What it is all about

- We read texts on static analysis, program analysis, and abstract interpretation
- Covered lattices, fixpoints, partial evaluation, symbolic execution, transfer functions, CFGs
- \( 	op \) (no possible values), \( ot \) (all possible values)
- So what?
- This paper is about cXprop, our implementation of these ideas with an eye towards pluggable abstract domains
Shameless plug:

I will give a presentation about

Fernando J. Corbató
and the talk he gave when receiving his Turing award:

On Building Systems That Will Fail

3:05 pm tomorrow (Friday) here (LCR)
A few other static analysis tools:

- Program Analyzer Generator (PAG)
  - Needs from the user:
    - lattice
    - transfer functions
    - language description
    - fixpoint solution method

- McCAT
  - Generalized Constant Propagation (GCP)

- Machine SUIF
  - based on SUIF
  - different interface
CIL

- C Intermediate Language – developed at UCB
- Cleans up C to a few core constructs
  - removes syntactic sugar (like “->” notation)
  - arrays become pointers
  - all loops become while loops
- Works on real programs
  - handles ANSI-C, Microsoft C, and GNU C
  - SPEC 95, linux kernel, , bzip
C : A million and one ways . . .

- To shoot yourself (and our analysis) in the foot
  - Stack manipulations
  - External calls
  - Floating point
  - Order of evaluation
  - Concurrency
  - Missing returns
Two important pieces:

- **Write sets**
  - in figure, \( x \) is \( \bot \) inside \( \text{foo} \)
  - but what if \( \text{foo} \) does not use \( x \)?

- **Pointer analysis**
  - what to do with reading or writing pointers
  - function pointer calls

```c
// x is global
x = 5;
foo();
...
x = 10;
foo();
...
```
let mult d1 d2 tp =
match d1, d2 with
  | Constant(z), _ when (isZero z) ->
    Constant (zero)
  | Bottom, _
  | _, Bottom -> Bottom
  | Constant(e1), Constant(e2) ->
    conc_to_abs (BinOp(Mult,e1,e2,tp))

int tricky () {
  int x = 1;
  int count = 0;
  do {
    int b = x;
    if (b != 1)
      x = 2;
    count += x;
  } while (count < 10);
  return x;
}
int tricky () {
    int x = 1;
    int count = 0;
    do {
        int b = x;
        if (b != 1)
            x = 2;
        count += x;
    } while (count < 10);
    return x;
}

let plusa (d1:t) (d2:t) tp =
    match d1, d2 with
    | Bottom, _
    | _ , Bottom -> Bottom
    | Even, Even
    | Odd, Odd -> Even
    | Even, Odd
    | Odd, Even -> Odd
Bitwise

let lnot (d, dk) tp =
  if ((dk = no_bottoms) && (d = I.zero))
    then TbTrue
  else if (I.logand d dk) <> I.zero then TbFalse
  else TbBottom

int tricky () {
  int x = 1;
  int count = 0;
  do {
    int b = x;
    if (b != 1)
      x = 2;
    count += x;
  } while (count < 10);
  return x;
}
Information

- \textit{entropy} in information theory (not physics)
- Information
  - \( j = \# \text{ of possible values that can be represented} \)
  - \( k = \# \text{ of values that might be represented} \)
- Information bits known
  - \( \lceil \log_2 j \rceil = \# \text{ of information bits} \)
  - \( \lceil \log_2 k \rceil = \# \text{ of information bits unknown} \)
  - \( \lceil \log_2 j \rceil - \lceil \log_2 k \rceil = \# \text{ of information bits known} \)
Domain Comparison

![Bar chart showing domain comparison across different datasets.](chart.png)
Conclusion

- cXprop performs abstract interpretation
- Convenient interface for new abstract domains
- Five domains already implemented:
  - parity, constant, value set, interval, bitwise
- Not all programs analyze the same way
- Pick the right domain for the program
- May be downloaded at:

  http://www.cs.utah.edu/~coop/research/cxprop