Problem Solving in Computer Science

Scribe notes by Salman Mirghasemi

06.11.2008

1 BDD Operations

Suppose that b_0 and b_1 are the two BDDs shown in figure 1.

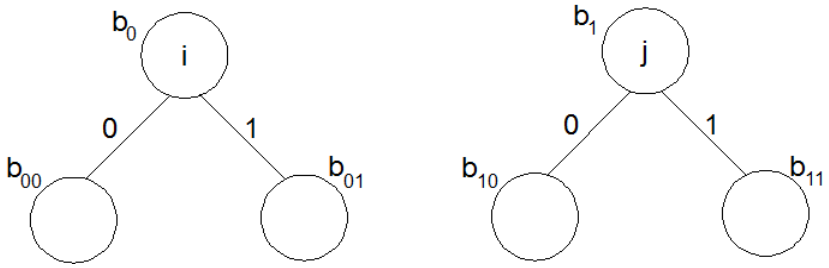

Figure 1: Two binary decision diagrams b_0 and b_1

$$\mathrm{b}_0 = (\neg x_i \wedge b_{00}) \vee (x_i \wedge b_{01})$$
$$b_1 = (\neg x_j \wedge b_{10}) \vee (x_j \wedge b_{11})$$

1. Conjunction

$$\mathrm{i} < \mathrm{j} : \mathrm{b}_0 \wedge b_1 = (\neg x_i \wedge (b_{00} \wedge b_1)) \vee (x_i \wedge (b_{01} \wedge b_1))$$
$$i = j : b_0 \wedge b_1 = (\neg x_i \wedge (b_{00} \wedge b_{10})) \vee (x_i \wedge (b_{01} \wedge b_{11}))$$

2. Disjunction

$$\mathrm{i} < \mathrm{j} : \mathrm{b}_0 \vee b_1 = (\neg x_i \wedge (b_{00} \vee b_1)) \vee (x_i \wedge (b_{01} \vee b_1))$$
$$i = j : b_0 \vee b_1 = (\neg x_i \wedge (b_{00} \vee b_{10})) \vee (x_i \wedge (b_{01} \vee b_{11}))$$

3. Negation

$$\neg b_0 = (\neg x_i \wedge \neg b_{00}) \vee (x_i \wedge \neg b_{01})$$

2 Concurret BDD computation

The BDD pool is in the shared memory and concurrent threads make a stream of MakeBDD calls to that shared structure.

T_1	T_2	T_3
MakeBDD(b_{11})	MakeBDD(b_{21})	MakeBDD(b_{13})
MakeBDD(b_{12})	MakeBDD(b_{13})	MakeBDD(b_{22})
$\vdots$	$\vdots$	$\vdots$

In concurrent operations two properties should be considered : safety and liveness.

1. Safety
   No call to MakeBDD returns a "wrong" answer.

   Linearizability: There is a sequential interleaving of all makeBDD calls such that the return values of all concurrent calls are exactly those that would be obtained from sequentially executing all calls in the order of the interleavings.

2. Liveness

   Wait Freedom: Each method call return within a finite number of steps.

   Lock Freedom: There is always some method call that returns in linear number of steps (guarantees overall progress but not progress on each thread).

A good synchronization solution locks as little data and as short time is possible.
You can find examples for Course-Grained and Fine-Grained synchronization in this book : *"The Art of Multiprocessor Programming, Written by H.Maurice and N.Shavit"*.