Model Counting with Applications to CodeHunt

Willem Visser

Stellenbosch University

South Africa

CodeHunt is built on

SAT or UNSAT?
And
Some solutions

Model Counting

SAT solutions?

Can we use this
for a Progress Bar?

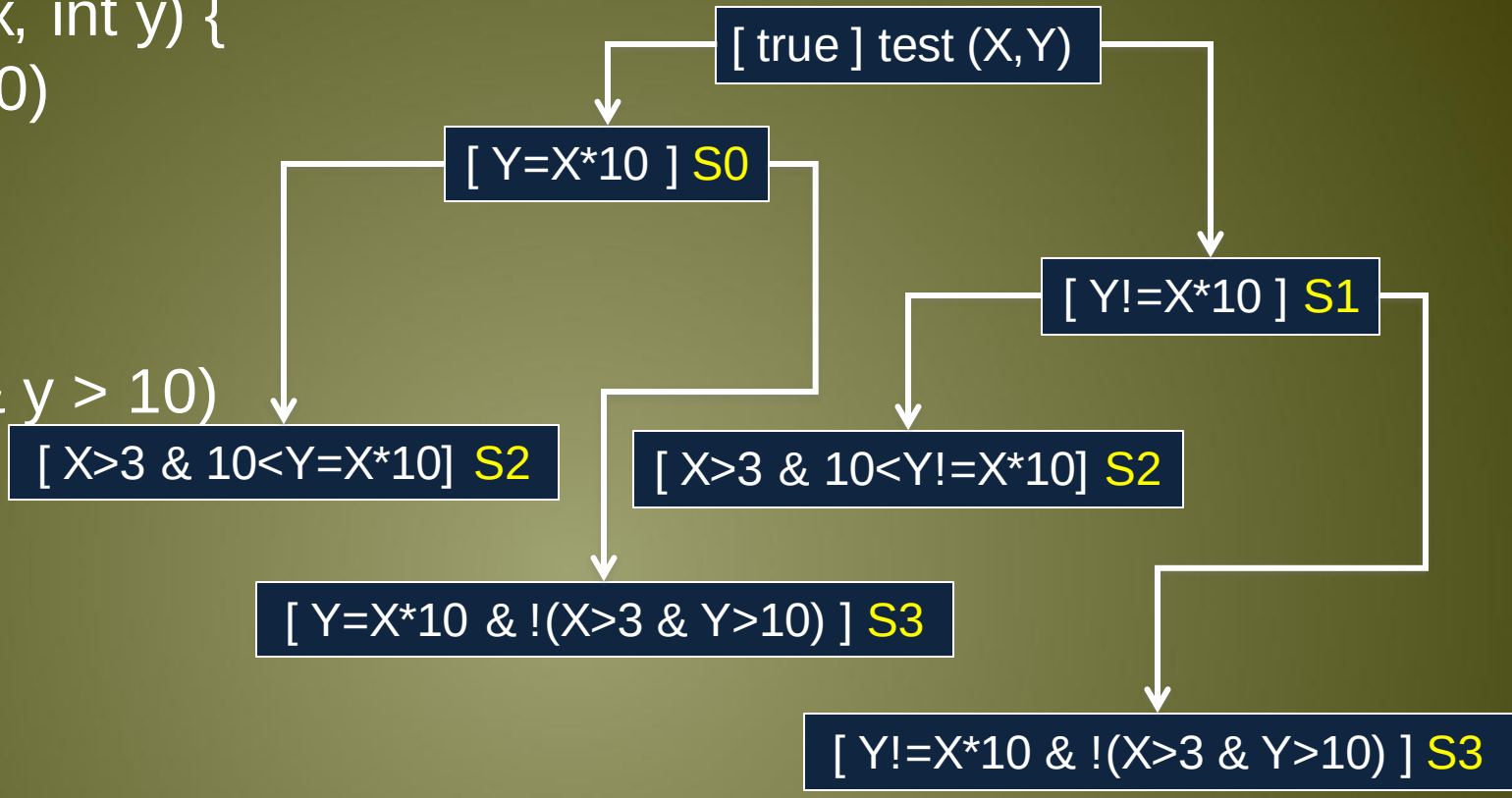
In a perfect world…

only linear integer constraints

and only uniform distributions

Symbolic Execution

```
void test(int x, int y) {
  if (y == x*10)
    S0;
  else
    S1;
  if (x > 3 && y > 10)
    S2;
  else
    S3;
}
```



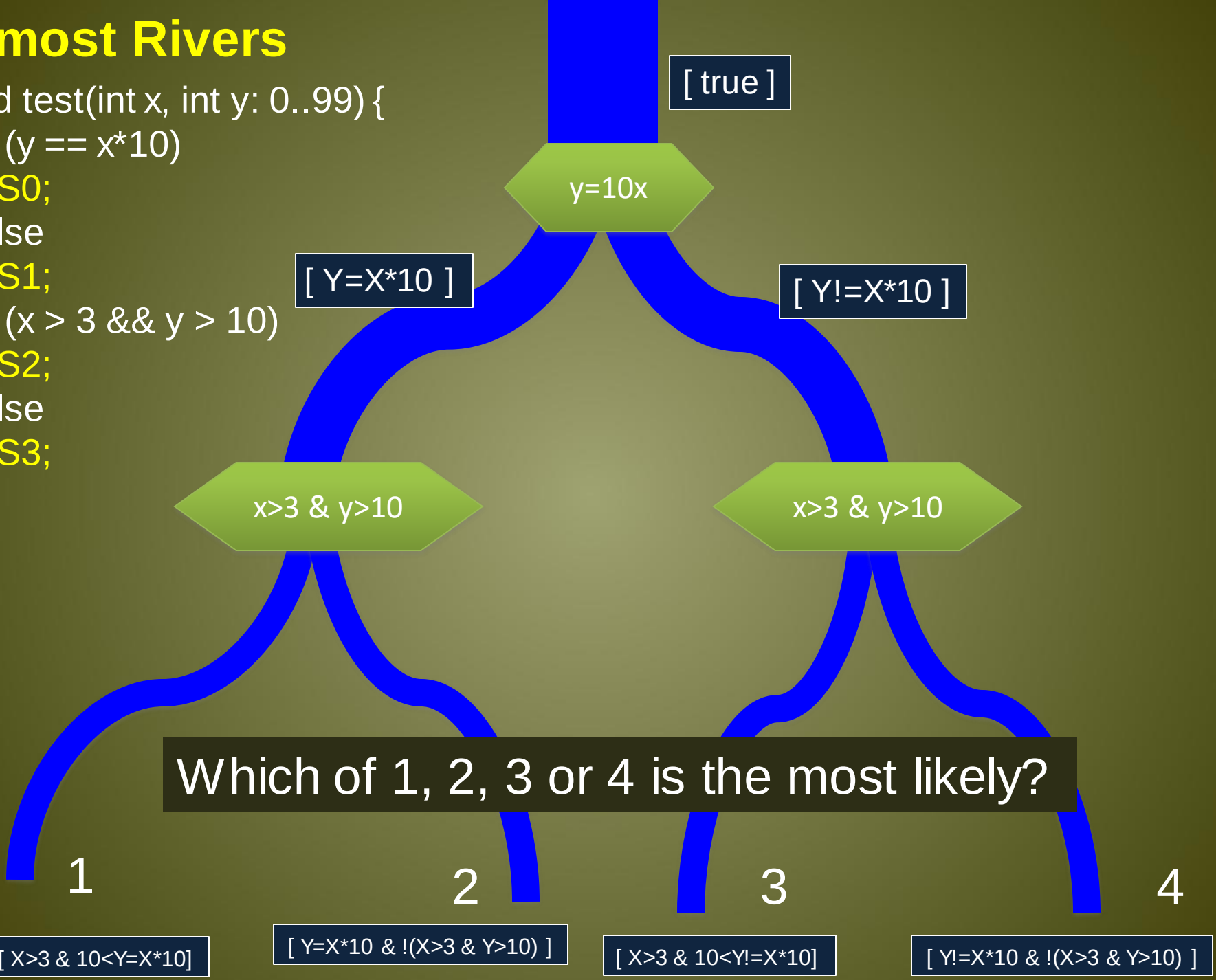
Test(1,10) reaches S0,S3
Test(0,1) reaches S1,S3
Test(4,11) reaches S1,S2

Photograph by Annie Griffiths Belt

NATIONAL GEOGRAPHIC

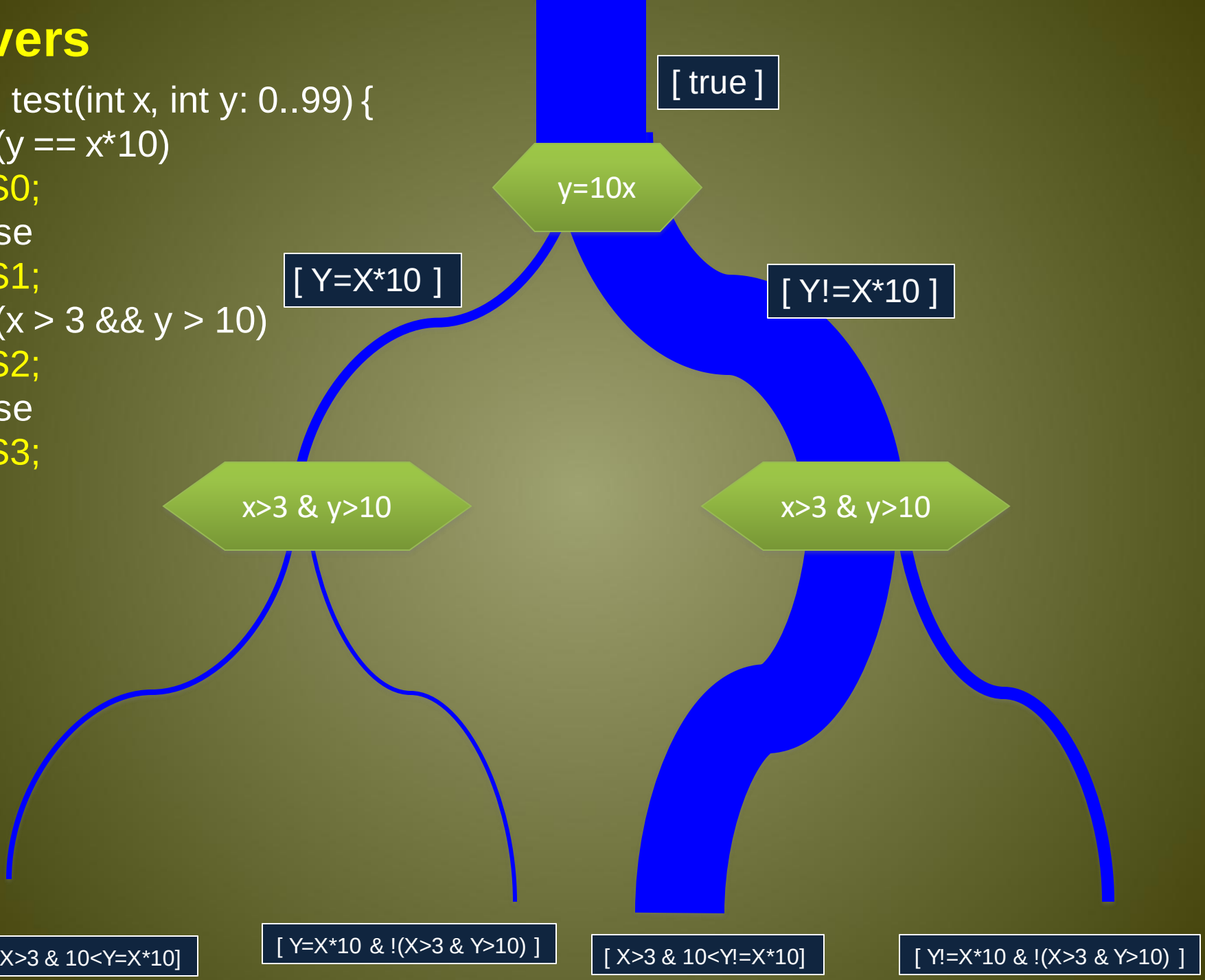
Almost Rivers

```
void test(int x, int y: 0..99) {
   if (y == x*10)
     S0;
   else
     S1;
   if (x > 3 && y > 10)
     S2;
   else
     S3;
}
```



Rivers

```
void test(int x, int y: 0..99) {
   if (y == x*10)
     S0;
   else
     S1;
   if (x > 3 && y > 10)
     S2;
   else
     S3;
}
```



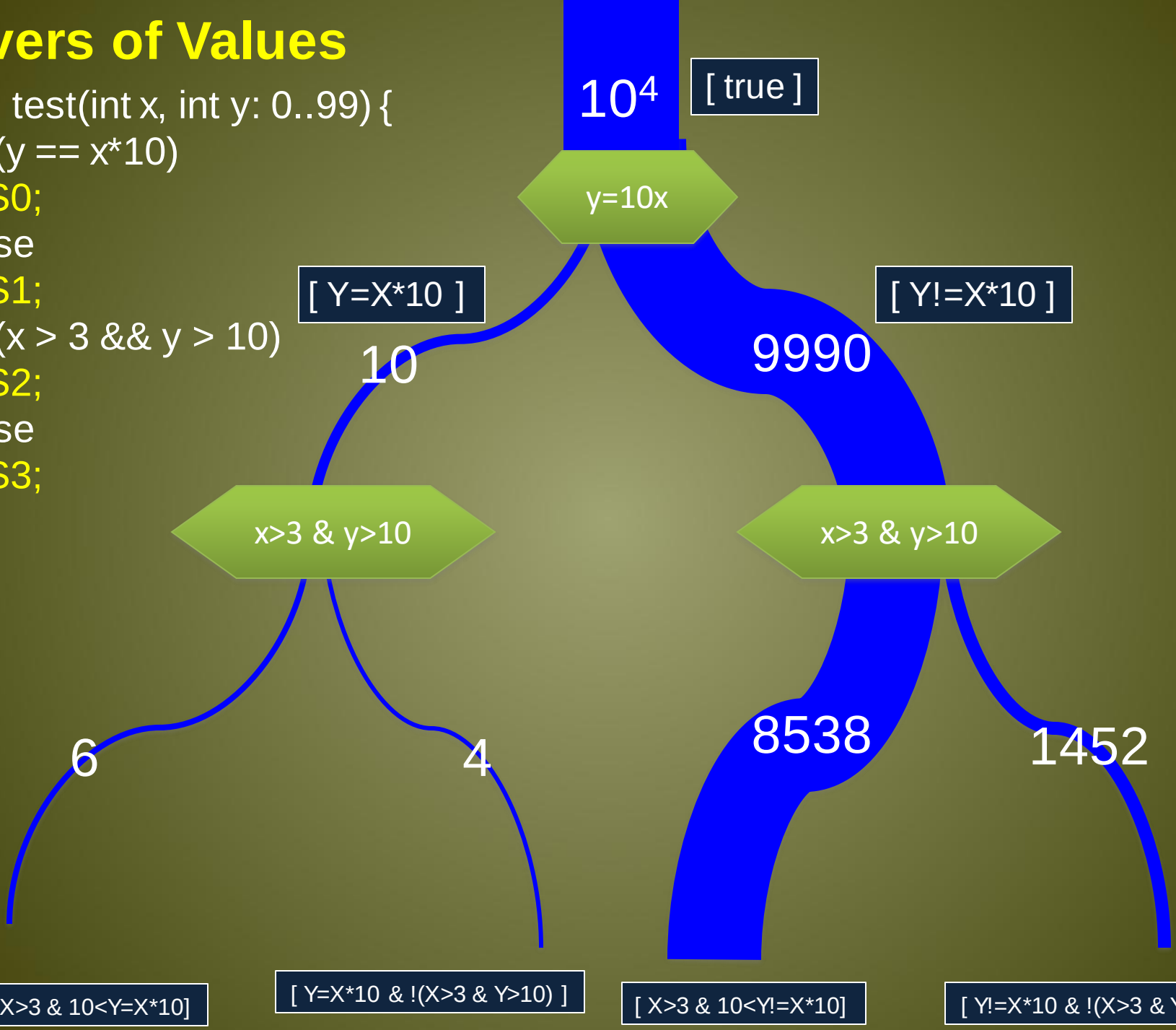
LattE Model Counter

http://www.math.ucdavis.edu/~latte/

Count solutions for
conjunction
of Linear Inequalities

Rivers of Values

```
void test(int x, int y: 0..99) {
   if (y == x*10)
     S0;
   else
     S1;
   if (x > 3 && y > 10)
     S2;
   else
     S3;
}
```



Program Understanding

10^4 [true]

y=10x

10 [Y=X*10]

9990 [Y!=X*10]

x>3 & y>10

x>3 & y>10

6

4

8538

1452

[X>3 & 10<Y=X*10]

[Y=X*10 & !(X>3 & Y>10)]

[X>3 & 10<Y!=X*10]

[Y!=X*10 & !(X>3 & Y>10)]

Probabilities

1

y=10x

0.001

0.999

x>3 & y>10

x>3 & y>10

0.6

0.4

0.855

0.145

0.0006

0.0004

0.8538

0.1452

[X>3 & 10<Y=X*10]

[Y=X*10 & !(X>3 & Y>10)]

[X>3 & 10<Y!=X*10]

[Y!=X*10 & !(X>3 & Y>10)]

Reliability

1

y=10x

0.001

0.999

x>3 & y>10

x>3 & y>10

0.9996 Reliable

0.6

0.4

0.855

0.145

0.0006

0.0004

0.8538

0.1452

[X>3 & 10<Y=X*10]

[Y=X*10 & !(X>3 & Y>10)]

[X>3 & 10<Y!=X*10]

[Y!=X*10 & !(X>3 & Y>10)]

Can you add a progress bar?

How wrong is the program?

Which correct program is *better*?

Triangle Classification

- 2 Correct versions
- 2 Buggy versions
  - One created ourselves accidentally
    - Small typo >= instead of <=
  - One from a student coding exercise
    - Missed one condition

```
public static int classify(int i, int j, int k) {
   if ((i <= 0) || (j <= 0) || (k <= 0))
       return 4;
   int type = 0;
   if (i == j) type = type + 1;
   if (i == k) type = type + 2;
   if (j == k) type = type + 3;
   if (type == 0) {
// Confirm it is a legal triangle before declaring it to be scalene.
      if ((i + j <= k) || (j + k <= i) || (i + k <= j))  type = 4;
      else type = 1;
      return type;
   }
// Confirm it is a legal triangle before declaring it to be isosceles or
// equilateral.
   if (type > 3) type = 3;
   else if ((type == 1) && (i + j > k))  type = 2;
   else if ((type == 2) && (i + k > j))  type = 2;
   else if ((type == 3) && (j + k > i))  type = 2;
   else type = 4;
   return type;
}
```

Correct1

```
public static int classify(int i, int j, int k) {
   if ((i <= 0) || (j <= 0) || (k <= 0))
      return 4;
   int type = 0;
   if (i == j) type = type + 1;
   if (i == k) type = type + 2;
   if (j == k) type = type + 3;
   if (type == 0) {
// Confirm it is a legal triangle before declaring it to be scalene.
      if ((i + j <= k) || (j + k <= i) || (i + k >= j))  type = 4;
      else type = 1;
      return type;
   }
// Confirm it is a legal triangle before declaring it to be isosceles or
// equilateral.
   if (type > 3) type = 3;
   else if ((type == 1) && (i + j > k))  type = 2;
   else if ((type == 2) && (i + k > j))  type = 2;
   else if ((type == 3) && (j + k > i))  type = 2;
   else type = 4;
   return type;
}
```

Buggy1

Correct2

```
public static int classify(int a, int b, int c) {
        if (a <= 0 || b <= 0 || c <= 0)
            return 4;
        if (! (a > c - b && a > b - c && b > a - c))
            return 4;
        if (a == b && b == c)
            return 3;
        if (a == b || b == c || a == c)
            return 2;
        return 1;
}
```

Buggy2

```
public static int classify(int a, int b, int c) {
        if ((a <= 0) || (b <= 0) || (c <= 0))
            return 4;
        if ((a <= c - b) || (a<= b - c) || (b<= a - c))
            return 4;
        if (a == b && b == c)
            return 3;
        if (((a == b) && (b != c)) ||
            ((a == c) && (a != b) && ((b == c) && (b != a))))
            return 2;
        return 1;
    }
```

Which of Correct1 or Correct2 is “better”?

```
public static int classify(int i, int j, int k) {
   if ((i <= 0) || (j <= 0) || (k <= 0))
       return 4;
   int type = 0;
   if (i == j) type = type + 1;
   if (i == k) type = type + 2;
   if (j == k) type = type + 3;
   if (type == 0) {
      if ((i + j <= k) || (j + k <= i) || (i + k <= j))
         type = 4;
      else type = 1;
      return type;
   }
   if (type > 3) type = 3;
   else if ((type == 1) && (i + j > k))  type = 2;
   else if ((type == 2) && (i + k > j))  type = 2;
   else if ((type == 3) && (j + k > i))  type = 2;
   else type = 4;
   return type;
}
```

```
public static int classify(int a, int b, int c) {
    if (a <= 0 || b <= 0 || c <= 0)
       return 4;
    if (! (a > c - b && a > b - c && b > a - c))
       return 4;
    if (a == b && b == c)
       return 3;
    if (a == b || b == c || a == c)
       return 2;
    return 1;
}
```

Correct1 All Partitions with sizes

463344	(k+i>j)&&((k+j)>i)&&(j+i)>k)&&(j!=k)&&(i!=k)&&(i!=j)&&(k>0)&&(j>0)&&(i>0)
159250	(k+i<=j)&&(k+j>i)&&(j+i>k)&&(j!=k)&&(i!=k)&&(i!=j)&&(k>0)&&(j>0)&&(i>0)
159250	(k+j<=i)&&(j+i>k)&&(j!=k)&&(i!=k)&&(i!=j)&&(k>0)&&(j>0)&&(i>0)
159250	(j+i<=k)&&(j!=k)&&(i!=k)&&(i!=j)&&(k>0)&&(j>0)&&(i>0)
10000	(i<=0)
9900	(j<=0)&&(i>0)
9801	(k<=0)&&(j>0)&&(i>0)
7252	(k+j>i)&&(j==k))&&(i!=k)&&(i!=j)&&(k>0)&&(j>0)&&(i>0)
7252	(j+i>k)&&(i!=k)&&(i==j)&&(k>0)&&(j>0)&&(i>0)
7252	(k+i>j)&&(i==k)&&(i!=j)&&(k>0)&&(j>0)&&(i>0)
2450	(k+j<=i)&&(j==k)&&(i!=k)&&(i!=j)&&(k>0)&&(j>0))&&(i>0)
2450	(j+i<=k)&&(i!=k))&&(i==j)&&(k>0)&&(j>0)&&(i>0)
2450	(k+i<=j)&&(i==k)&&(i!=j)&&(k>0)&&(j>0)&&(i>0)
99	(i==k)&&(i==j)&&(k>0)&&(j>0)&&(i>0)

Correct2 All Partitions with sizes

463344	(a!=c)&&(b!=c)&&a!=b&&b>a-c&&(a>(b-c)&&(a>(c-b)&&(c>0)&&(b>0)&&(a>0)
161700	b<=(a-c))&&(a>(b-c)))&&(a>(c-b)))&&(c>0))&&(b>0))&&(a>0)
161700	(a<=(b-c))&&(a>(c-b)))&&(c>0))&&(b>0))&&(a>0)
161700	(((a<=(c-b))&&(c>0))&&(b>0))&&(a>0)
10000	(a<=0)
9900	(b<=0)&&(a>0)
9801	(c<=0)&&(b>0)&&(a>0)
7252	a==c&&(b!=c)&&(a!=b))&&b>a-c&&a>b-c&&a>c-b&&c>0&&b>0&&(a>0)
7252	b!=c&&a==b&&(b>(a-c)))&&(a>(b-c)))&&(a>(c-b)))&&(c>0))&&(b>0))&&(a>0)
7252	b==c&&a!=b&&(b>(a-c)))&&(a>(b-c)))&&(a>(c-b)))&&(c>0))&&(b>0))&&(a>0)
99	(i==k)&&(i==j)&&(k>0)&&(j>0)&&(i>0)

Correct2 is “better”

it has less paths dealing with the Illegal Triangle case

Test harness

```
public static void check(int a, int b, int c) {
        int test = classifyBuggy1(a, b, c);
        int oracle = classifyCorrect1(a, b, c);
        assert (test == oracle);
}
```

Record path conditions
when assertion fails and
count their sizes

Counting size of failing partitions

Correct vs Buggy1

2 failing paths: 463344 and 154448

Correct vs Buggy2

2 failing paths: 7252 and 7252

Buggy2 is closer than Buggy1

Observation

- Buggy1 is closer in edit distance to a correct version
  - One might intuitively think it is thus “closer” to correct
  - Clearly this is not the case, and, which syntactically correct version should one consider?
  - However, might it be an issue for a progress bar?

Side track

- Mutations in programs can have bigger effect than one might think
- Notice how a small <= to >= change made the program incorrect on more inputs than a logical mistake
- Can we rank mutation operators according to how much of the behavior of a program they mutate?

Mutation Example

```
boolean classify1(int i, int j : 0..99) {
   return i <= j;
}
```

1 99% wrong

```
boolean classify1(int i, int j) {
   return i >= j;
}
```

2 1% wrong

```
boolean classify1(int i, int j) {
   return i < j;
}
```

3 49.5% wrong

```
boolean classify1(int i, int j) {
   return i == j;
}
```

4 49.5% wrong

```
boolean classify1(int i, int j) {
   return true;
}
```

Lets try it…

10

```
int Correct(int x : 0..9)
   return x*2;
```

Cut and paste

```
int A(int x)
   if (x == 2) return 4;
   if (x == 6) return 12;
   return 0;
```

7

saw the comment…

```
int A(int x)
   if (x == 1) return 2;
   if (x == 2) return 4;
   if (x == 6) return 12;
   return x+3;
```

6

and one more…

```
int A(int x)
   if (x == 1) return 2;
   if (x == 2) return 4;
   if (x == 6) return 12;
   return 0;
```

6

And Again…

```
Int B(int x, int y)
   if (x == 0 && y == 0) return 0;
   if (x == 0 && y == 1) return 1;
   if (x == 1 && y == 0) return 1;
   return 0;
```

9997

```
Int B(int x, int y)
   if (x == 0 && y == 0) return 0;
   if (x == 0 && y == 1) return 1;
   if (x == 1 && y == 0) return 1;
   if (x == 0 && y == 33) return 33;
   if (x == 33 && y == 0) return 33;
   return 0;
```

9995

And Again...

10000

```
int Correct(int x,y : 0..99)
    return x+y;
```

```
Int B(int x, int y)
   if (x == 0 && y == 0) return 0;
   if (x == 0 && y == 1) return 1;
   if (x == 1 && y == 0) return 1;
   if (x == 0 && y == 33) return 33;
   if (x == 33 && y == 0) return 33;
   return x-y;
```

9898

```
Int B(int x, int y)
   return x-y;
```

9900

Oh no, backwards!

```
Int B(int x, int y)
   if (x >= y)
      return x – y;
   return y - x;
```

9801

Time passed and he played CodeHunt…

To get to the interesting examples!

Last one...

```
Bool Correct(int x : 0..199)
    return (x>=50) ? false : true;
```

49

```
Bool C(int x) {
   if (x == 0) return true;
   return false;
```

46

```
Bool C(int x) {
   if (x == 0) return true;
   if (x == 4) return true;
   if (x == 32) return true;
   if (x == 36) return true;
   return false;
```

46

```
Bool C(int x) {
   if (x == 0) return true;
   if (x == 4) return true;
   if (x == 32) return true;
   if (x == 36) return true;
   if (x >= 64) return false;
   return false;
```

14

```
Bool C(int x) {
   if (x >= 64) return false;
   else return true;
```

Conclusions

- The bounds are important
  - Even in CodeHunt now you can expose the bounds PEX is using
  - For model counting the bounds must be exposed
- Should we give raw scores or descriptions
  - Way Off, A little better, Now you are getting there,…
- Can we actually do Model Counting for all the required domains?
  - Strings are new to the model counting game

The Green Framework

http://green-solver.googlecode.com

Already integrated into
Symbolic PathFinder

Resources

- ISSTA 2012
  - Probabilistic Symbolic Execution
- FSE 2012
  - Green: Reduce, Reuse and Recycle Constraints…
- ICSE 2013
  - Software Reliability with Symbolic PathFinder
- PLDI 2014
  - Compositional Solution Space Quantification for Probabilistic Software Analysis
- FSE 2014
  - Statistical Symbolic Execution with Informed Sampling
- ASE 2014
  - Exact and Approximate Probabilistic Symbolic Execution for Nondeterministic Programs
- Implemented in Symbolic PathFinder
  - Using LattE