15-213

"The course that gives CMU its Zip!"

Verifying Programs with BDDs
Sept. 22, 2006

Topics

- Representing Boolean functions with Binary Decision Diagrams
- Application to program verification

Verification Example

```
int abs(int x) {
  int mask = x>>31;
  return (x ^ mask) + ~mask + 1;
}
```

```
int test_abs(int x) {
  return (x < 0) ? -x : x;
}
```

Do these functions produce identical results?

How could you find out?

How about exhaustive testing?

More Examples

```
int addXY(int x, int y)
{
  return x+y;
}
```

?

=

```
int addYX(int x, int y)
{
  return y+x;
}
```

```
int mulXY(int x, int y)
{
  return x*y;
}
```

?

=

```
int mulYX(int x, int y)
{
  return y*x;
}
```

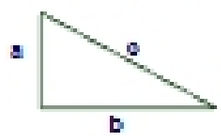

How Can We Verify Programs?

Testing

- Exhaustive testing not generally feasible
- Currently, programs only tested over small fraction of possible cases

Formal Verification

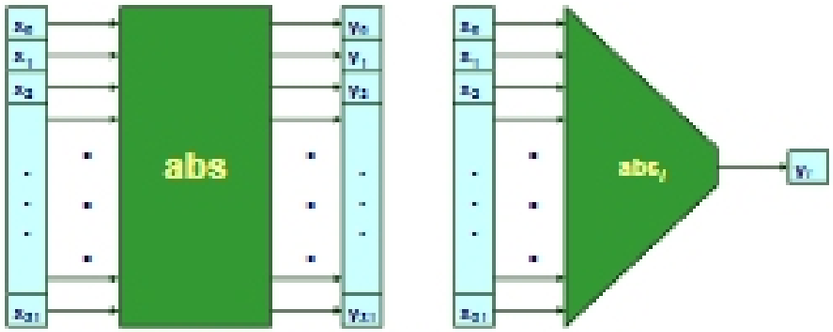
- Mathematical "proof" that code is correct



- Did Pythagoras show that $a^2 + b^2 = c^2$ by testing?

Bit-Level Program Verification

```
int abs(int x) {
  int mask = x>>31;
  return (x ^ mask) + ~mask + 1;
}
```



- View computer word as 32 separate bit values
- Each output becomes Boolean function of Inputs

Extracting Boolean Representation

```
int bitOr(int x, int y)
{
  return ~(~x & ~y);
}
```

```
int test_bitOr(int x, int y)
{
  return x | y;
}
```

Do these functions produce identical results?

Straight-Line Evaluation

x	
y	
v1 =	~x
v2 =	~y
v3 =	v1 & v2
v4 =	~v3
v5 =	x | y
t =	v4 == v5

Tabular Function Representation

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

- List every possible function value

Complexity

- Function with *n* variables

Algebraic Function Representation

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$x_2 \cdot x_3$

$x_1 \cdot x_3$

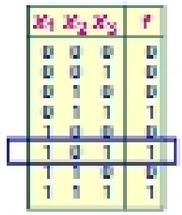
- $f(x_1, x_2, x_3) = (x_1 + x_2) \cdot x_3$
- Boolean Algebra

Complexity

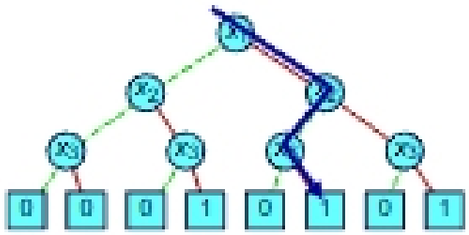
- Representation
- Determining properties of function
  - E.g., deciding whether two expressions are equivalent

Tree Representation

Truth Table



Decision Tree

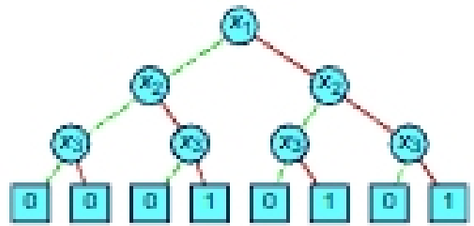


- Vertex represents decision
- Follow green (dashed) line for value 0
- Follow red (solid) line for value 1
- Function value determined by leaf value

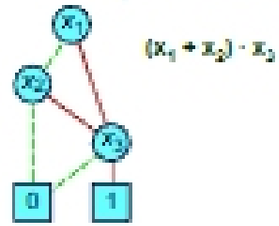
Complexity

Ordered Binary Decision Diagrams

Initial Tree



Reduced Graph



Canonical representation of Boolean function

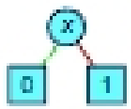
- Two functions equivalent if and only if graphs isomorphic
  - Can be tested in linear time
- Desirable property: *simplest form is canonical.*

Example Functions

Constants

- 0 Unique unsatisfiable function
- 1 Unique tautology

Variable

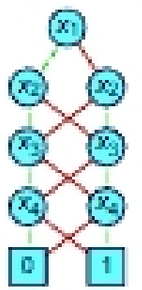


Treat variable as function

Typical Function

- $(x_1 + x_2) \cdot x_4$
- No vertex labeled x_3
  - Independent of x_3
- Many subgraphs shared

Odd Parity



Linear representation

More Complex Functions

Functions

- Add 4-bit words a and b
- Get 4-bit sum s
- Carry output bit Cout

Shared Representation

- Graph with multiple roots
- 31 nodes for 4-bit adder
- 571 nodes for 64-bit adder
- Linear growth!