CS 243 - Section 7

Winter 2021

March 5, 2021

1 Pointer Analysis

Consider the following program:

```
class A {
    public A f(A c) {
        if (c != null) {
            c = new A(); // h1
        }
        return c;
    }
    public static void main(String[] args) {
        A a = new A(); // h2
        A b = new A(); // h3
        a = f(a);
        b = f(b);
    }
}
```

1. Perform context-sensitive, flow-insensitive analysis. What are pts tuples inferred from the code?

   With a context-sensitive analysis, the two calls to f will have their own contexts; call the parameter to the first call to f, c1 and the parameter to the second call to f, c2. Call the allocation in f (now separate), h1a and h1b.

   pts(a, h2)
   pts(b, h3)
   pts(c1, h2)
   pts(c1, h1a)
   pts(a, h1a)
   pts(c2, h3)
   pts(c2, h1b)

pts(b, h1b)

2. Now perform context-insensitive, flow-insensitive analysis. What are the pts tuples inferred from the code?

   Now the calls to f are not separate.

   pts(a, h2)
   pts(b, h3)
   pts(c, h2)
   pts(c, h1)
   pts(a, h1)
   pts(c, h3)
   pts(b, h1)

   These below are an artifact of the context-insensitivity (the above is very similar to the context-sensitive analysis, but the fact that the c's are not separate means that the parameters across the calls alias each other [even though at runtime they do not]).

   pts(b, h2)
   pts(a, h3)

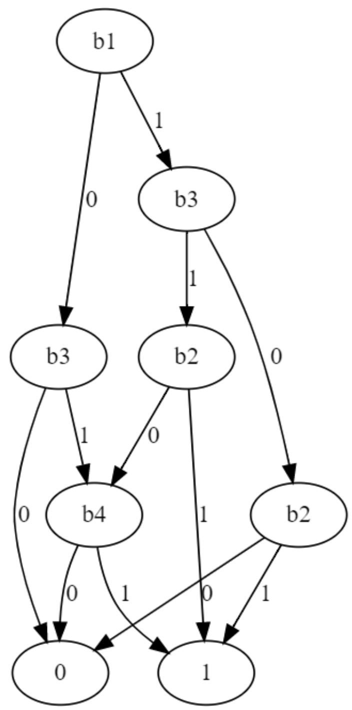
3. What are the true set of pts tuples inferred from the code (for a and b)?

   pts(a, h2)
   pts(b, h3)
   pts(a, h1a)
   pts(b, h1b)

2 BDD Variable Ordering

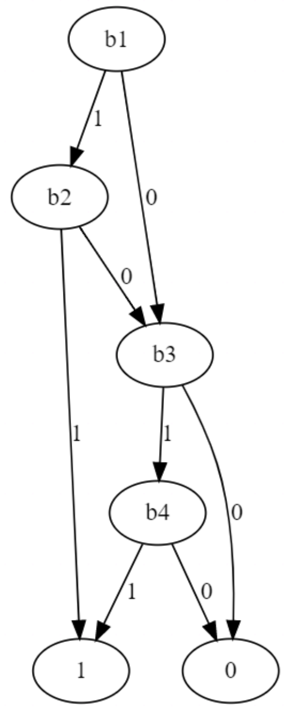
1. Consider the following boolean expression: `(b1 ∧ b2) ∨ (b3 ∧ b4)`. What is the worst-case ordering (in terms of number of nodes needed)? Show the BDD for this ordering.

   b1 $\geq$ b3 $\geq$ b2 $\geq$ b4.



2. What is the optimal variable ordering (in terms of number of nodes used)? Show the BDD for this ordering as well.

   b1 $\geq$ b2 $\geq$ b3 $\geq$ b4.



3. Now let's generalize; consider the boolean expression $(b_1 \wedge b_2) \vee (b_3 \wedge b_4) \vee \cdots \vee (b_{2n-1} \wedge b_{2n})$. What do you think the optimal ordering will be and how many nodes will it take (in terms of n)? What about for the worst-case ordering? (No need for a proof, just your rough intuition should suffice).

   It turns out that the variable ordering and relationship between the number of nodes and n generalizes beyond the specific case we've worked through here.

   The optimal ordering will be $b_1 \geq b_2 \geq \cdots \geq b_{2n-1} \geq b_{2n}$. If we look above, we can see that for $n = 2$, we had 6 nodes, or $2n + 2$ nodes (one node for each variable and 2 for the sink nodes).

   The pessimal ordering (at least one of them) will be $b_1 \geq b_3 \geq \cdots \geq b_{2n-1} \geq b_2 \geq b_4 \geq \cdots \geq b_{2n}$. This is a little harder to see (and even harder to prove), but you will require an exponential number of nodes; 2^{n+1} to be specific (we can see this for the $n = 2$ case since above, we see we required $2^3 = 8$ nodes).