} (\text{Req0}=1 \rightarrow \text{Ack0}=1))$$

$$\text{AG} (\text{AF} (\text{Req1}=1 \rightarrow \text{Ack1}=1))$$

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

6

Fairness Condition

- Fairness condition (constraint), expressed as a temporal logic formula, allows verification to concentrate on "interesting cases", i.e those computation path where the system is "active"
- A computation path satisfies a fairness condition is the formula specifying the condition is true infinitely often on this path
- Example: "at least one of the arbiters receives request signals"

$$\text{Req0} = 1 + \text{Req1} = 1$$

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

7

Temporal Logic Operators (X,F,G,U)

- neXt: Xp is true if p is true at the next moment
- Future: Fp is true if p is true at some moment in the future
- Global: Gp is true if p is true always in the future
- Until: pUq is true if q holds at some time in the future, until which time p holds

Temporal operators are reducible to each other

Examples: $Gp = (F(p'))'$

$$Xp = \text{false} U p$$

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

8

Computation Path Quantifiers (A,E)

- A - universal (for all path of the model)
- E - existential (there exists such path)

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

9

Kripke model and CTL

- Kripke model (sequential system) is a set of states connected by transitions and a relation of temporal order ($<$) defined on states
- A model, by definition, provides valuation L , which assigns truth/falsehood to every atomic proposition (output variable) in every state
- Temporal order defines a computational tree which branches towards the future
- Temporal logic formulas defined over such a tree constitute Computational Tree Logic (CTL)

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

10

History

- A subset of temporal logic was defined by Clarke and Emerson (1981), which allows for efficient computation whether a formula is satisfied in a given state of a given finite model
- A BDD based implementation of CTL model checking with fairness conditions was proposed by Burch, Clarke, and McMillan (1990)

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

11

CTL formulas

- An atomic proposition is a CTL formula
- If f and g are CTL formulas, then f' , $f \& g$, $f + g$, EXf , $E(fUg)$, EGf , are also CTL formulas
- Other operators can be derived
$$EFf = E(f+f')Ug$$
$$AXf = (EX f)'$$
$$AGf = (EF f)'$$
$$A f f = (EG f)'$$
$$A f U g = (E f' U (f + g))' \& (EG g)'$$

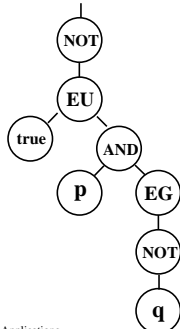
June 1, 2000

ECE 510 OCE: BDDs and Their Applications

12

Example of CTL Formula Evaluation

- Formula is $AG(p \rightarrow AFq)$
- Transform the formula
 $AG(p \rightarrow AFq) =$
 $(EF [p' + (EGq)'])' =$
 $(E \text{ true } U [p \ \& \ EGq'])'$
- Build the parse tree
- Recursively evaluate sets of states that satisfy the condition of each node
- Return the root value



June 1, 2000

ECE 510 OCE: BDDs and Their Applications

13

Symbolic Model Checking Theorem

- Theorem (Clarke-Emerson): For a Kripke model (S, S_0, R, A, L) , where S is the set of states, S_0 is the set of initial states, R is the transition relation, A is the set of atomic propositions (outputs), and L is the output function, which says what outputs are 1 in each state, the follows is true:
 - $EFp = \mu y. (p + EXy)$
 - $EGp = \nu y. (p \ \& \ EXy)$
 - $E(q \ U \ p) = \mu y. (p + (q \ \& \ EXy))$

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

14

Symbolic Model Checking Algorithm

```

function EVAL( f )
{
  case
  f is an atomic proposition: return f
  f = p': return [ EVAL(p) ]'
  f = p & q: return EVAL(p) & EVAL(q)
  f = EXp: return EVAL_EX( EVAL(p) )
  f = E(pUq): return EVAL_EU(EVAL(p), EVAL(q),false)
  f = EGp: return EVAL_GP(EVAL(p), true )
}
    
```

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

15

Specialized Procedures

```
bdd EVAL_EX(p) { return  $\exists s'(R(s,s') \ \& \ p(s'))$  }
```

```
bdd EVAL_EU(p,q,y) {  
  y' = q + ( p & EVAL_EX(y) )  
  if ( y'=y ) return y  
  else return EVAL_EU( p,q,y' ) }
```

```
bdd EVAL_EG(p,y) {  
  y' = ( p & EVAL_EX(y) )  
  if ( y'=y ) return y  
  else return EVAL_EG( p,y' ) }
```

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

16

Finite Automata (String Acceptors)

- Definition. A finite automaton A is a transition structure with an acceptance set C: $A = \langle T, C \rangle = \langle X, S, \delta, S_0, C \rangle$
- Let input string be x and corresponding run s, where $x = (x_1, x_2, \dots, x_n)$, $x_i \in X$, $0 \leq i \leq n$. Given a starting state $s_0 \in S_0$, the string x produces run $s^x = (s_0^x, s_1^x, \dots, s_n^x)$, where $s_i^x \in \delta(s_{i-1}, x) \subseteq S$. The input string x is accepted iff $s_n^x \in C$.

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

17

L-Automata (Infinite String Acceptors)

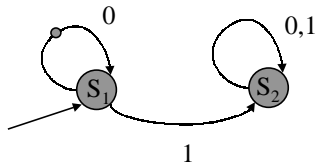
- Definition. An L-automaton A is a state transition structure with edge acceptance set E^A and node set C^A
 $A = \langle T, E^A, C^A \rangle = \langle X, S, \delta, S_0, E^A, C^A \rangle$
- The string x is accepted iff either the run s^x has an infinite number of STG edges belonging to the set of recur edges E^A , or there exist a cycle set $C_i \in C^A$ such that $S^\infty(s^x) \subseteq C_i$, where $S^\infty(s^x)$ is the set of states that occur infinitely often in the run s^x .

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

18

Example of L-automaton



This automaton expresses the complement of the liveness property: "If I wait at this red light long enough, it will eventually turn green"

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

19

Language Containment Algorithm

- Find the product of the property automaton and the process automaton
- Compute the set of reachable states $R(x)$
- Remove the recur edges and restrict the transition relation to reachable states
 $T^-(x,y) = T(x,0,y) \ \& \ R(x)$
- Compute the transitive closure T^{-*} of T^- .
- For all cycle sets C_i , compute the set of cycles of T^{-*} not contained in C_i
 $NC_i(x) = \exists_y [T^{-*}(x,y) \ \& \ T^{-*}(y,x) \ \& \ C_i^c(y)]$
- The language containment check succeeds iff the intersection of all these cycles is empty

June 1, 2000

ECE 510 OCE: BDDs and Their Applications

20

$$\bigcap NC_i(x) = 0$$
