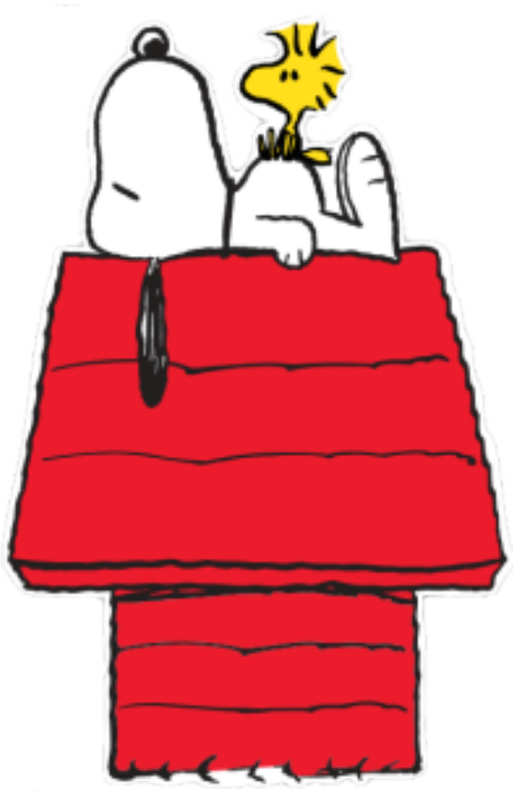
Towards a unifying framework for tuning analysis precision by program transformation

Mila Dalla Preda
University of Verona, Italy

Gabbrielli's Festschrift, 27th November 2020

Program Analysis



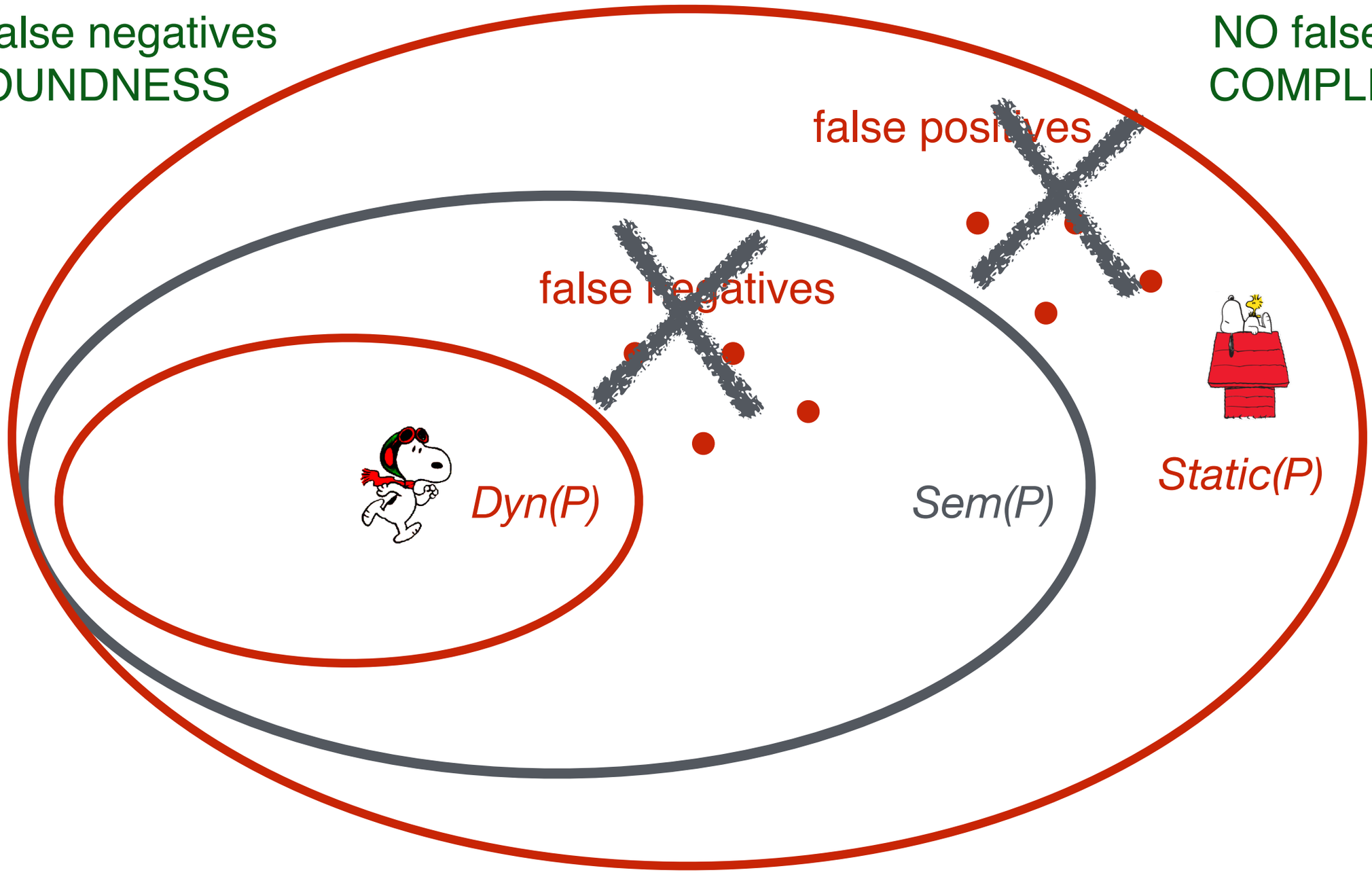
Static program analysis

Dynamic program analysis

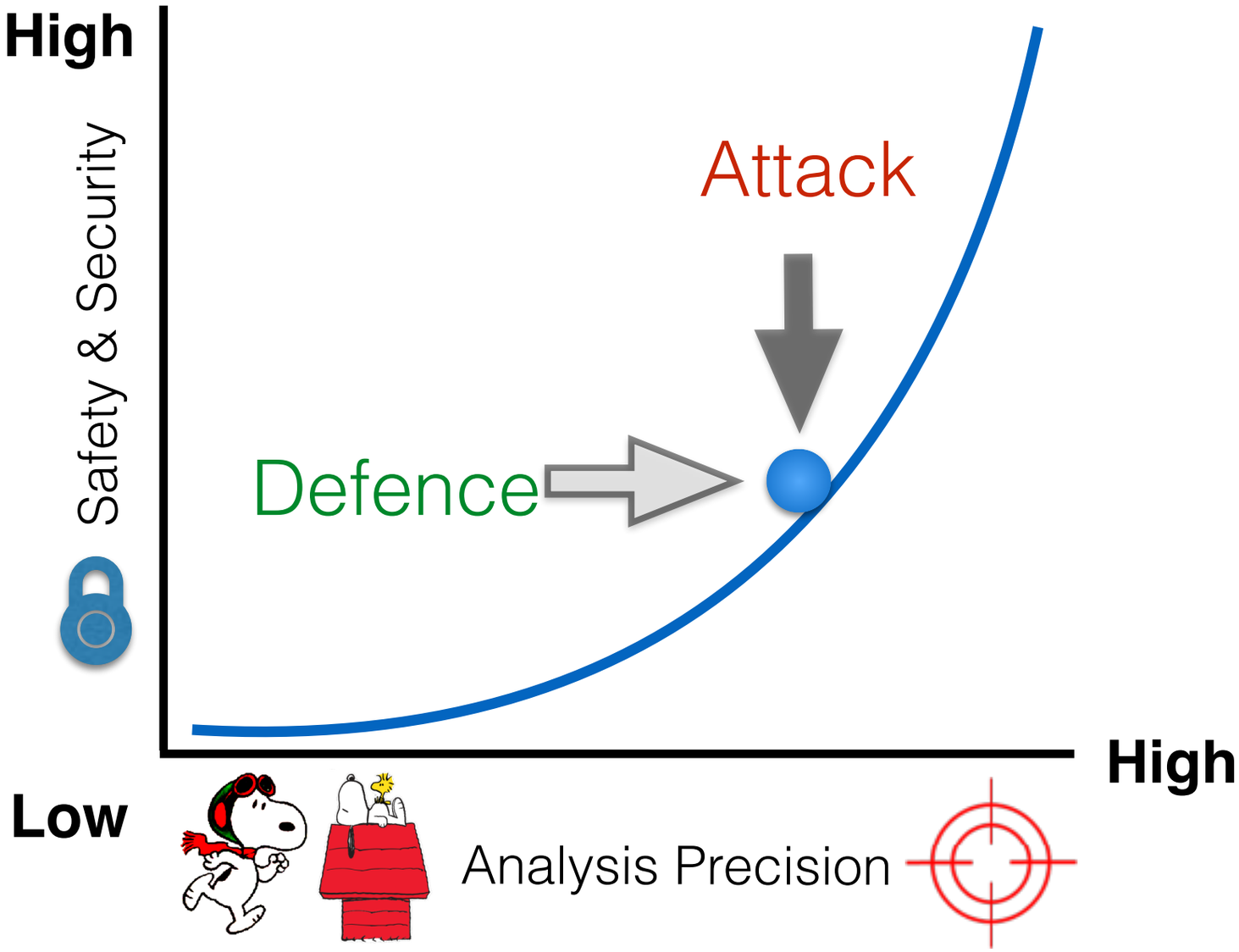
Program Analysis

NO false negatives
SOUNDNESS

NO false positives
COMPLETENESS

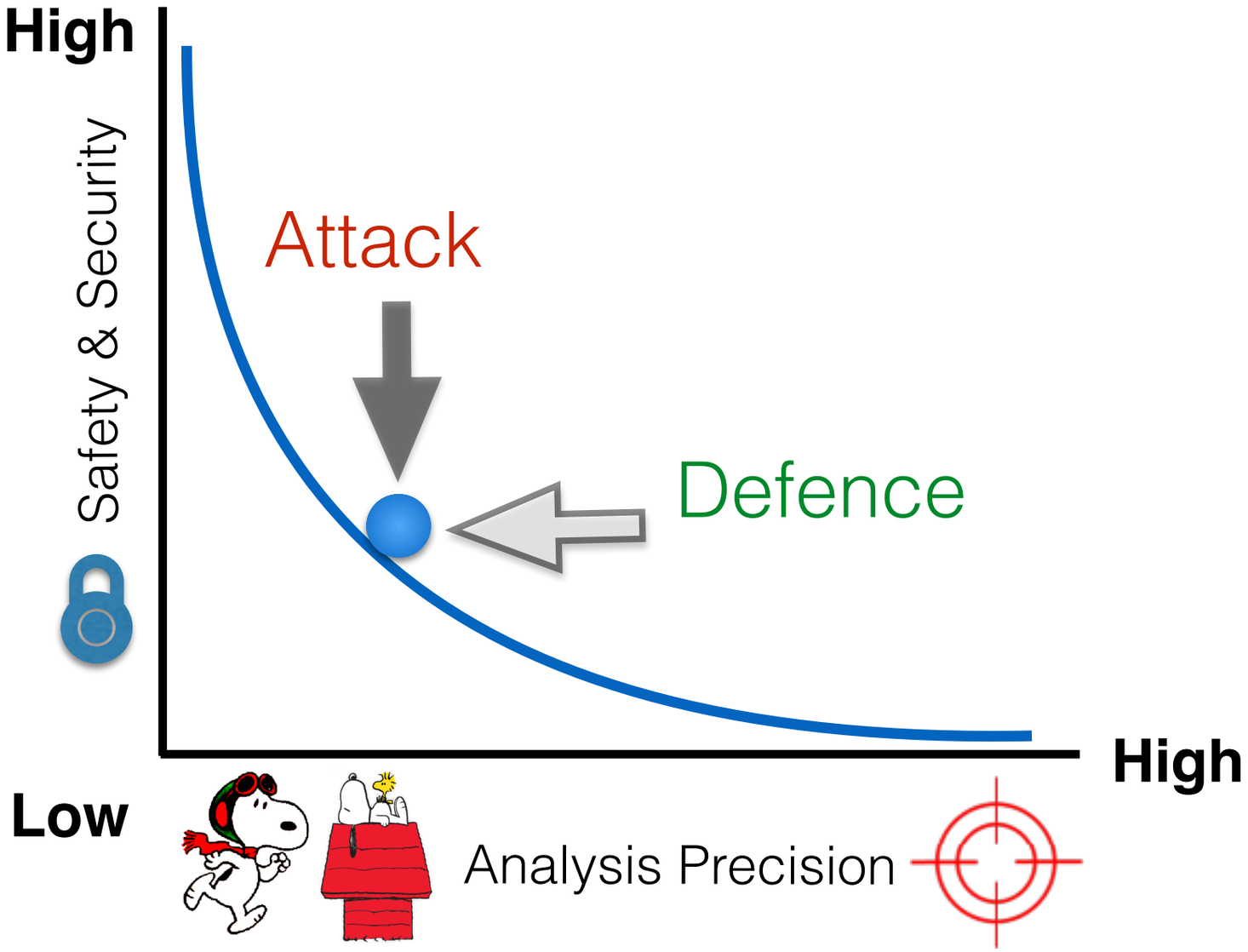


Code Debugging/Verification



The attacker exploits bugs

Software Protection



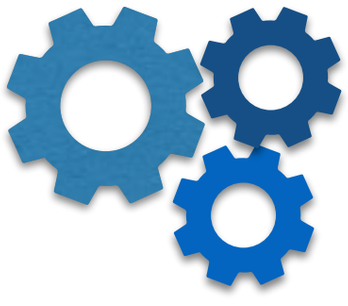
The attacker reverse engineer code

Code Obfuscation

[Collberg et al. POPL 1998]
A program transformation O: Programs ⟶ Programs is a code obfuscation if:

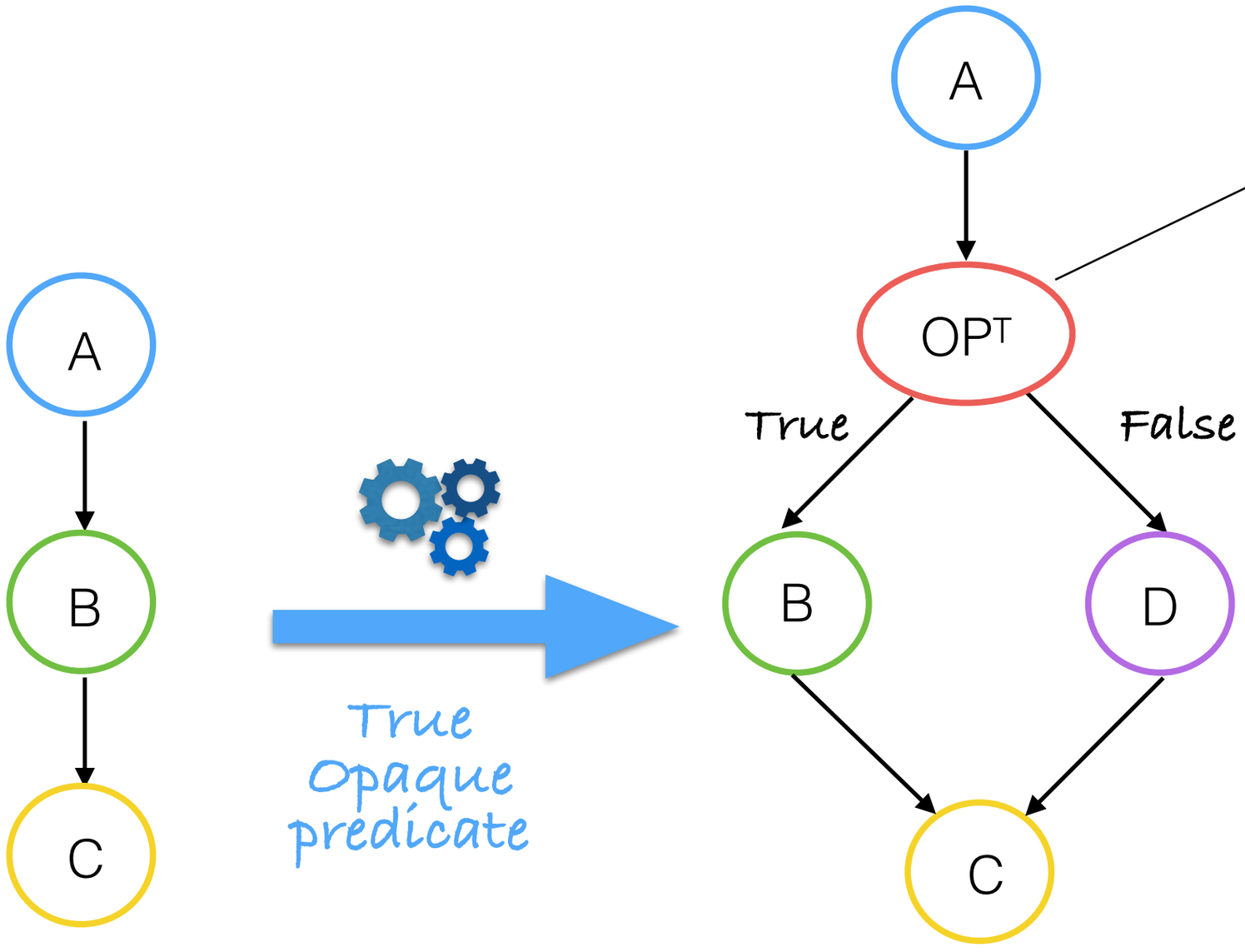
- O **preserves the observational behavior** of programs
- O(P) is **more difficult to analyse**

```
class A { public int Count() { return 1; } }

class Program
{
    static void Main(string[] args)
    {
        var seq = "Roslyn";
        var a = new A();

        if (seq.Count() > 0 && seq.Count() < 10) Console.WriteLine();
        if (0 < seq.Count()) Console.WriteLine();

        if ("Roslyn".Count() >= 1) {
            Console.WriteLine();
        }

        if (1 <= "Roslyn".Count()) {
            Console.WriteLine();
        }

        // these should not trigger our code issue
        if (a.Count() > 0) Console.WriteLine();
        if (0 < a.Count()) Console.WriteLine();

        if (1 < seq.Count()) {
            Console.WriteLine();
        }

        if (0 < seq.Count()) {
            Console.WriteLine();
        }

        if (seq.Count() > 1) Console.WriteLine();
        if (1 < seq.Count()) Console.WriteLine();
```



Code obfuscation

Code Obfuscation & Static Analysis

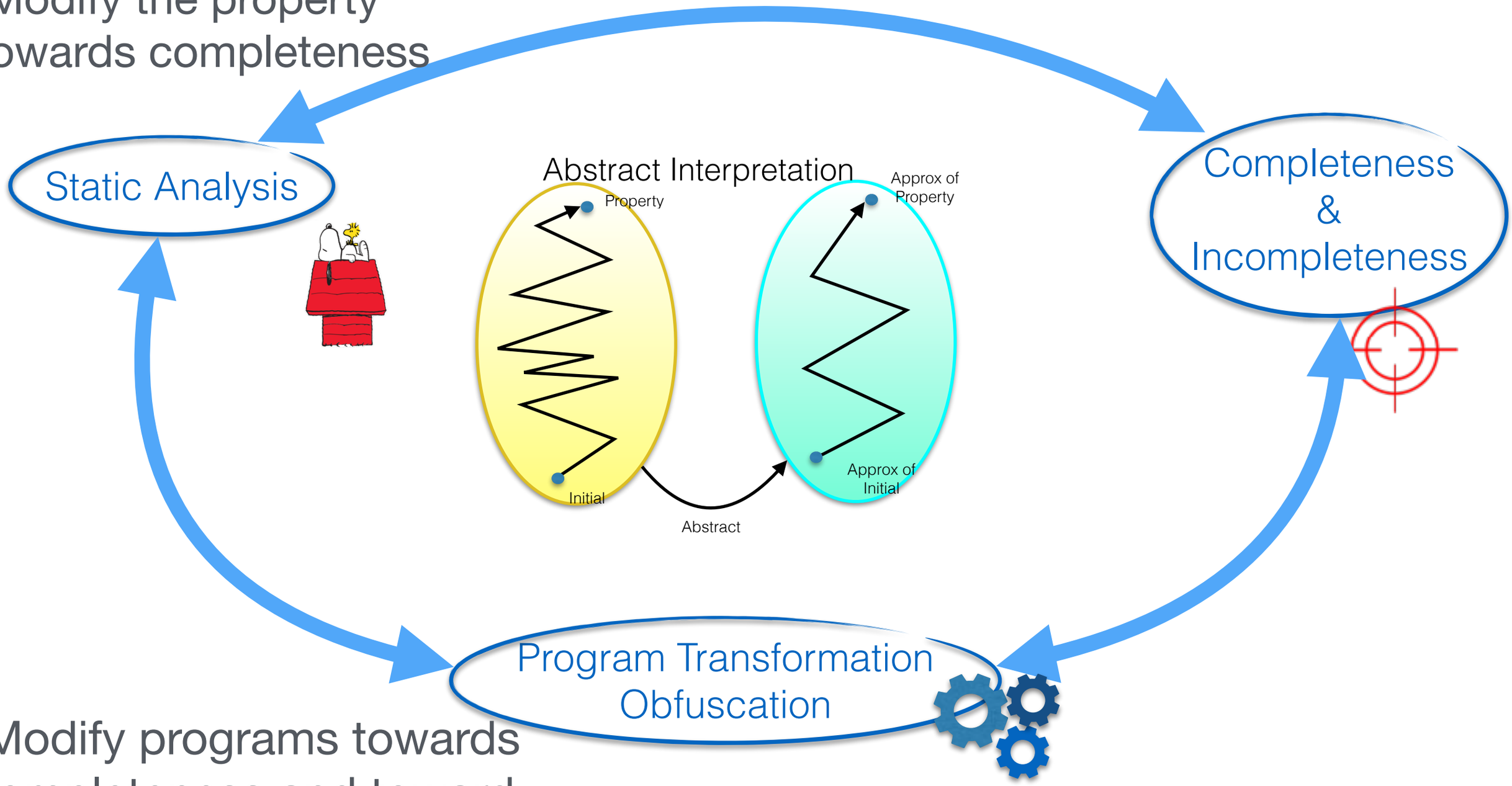


Add *false positive* for the control flow analysis

Incompleteness

Code Transformations & Static Analysis

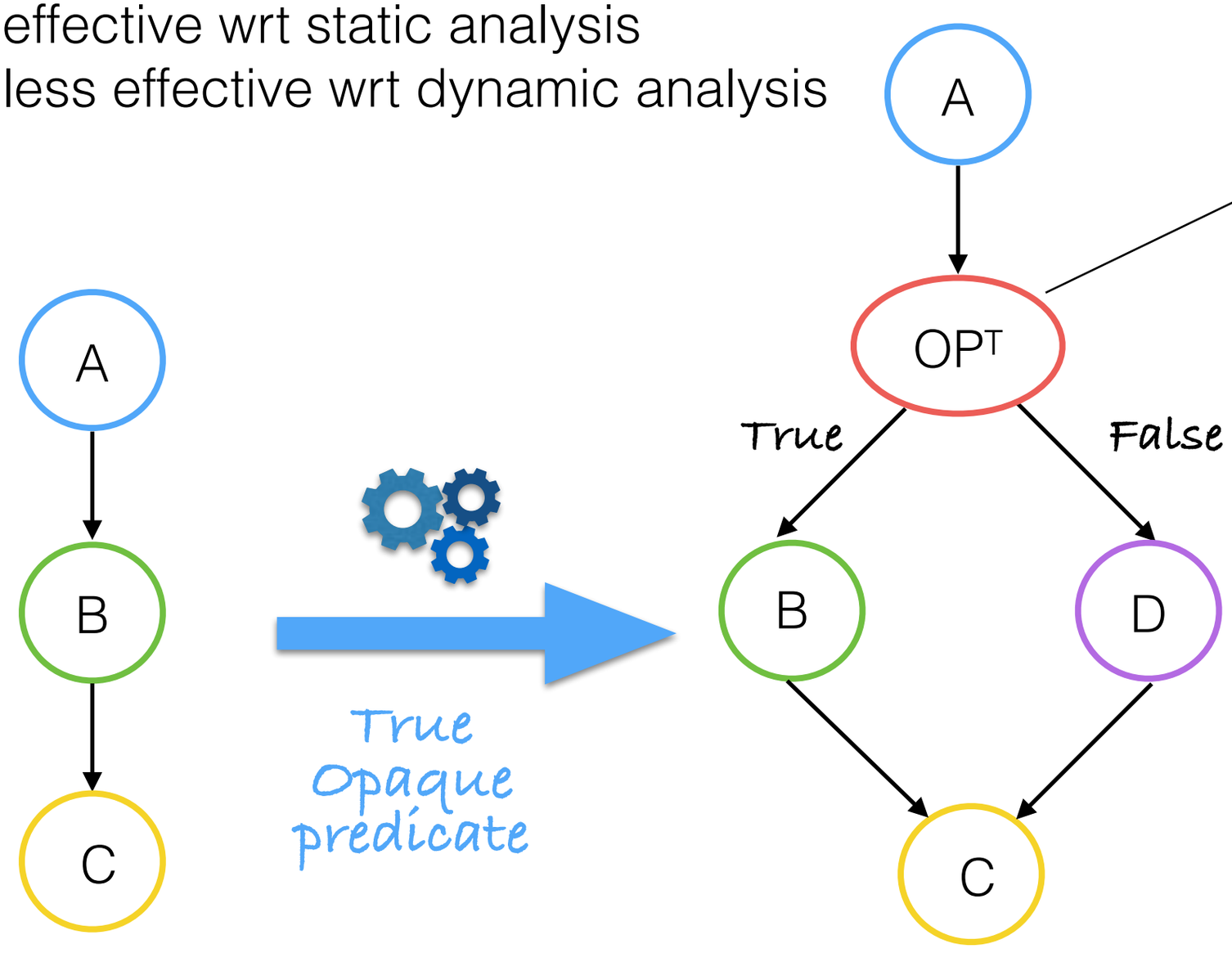
➡Modify the property towards completeness



➡Modify programs towards completeness and toward incompleteness

Code Obfuscation & Static Analysis

➡ effective wrt static analysis
➡ less effective wrt dynamic analysis



Add *false positive* for the control flow analysis

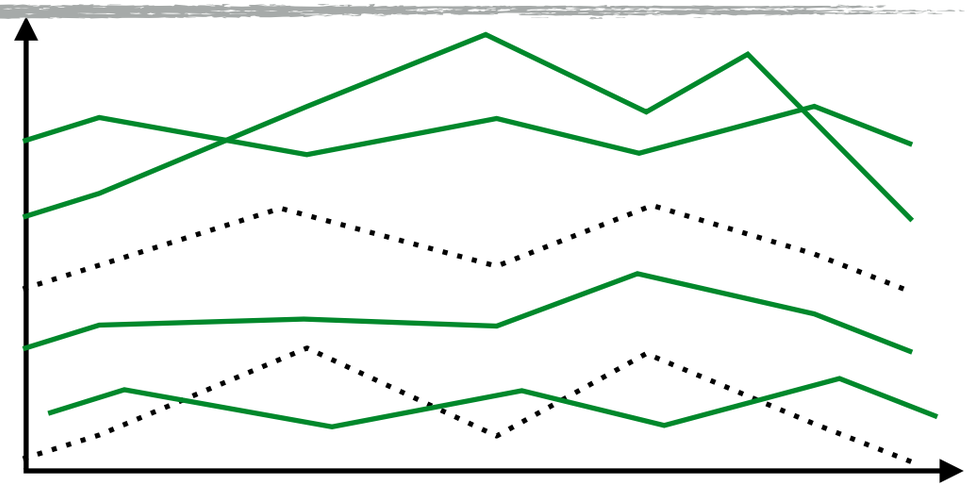
Incompleteness

Code Obfuscation & Dynamic Analysis

What does it mean to complicate/confuse dynamic analysis?

Dynamic Analysis

Analyze a finte subset of finte program traces to infer informations of the whole program, like in program testing and fuzzing

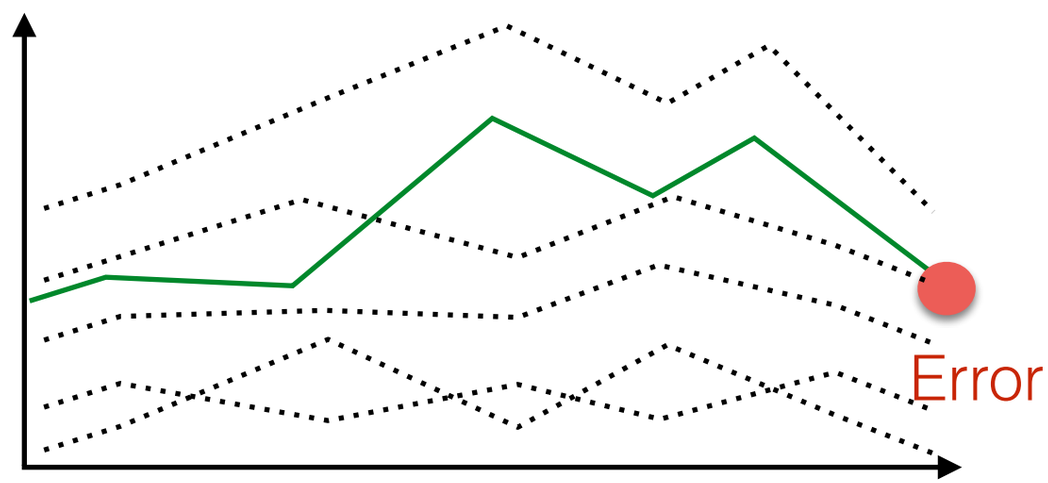


➡ Increase false negatives?

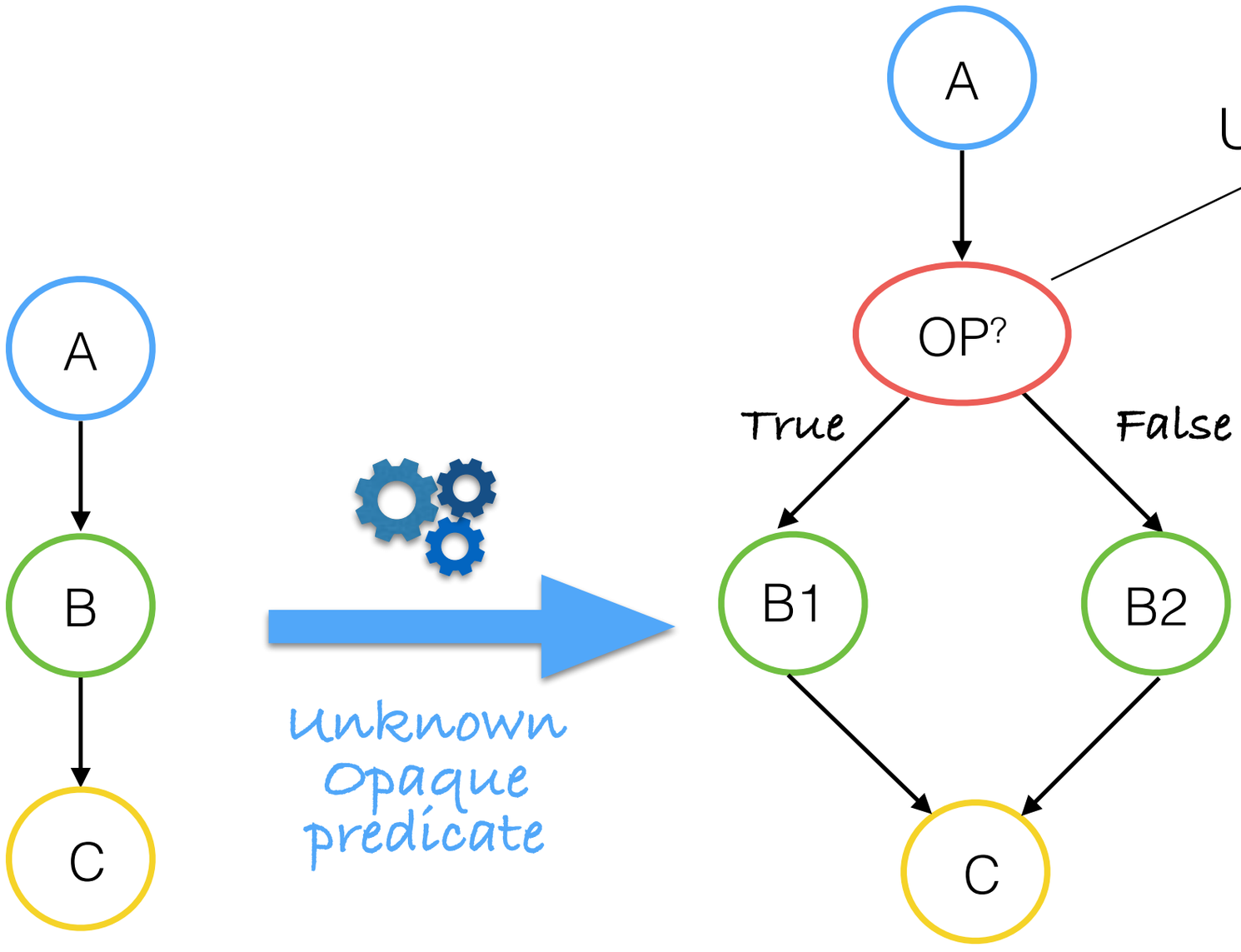
➡ Soundness can be forced or harmed by transforming the analysis or the program?

Analyze a single trace to better understand what went wrong or for runtime monitoring/verification

➡ Insert useless computations



Code Obfuscation & Dynamic Analysis



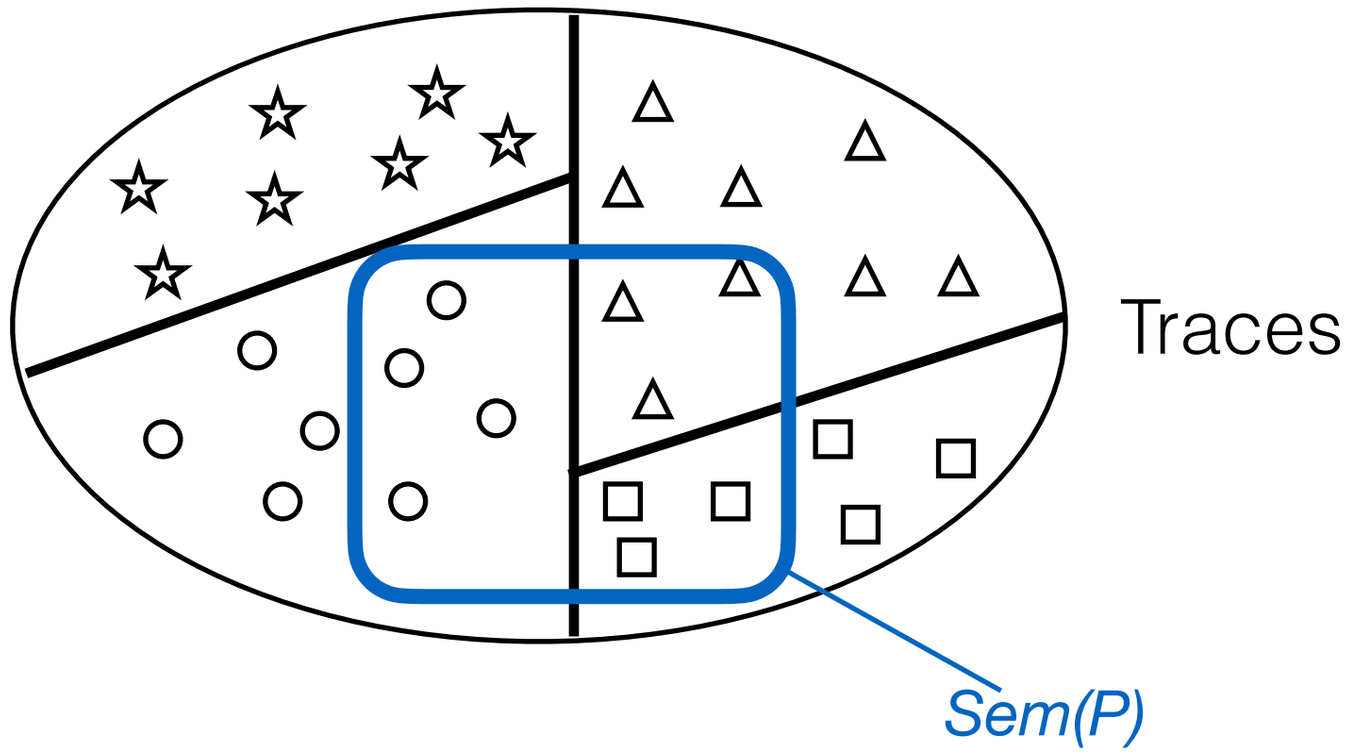
Add *diversification* for the control flow analysis

Unsoundenss?

Formalizing Dynamic Analysis

Property of single trace (no properties of sets of traces)

Equivalence Relation $\mathcal{A}$



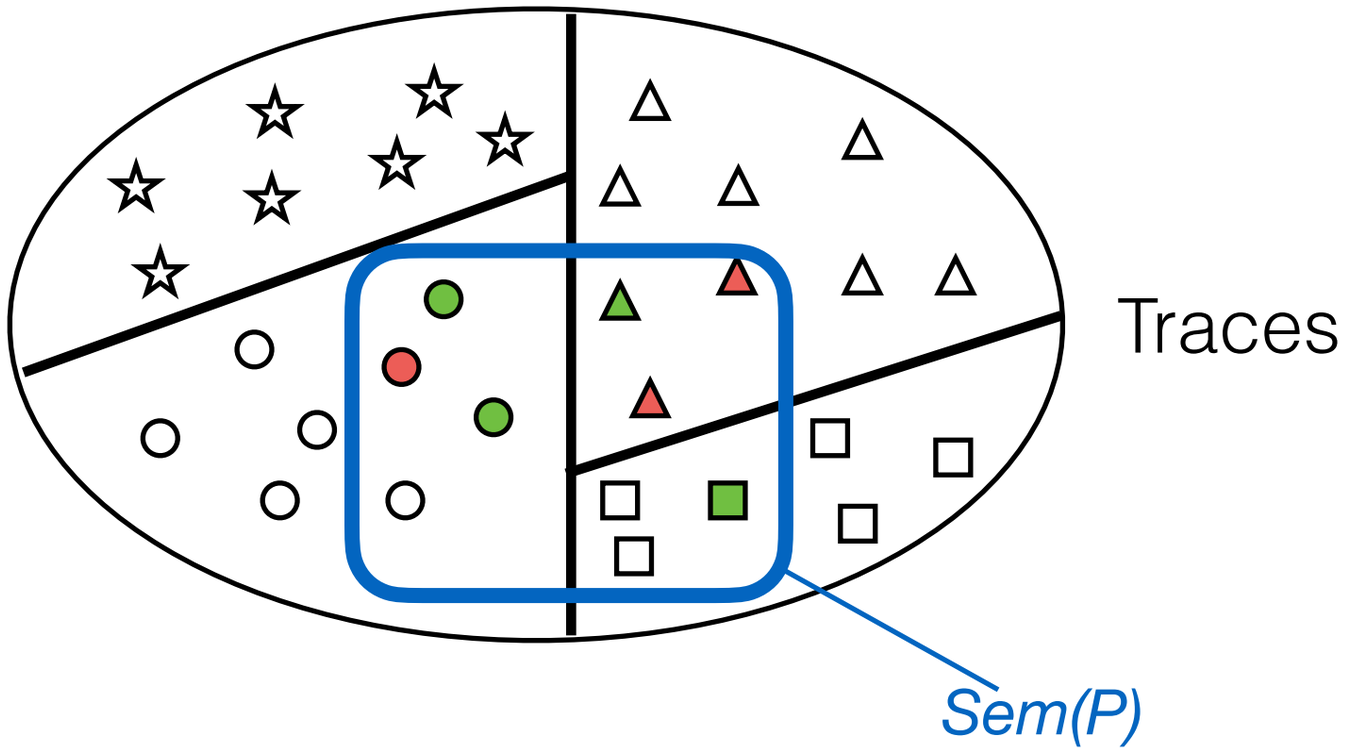
$$\mathcal{A}(Sem(P)) = \{[\sigma]_{\mathcal{A}} \mid \sigma \in Sem(P)\}$$

Formalizing Dynamic Analysis

Property of single trace (no properties of sets of traces)

Equivalence Relation $\mathcal{A}$

Dynamic Analysis
(Exe(P),$\mathcal{A}$)
sound
unsound



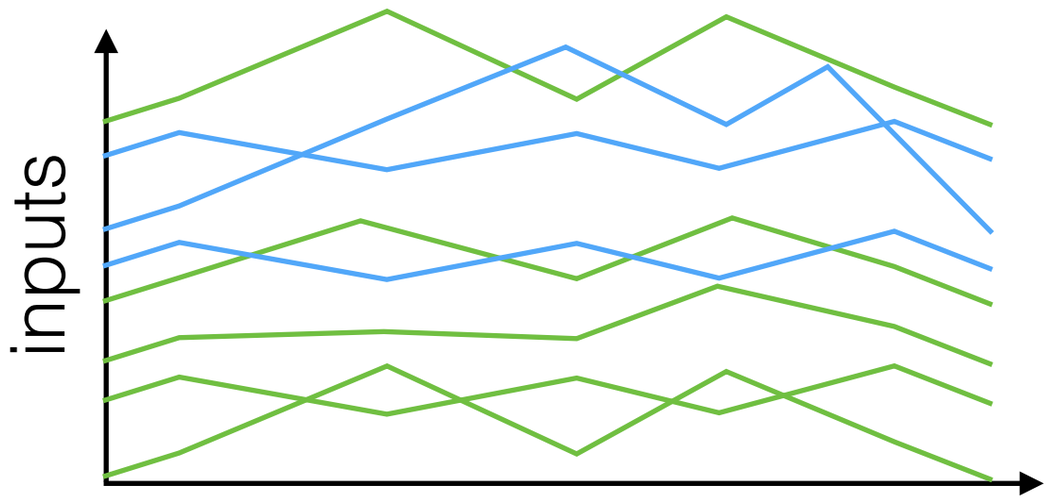
Soundness

$$\mathcal{A}(Sem(P)) = \mathcal{A}(Exe(P))$$

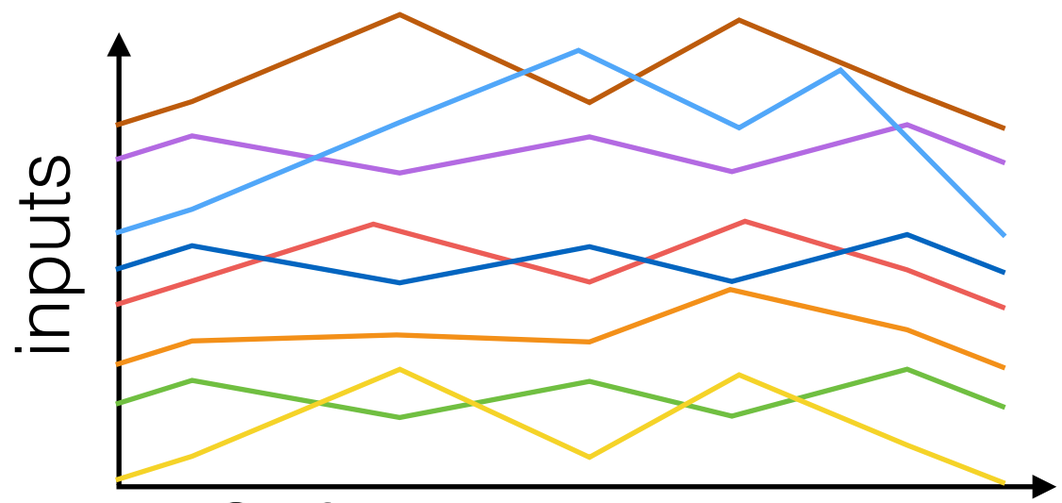
$$\mathcal{A}(Sem(P)) = \{[\sigma]_{\mathcal{A}} \mid \sigma \in Sem(P)\}$$

Obfuscating Dynamic Analysis

The key for harming dynamic analysis is **diversification wrt the property being analysed**



Original program



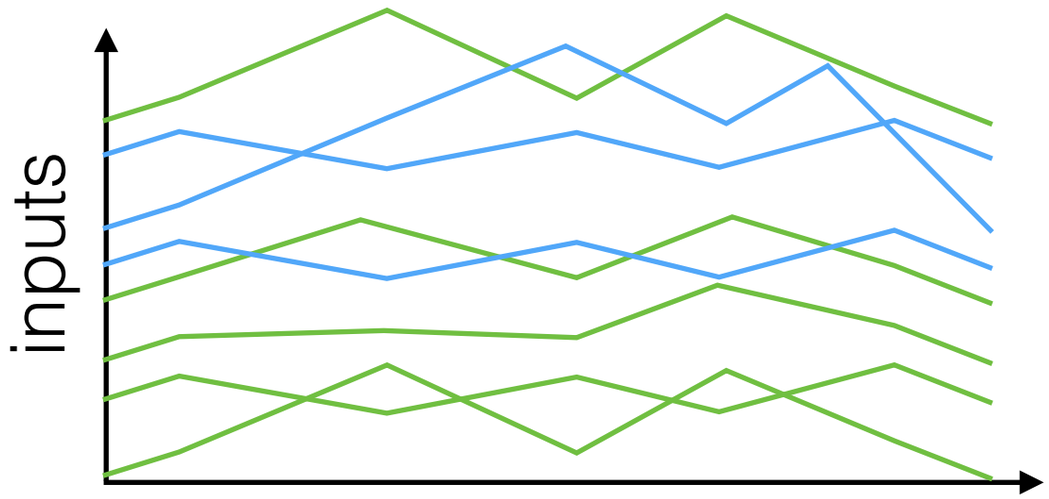
Obfuscated program

Colours represents the equivalence classes wrt $\mathcal{A}$

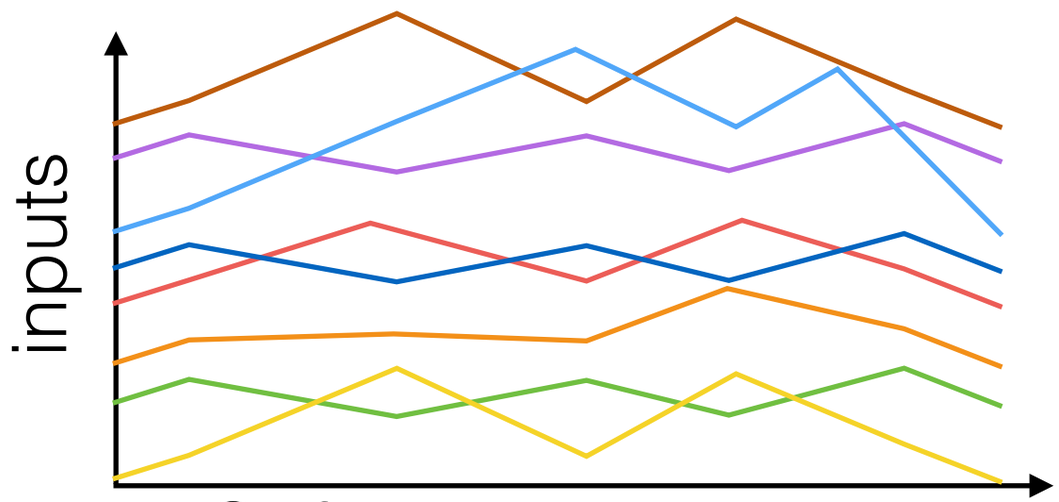
Ideally: specialise the program for every input wrt $\mathcal{A}$

Obfuscating Dynamic Analysis

The key for harming dynamic analysis is **diversification wrt the property being analysed**



Original program

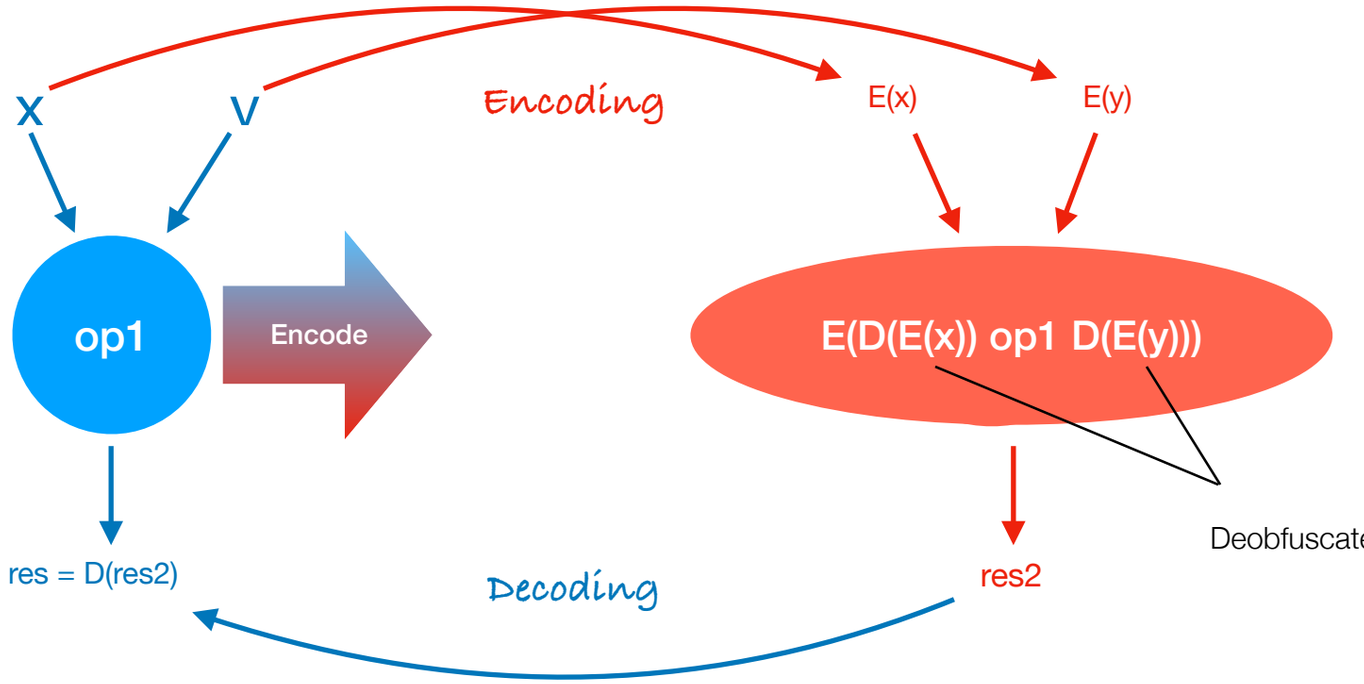


Obfuscated program

A program transformation O: Programs ⟶ Programs obfuscates property $\mathcal{A}$:

- O **preserves the observational behavior** of programs
- The property $\mathcal{A}$ of O(P) is **diversified** wrt P

Data Obfuscation



Data Obfuscation

P

```
input x;
sum := 0;
while x < 50
```

• $X = [x, 49]$

```
    sum := sum + x;
    x := x + 1;
```

$\mathcal{T}(P)$

```
input x;
x := 2*x;        (Encoding)
sum := 0;
while x < 2*50
```

• $X = [x, 2 * 50 - 1]$

```
    sum := sum + x/2;
    x := x + 2;
x:= x/2;         (Decoding)
```

No effects on dynamic analysis

Dynamic Data Obfuscation

P

```
input x;
sum := 0;
while x < 50
• X = [x, 49]
    sum := sum + x;
    x := x + 1;
```

Diversification
Parametric data encoding

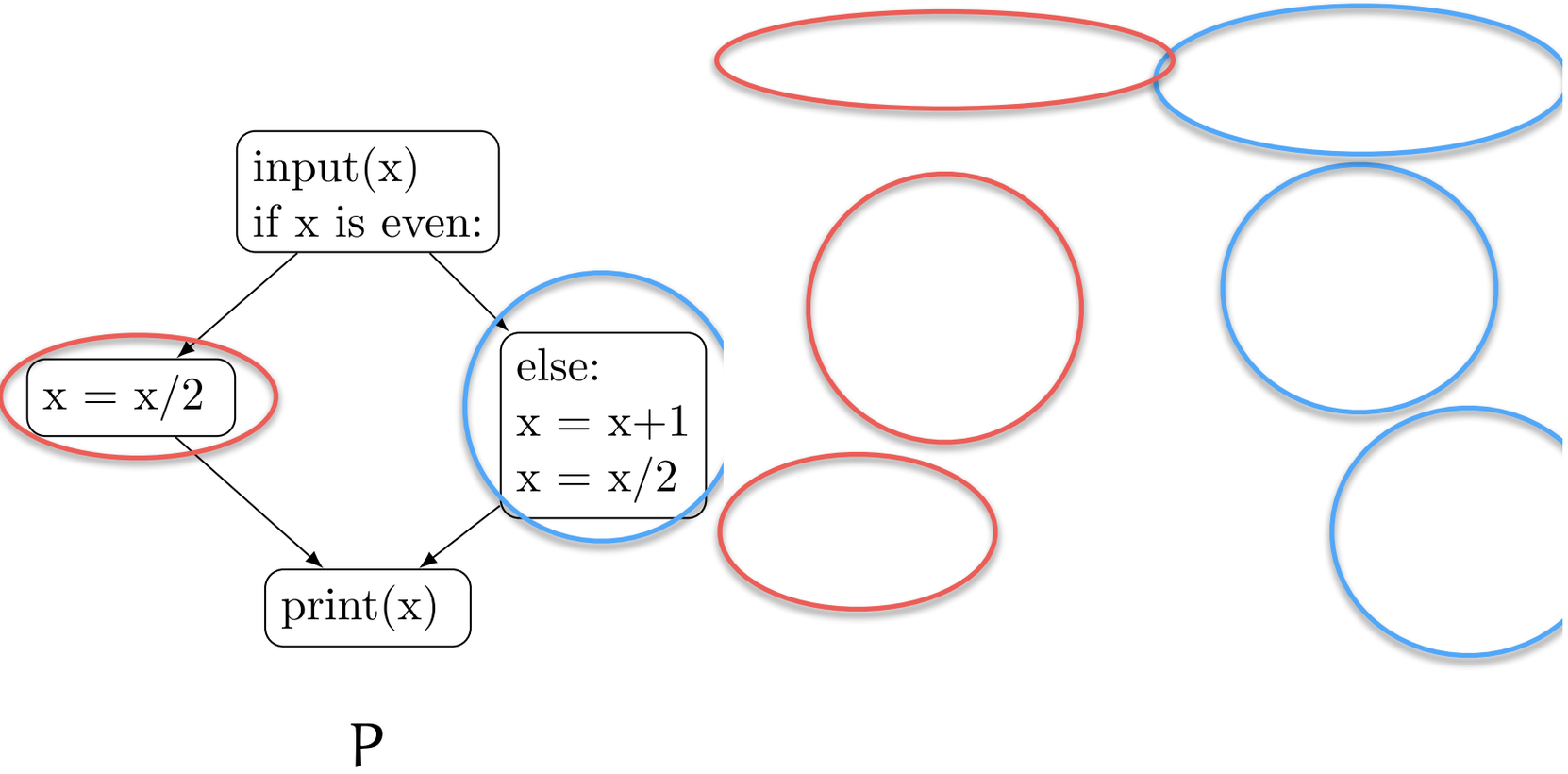
Dynamic analysis needs to observe at least n traces

$\mathcal{T}_n(P)$

```
input x;
n := select(N,x);
x := n*x;        Encoding
sum := 0;
while x < n*50
• X = [x, n * 50 − 1]
    sum := sum + x/n;
    x := x + n;
x:= x/n;         Decoding
```

Dynamic CFG Obfuscation

Diversification



P

Open Issues

➡ Properties of set of traces, other properties? Topological characterisation wrt to the kind of property being analysed

➡Model validation (ORAM, fuzzers, input generator and recognisers,…)

➡Potentiality and limits of code obfuscation for dynamic analysis

➡ Extend the model with measure of likelihodd of the inputs (probability distribution over the input)

THANKS

I hope I haven't annoyed you too much !!!