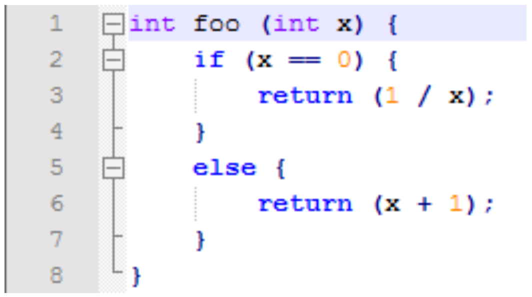
Bug hunting and Compressed Sensing

Suhail Shergill

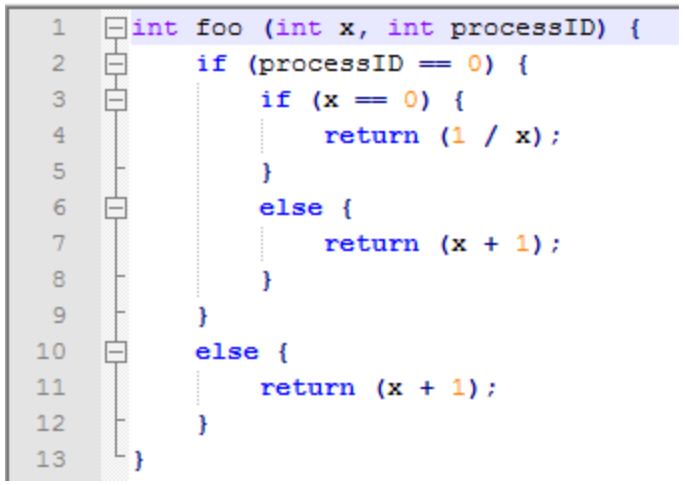
A buggy program



```
int foo (int x) {
    if (x == 0) {
        return (1 / x);
    }
    else {
        return (x + 1);
    }
}
```

(x == 0)	(x != 0)	Failure?
TRUE	FALSE	TRUE
FALSE	TRUE	FALSE

A buggy parallel program



```
int foo (int x, int processID) {
    if (processID == 0) {
        if (x == 0) {
            return (1 / x);
        }
        else {
            return (x + 1);
        }
    }
    else {
        return (x + 1);
    }
}
```

(pID == 0)	(pID != 0)	(x == 0)	(x !=0)	Failure?
TRUE	FALSE	TRUE	FALSE	TRUE
TRUE	FALSE	FALSE	TRUE	FALSE
FALSE	TRUE	FALSE	FALSE	FALSE

Uniform Sampling

The data matrix

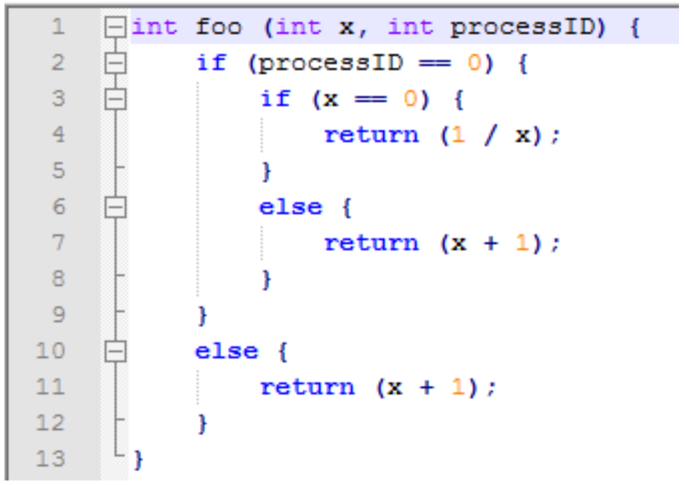
	Feature 1	Feature 2	Feature 3	…
Process 1	0	1	0	…
Process 2	0	0	1	…
Process 3	0	0	0	…
…	…	…	…	…

.. and the problem?

Is there any hope?

- Redundancy amongst processes
  - Eg. All nodes except for the root node (processID == 0) are doing the same thing in our example
- Redundancy amongst the features
- Additional structure amongst the features

Where is the structure?



```
int foo (int x, int processID) {
    if (processID == 0) {
        if (x == 0) {
            return (1 / x);
        }
        else {
            return (x + 1);
        }
    }
    else {
        return (x + 1);
    }
}
```

(pID == 0)	(pID != 0)	(x == 0)	(x !=0)	Possible?
TRUE	FALSE	TRUE	FALSE	YES
TRUE	FALSE	FALSE	TRUE	YES
FALSE	TRUE	FALSE	FALSE	YES
FALSE	FALSE	FALSE	FALSE	NO
FALSE	TRUE	TRUE	FALSE	NO
TRUE	FALSE	TRUE	TRUE	NO

So what’s the proposed strategy?

	Feature 1	Feature 2	Feature 3	…
Process 1	0	1	0	…
Process 2	0	0	1	…
Process 3	0	0	0	…
…	…	…	…	…

- Redundancy => finite alphabet matrix (with block constant structure)
- Use program structure to constrain the set of possible recovery matrices

Questions?