The Potential of Sampling for Dynamic Analysis

Joseph L. Greathouse Todd Austin

Advanced Computer Architecture Laboratory
University of Michigan

UNIVERSITY OF MICHIGAN

PLAS, San Jose, California
June 5, 2011

Dynamic Security Analysis

- Finds flaws in programs as they run
  - Makes programs more robust
- Insert checks around instructions

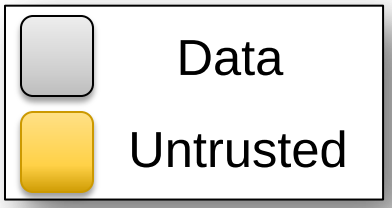
```
y = x->data;
*w += y;
z = 75/y;
```

Dynamic Security Analysis

- Finds flaws in programs as they run
  - Makes programs more robust
- Insert checks around instructions

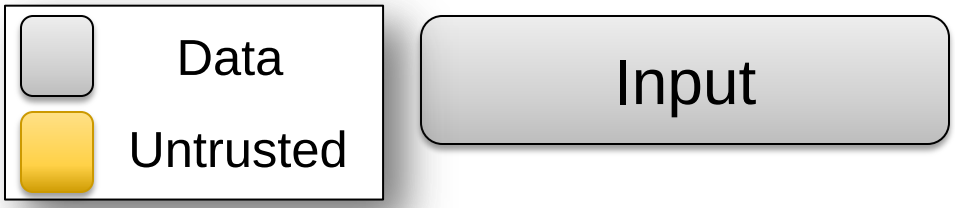
```
check(x!=NULL);
y = x->data;
check(w!=NULL);
*w += y;
check(y!=0);
z = 75/y;
```

Dynamic Information Flow Tracking

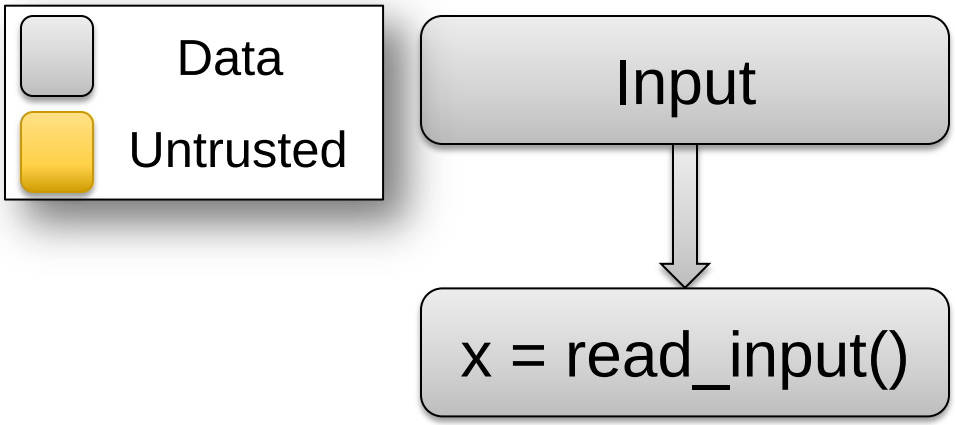


1. Inputs are untrusted
2. Propagate untrusted status while executing
3. Check trustedness for safety

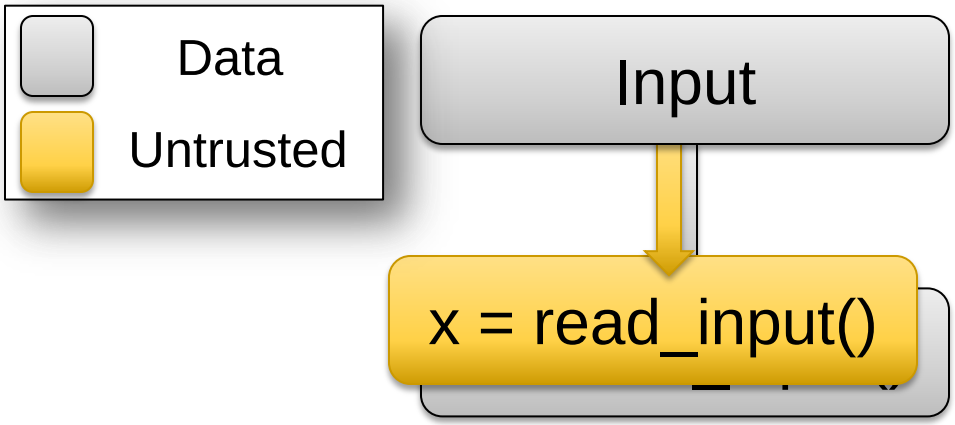
Dynamic Information Flow Tracking



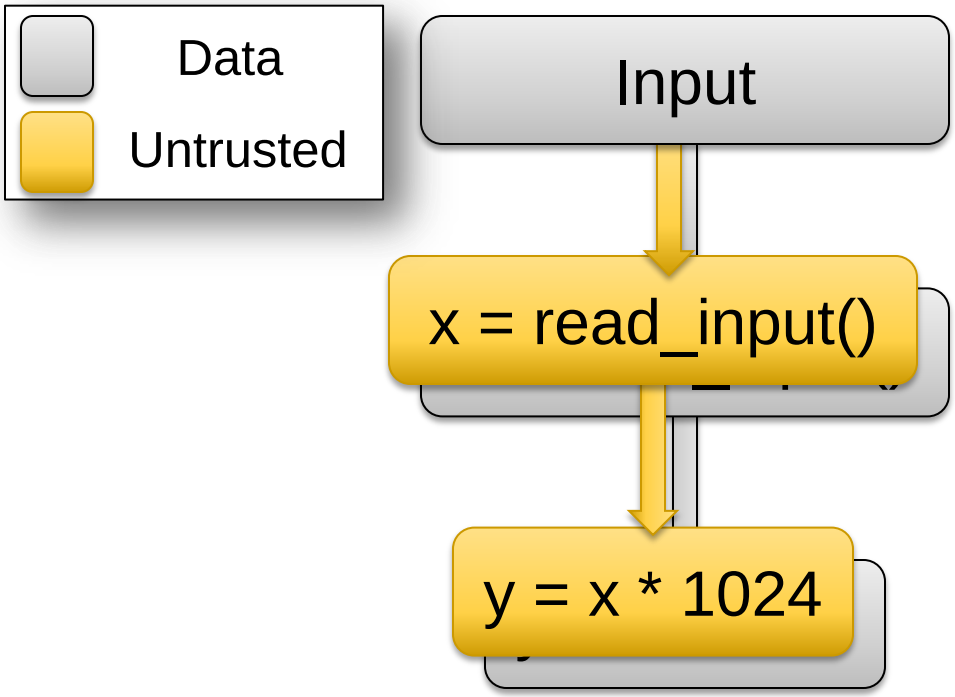
Dynamic Information Flow Tracking



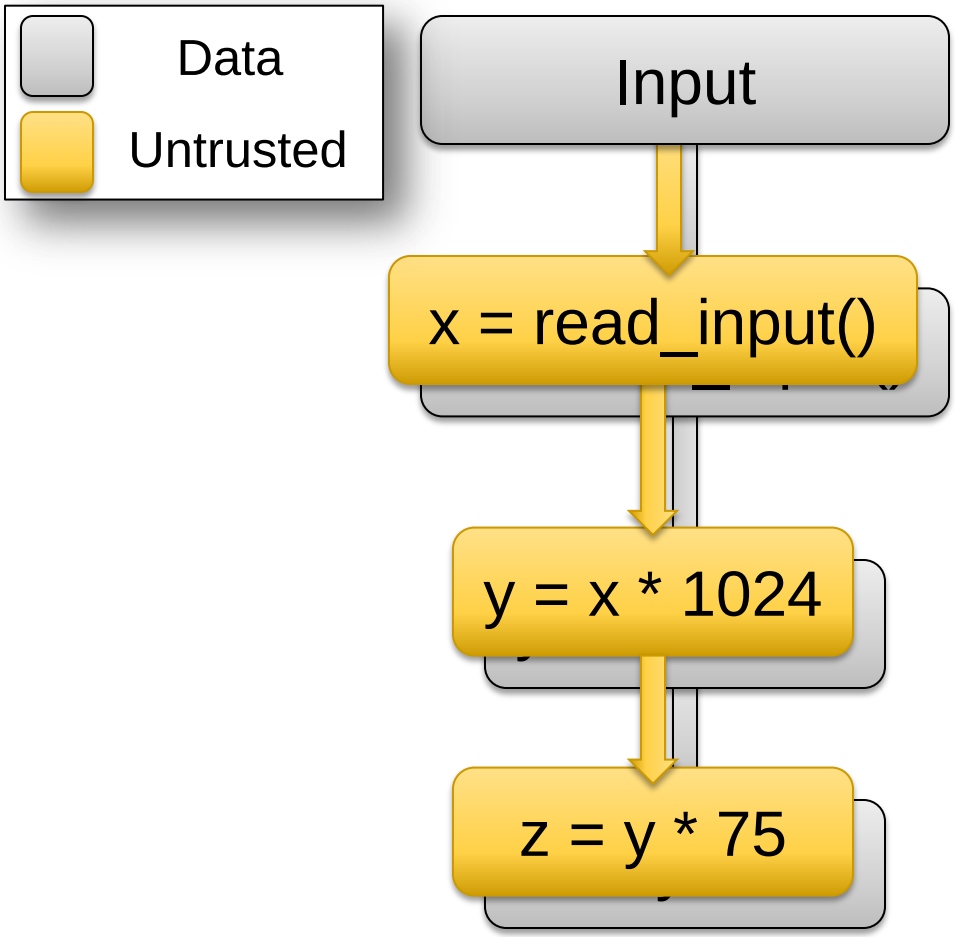
Dynamic Information Flow Tracking



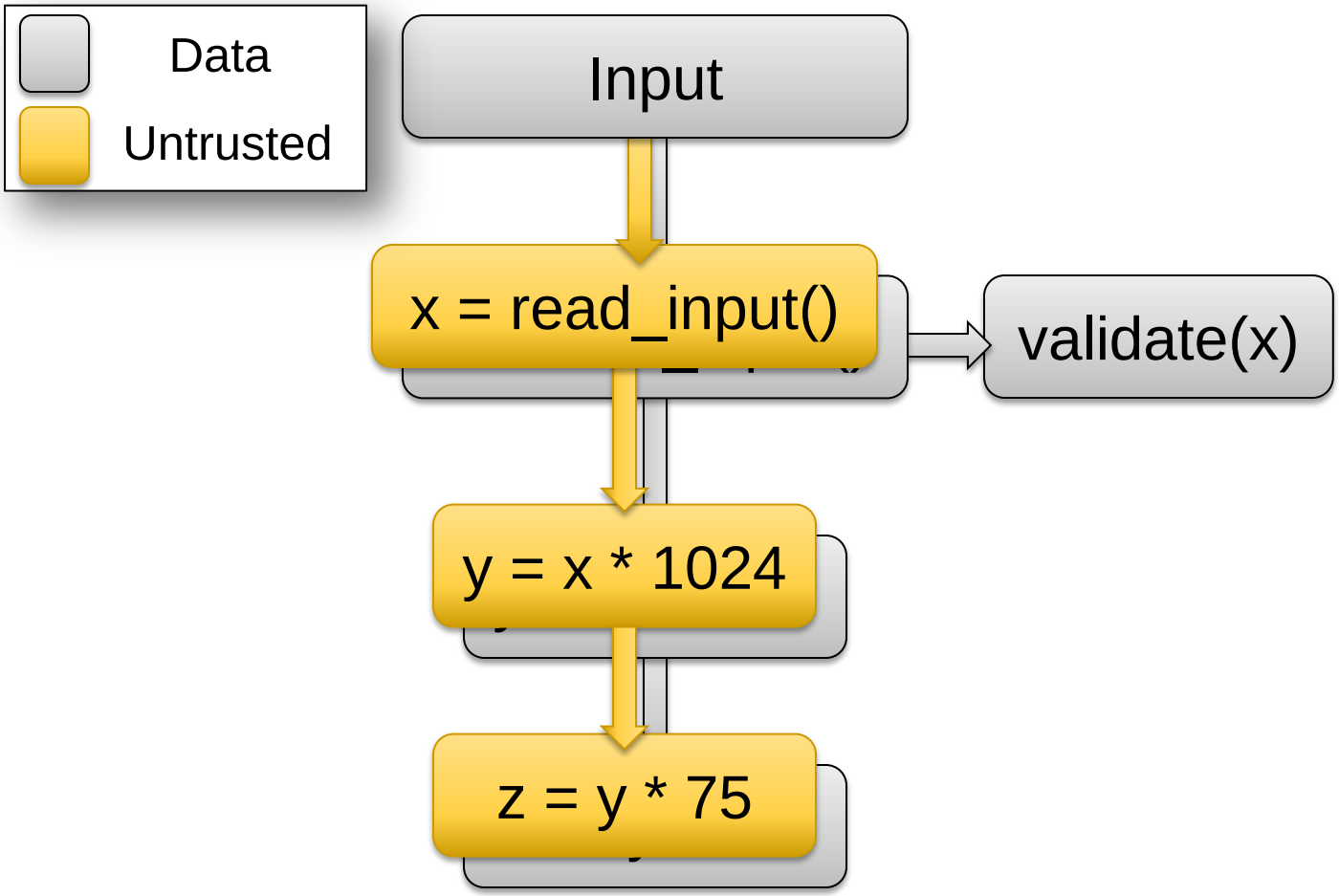
Dynamic Information Flow Tracking



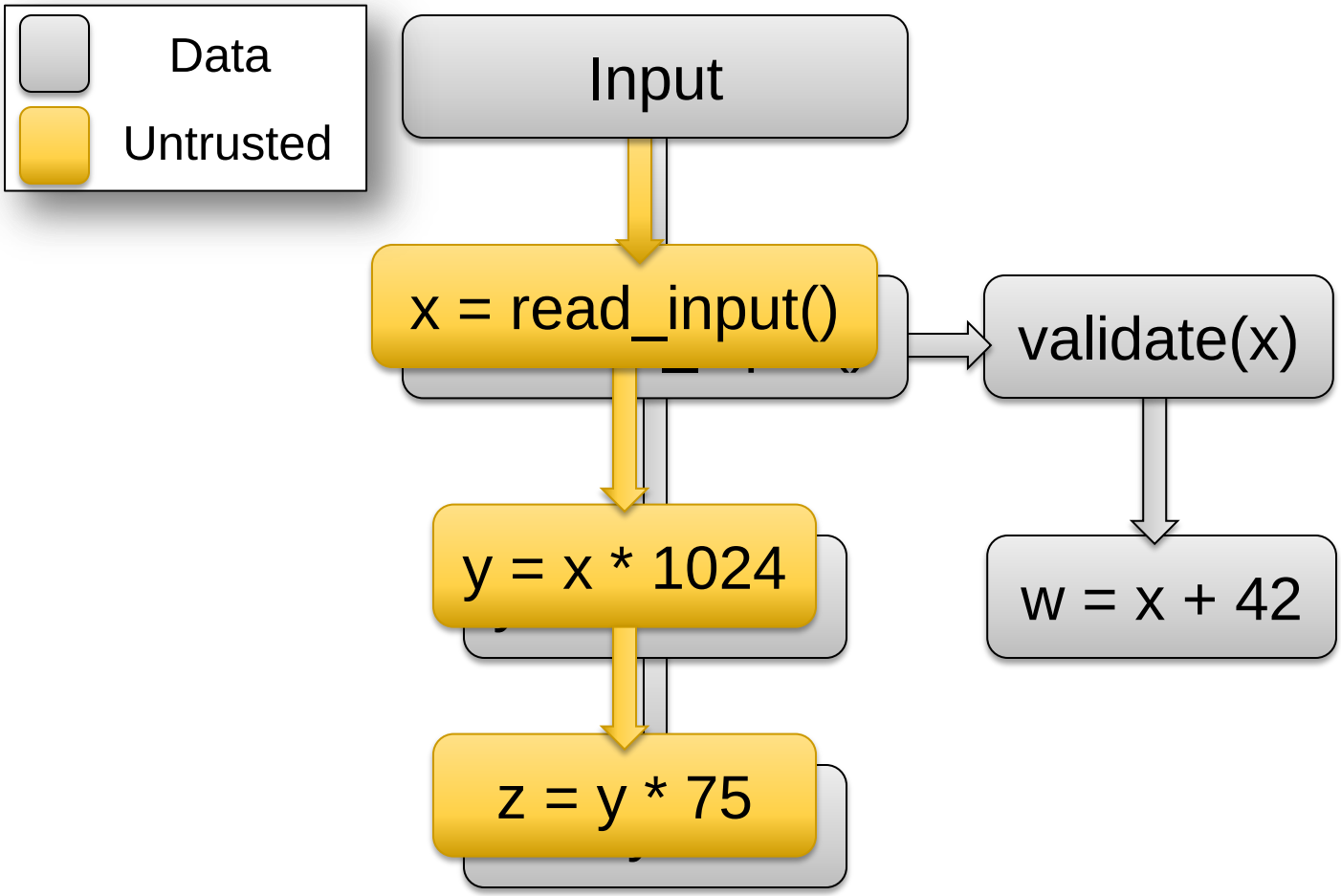
Dynamic Information Flow Tracking



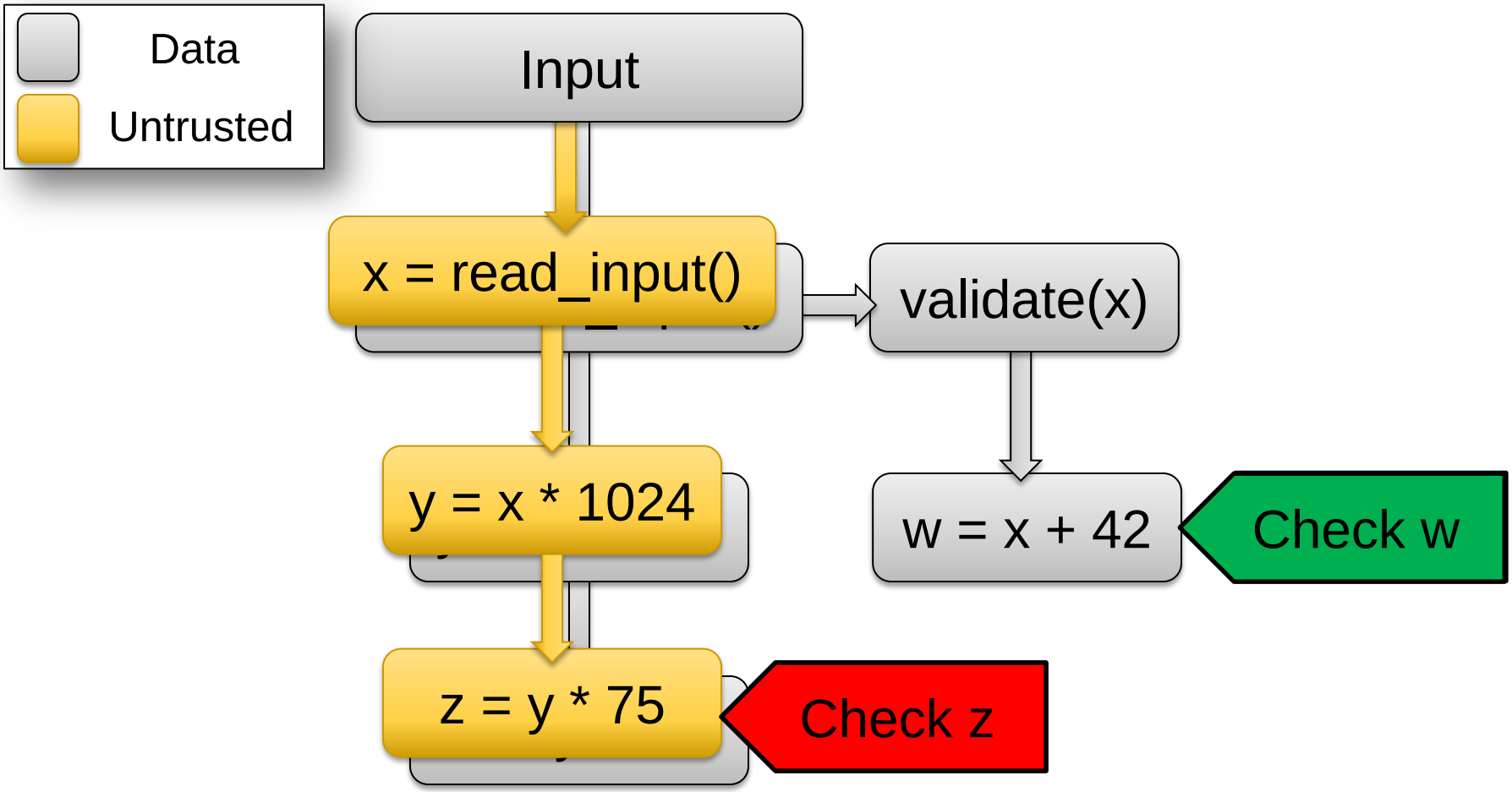
Dynamic Information Flow Tracking



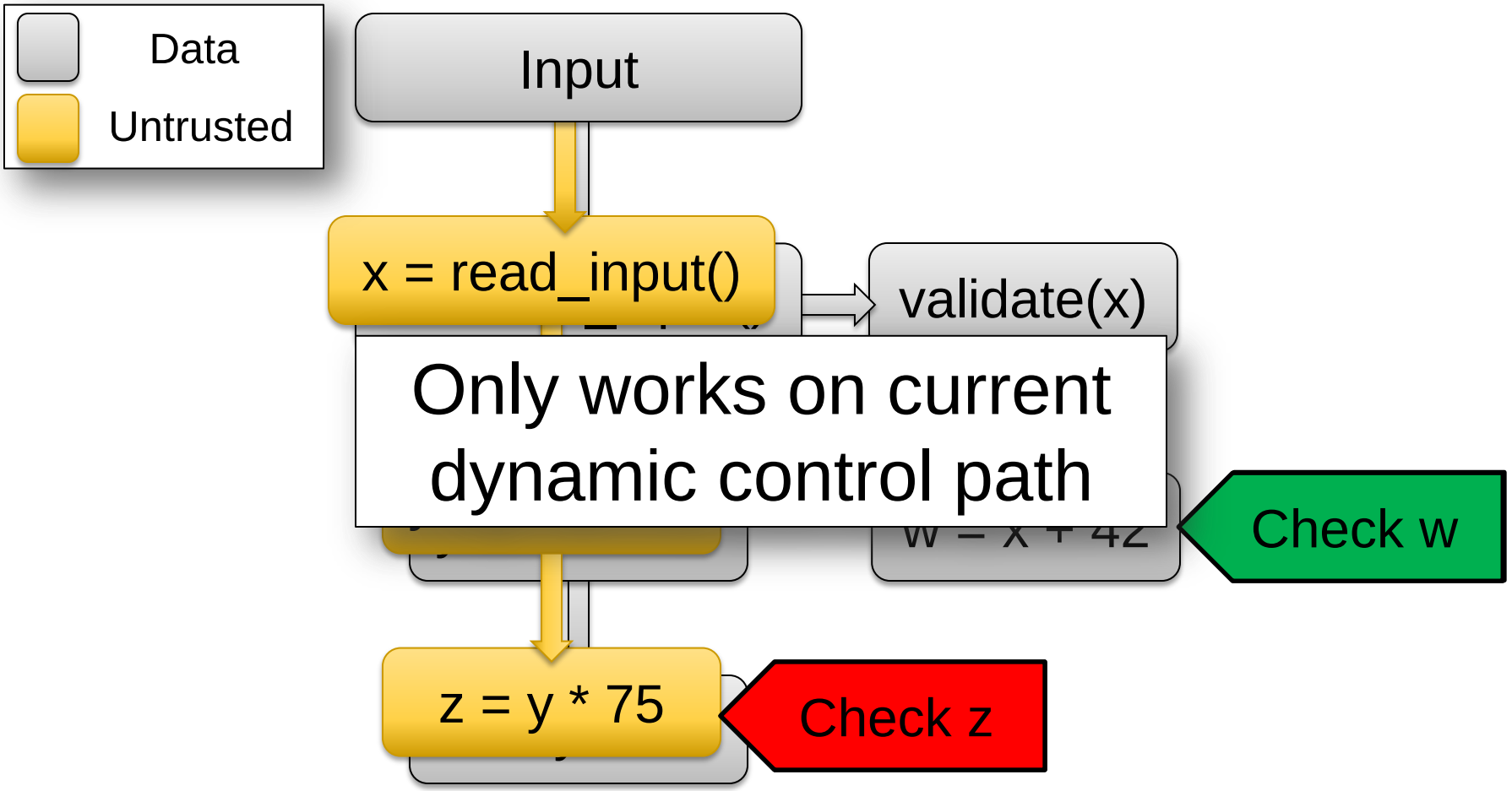
Dynamic Information Flow Tracking



Dynamic Information Flow Tracking



Dynamic Information Flow Tracking



Problem: Dynamic Analyses are Slow

Assertion Checking

2x

Race Detection

100x

DIFT

150x

Symbolic Execution

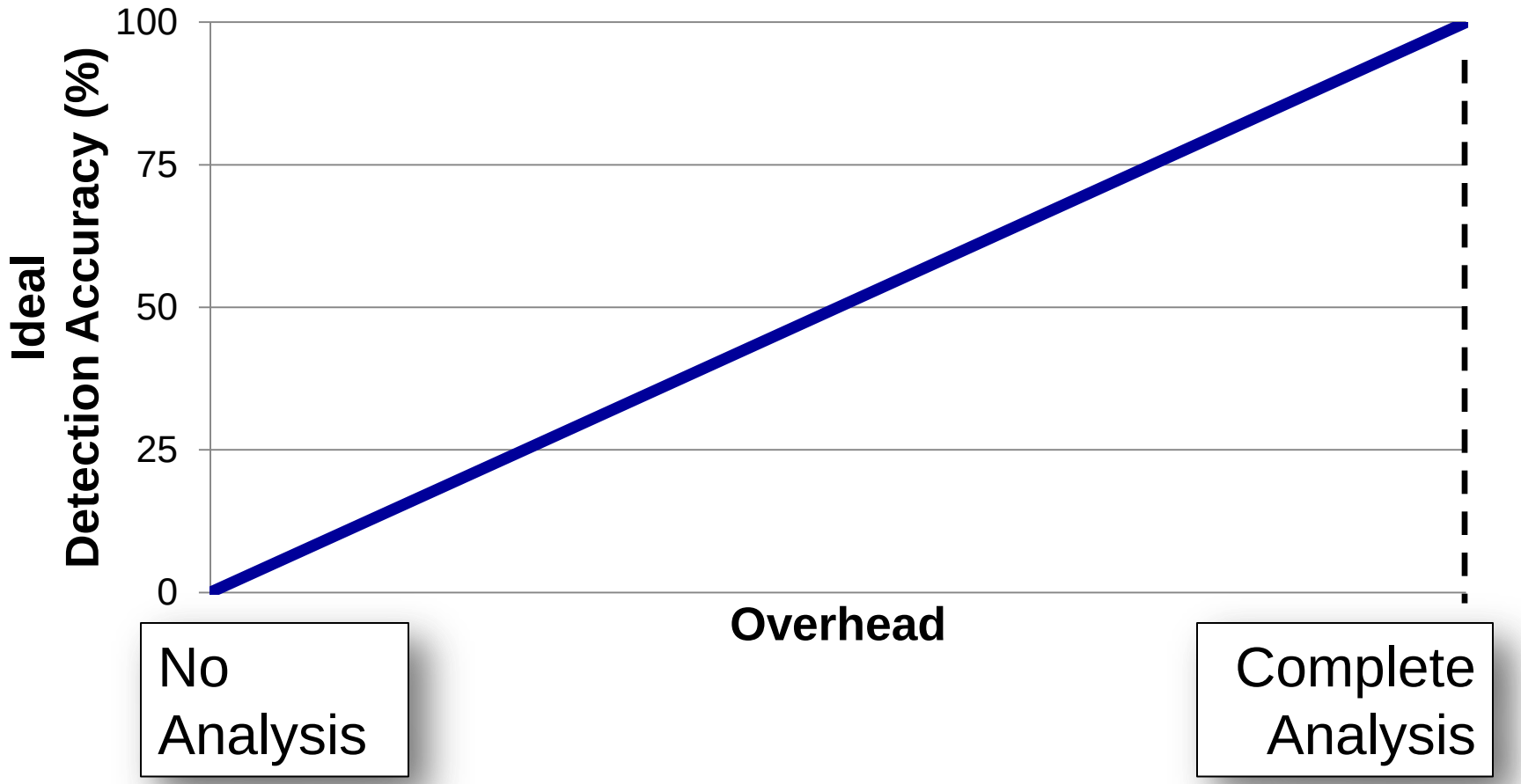
200x

Atomicity Checking

400x

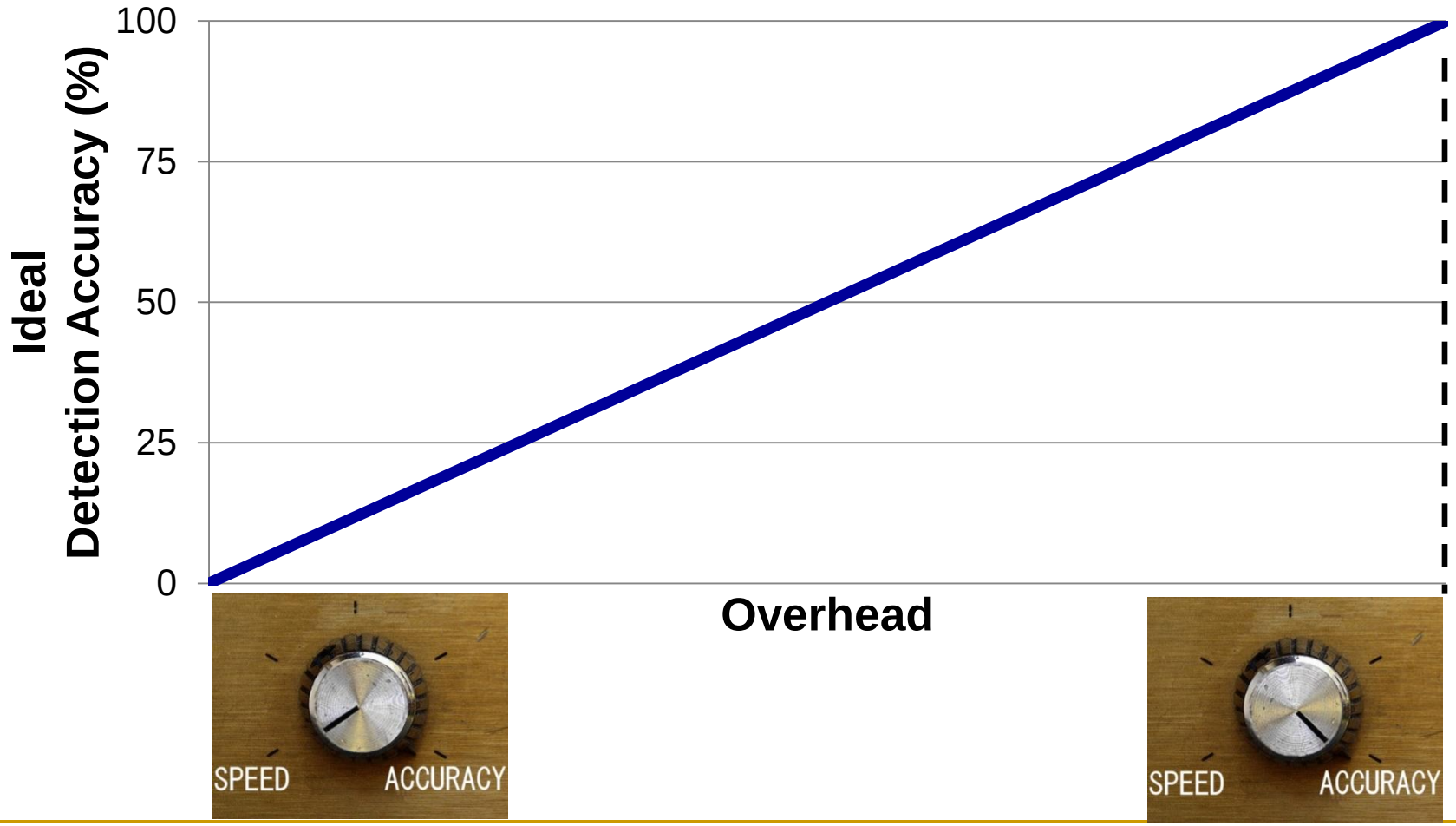
Dynamic Analysis Sampling

- Lower overheads by skipping some analyses

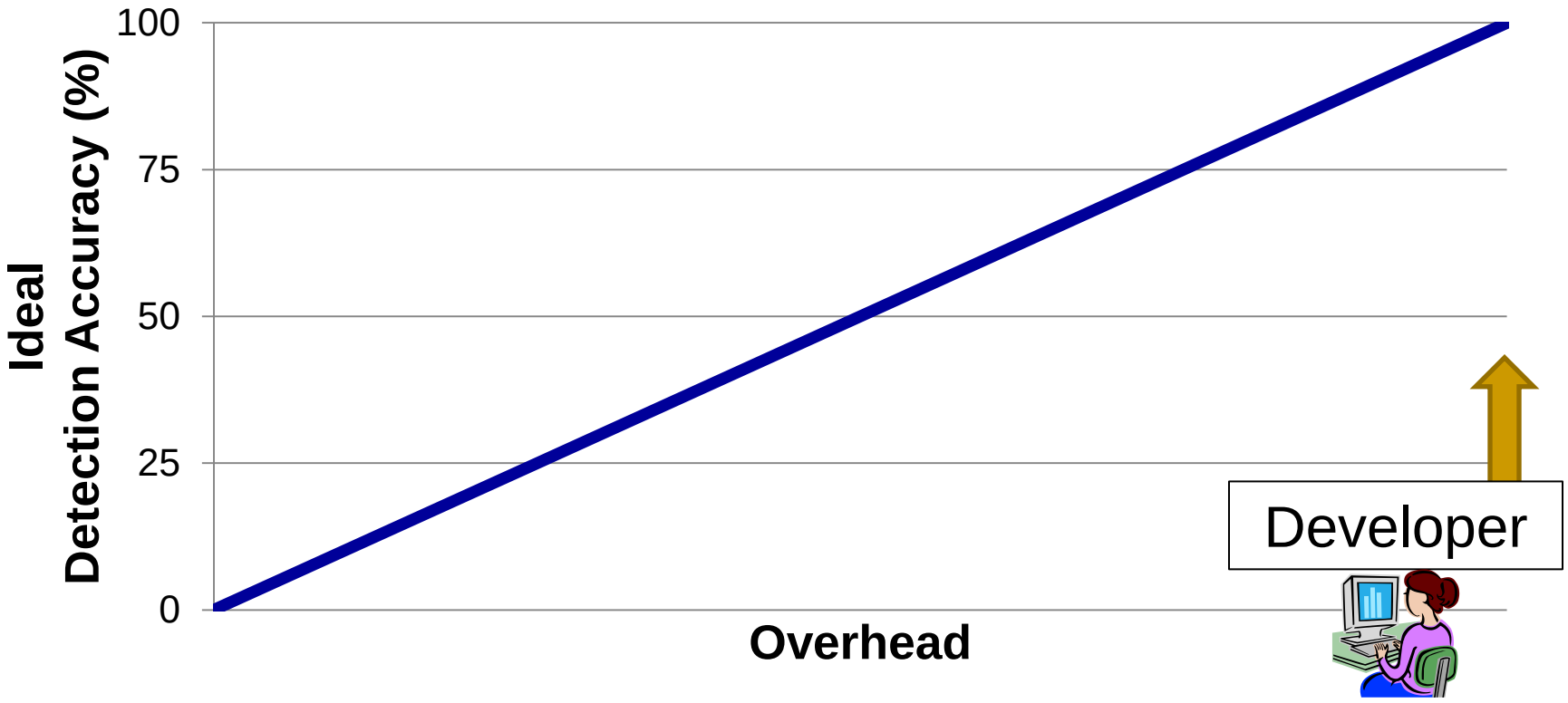


Dynamic Analysis Sampling

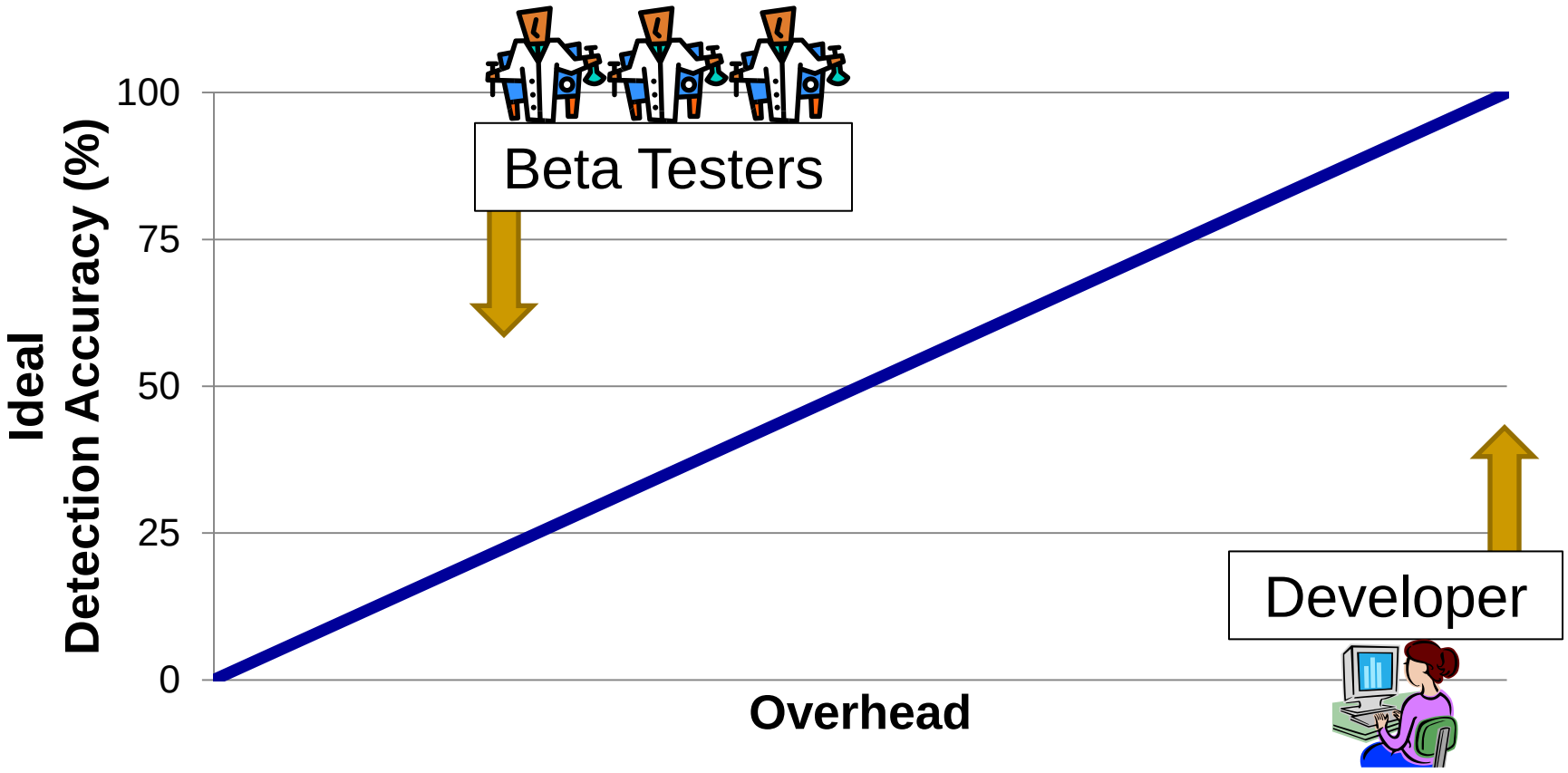
- Lower overheads by skipping some analyses



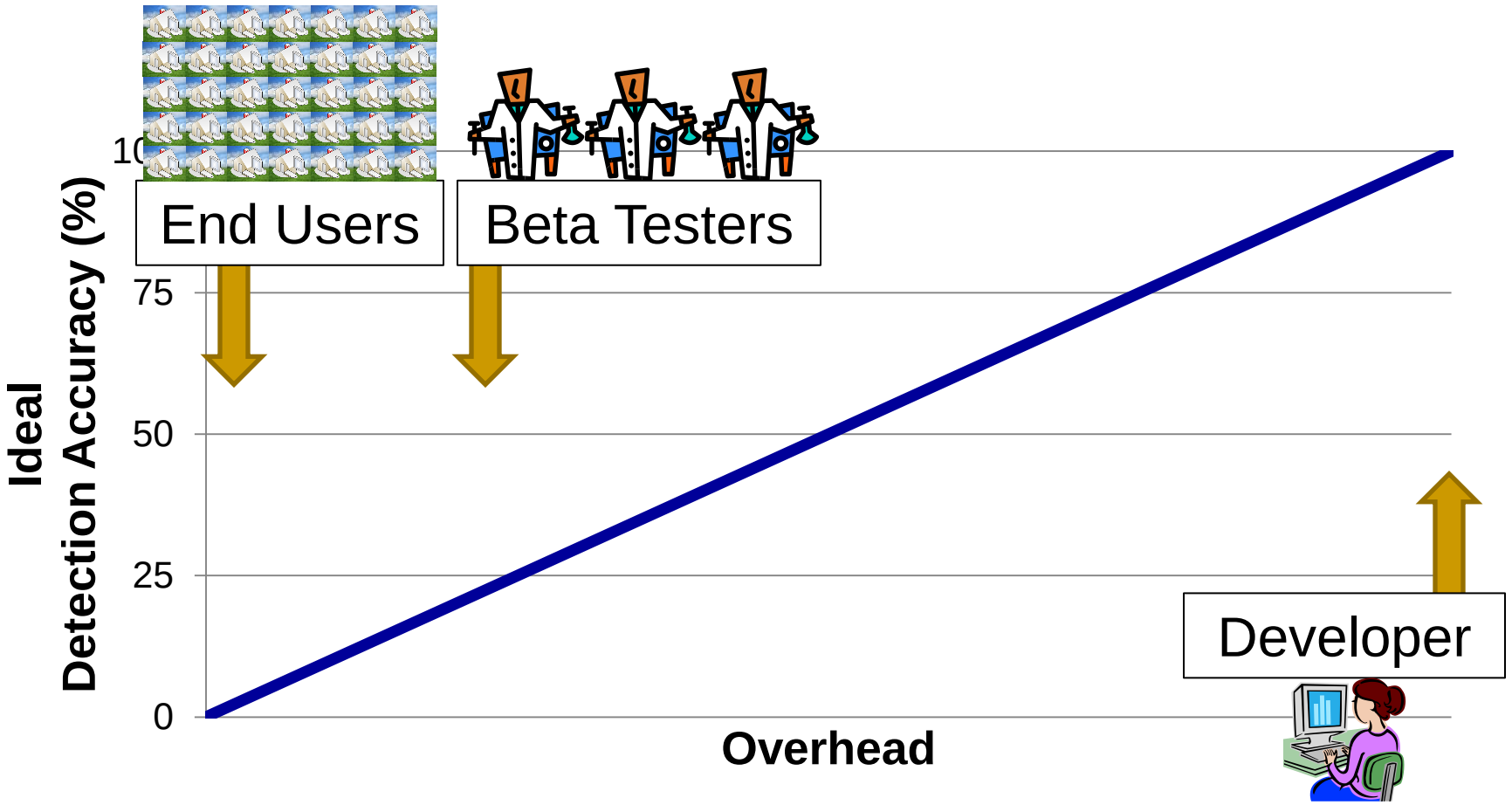
Sampling Allows Distribution



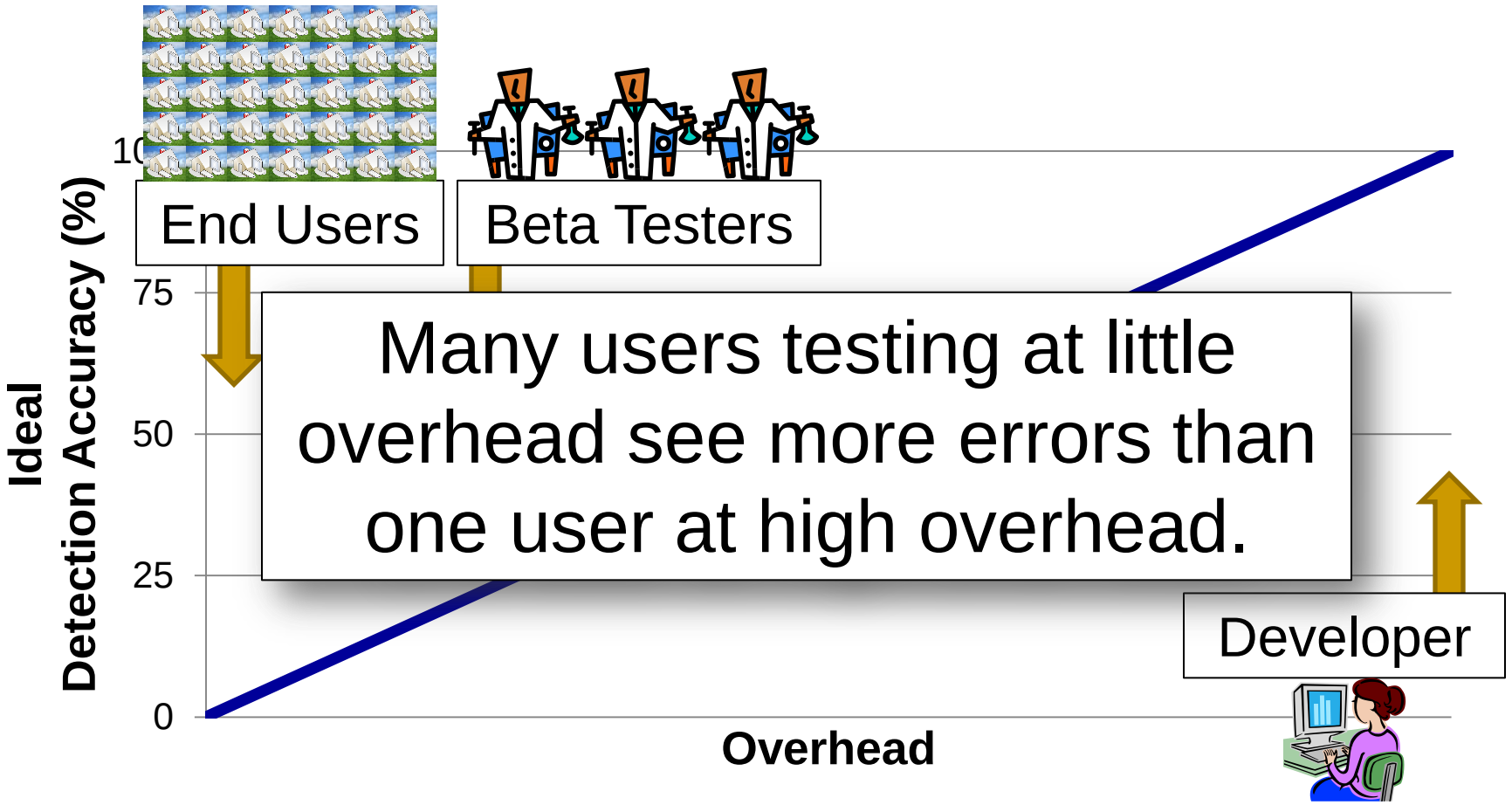
Sampling Allows Distribution



Sampling Allows Distribution



Sampling Allows Distribution



Sampling Assertion Checking

- Perform a random subset of checks

Static Code

```
check(x!=NULL);
y = x->data;
check(w!=NULL);
*w += y;
check(y!=0);
z = 75/y;
```

Sampling Assertion Checking

- Perform a random subset of checks

Static Code

```
check(x!=NULL);
y = x->data;
check(w!=NULL);
*w += y;
check(y!=0);
z = 75/y;
```

Sampling Assertion Checking

- Perform a random subset of checks

Dynamic #1	Dynamic #2	Dynamic #3
check(x!=NULL);		
y = x->data;	y = x->data;	y = x->data;
	check(w!=NULL);	
*w += y;	*w += y;	*w += y;
		check(y!=0);
z = 75/y;	z = 75/y;	z = 75/y;

Sampling Assertion Checking

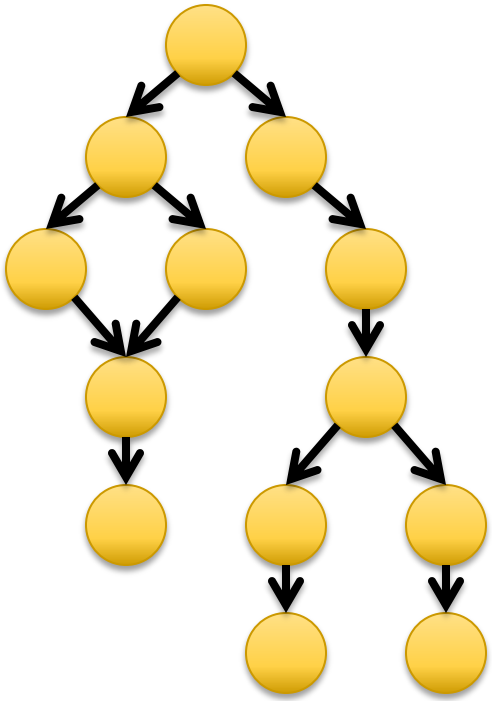
- Perform a random subset of checks

Dynamic #1	Dynamic #2	Dynamic #3
check(x!=NULL);		
y = x->data;	y = x->data;	y = x->data;
	check(w!=NULL);	
*w += y;	*w += y;	*w += y;
		check(y!=0);
z = 75/y;	z = 75/y;	z = 75/y;

- 1/1000 checks: ~3x slowdown→30% overhead

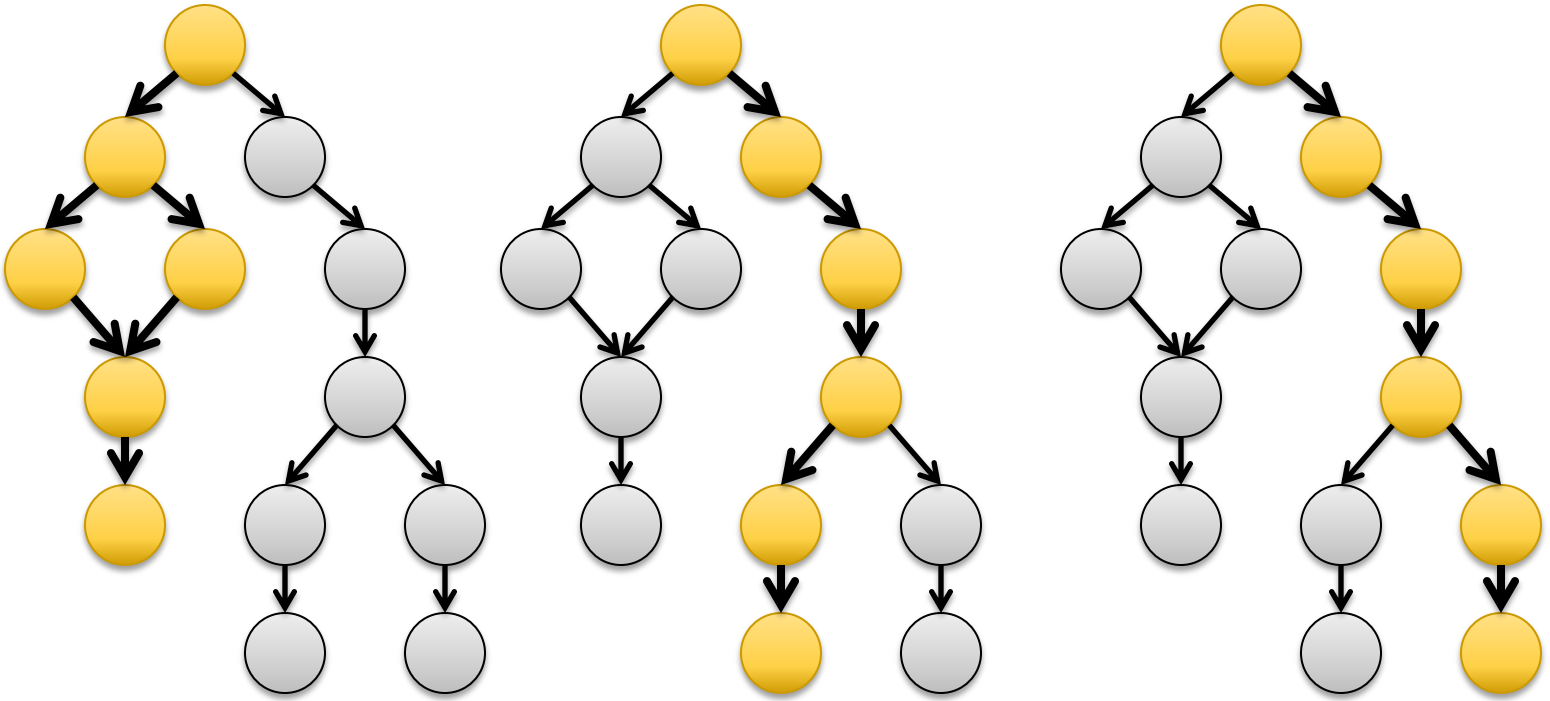
Sampling DIFT

- Must sample dataflows instead of instructions



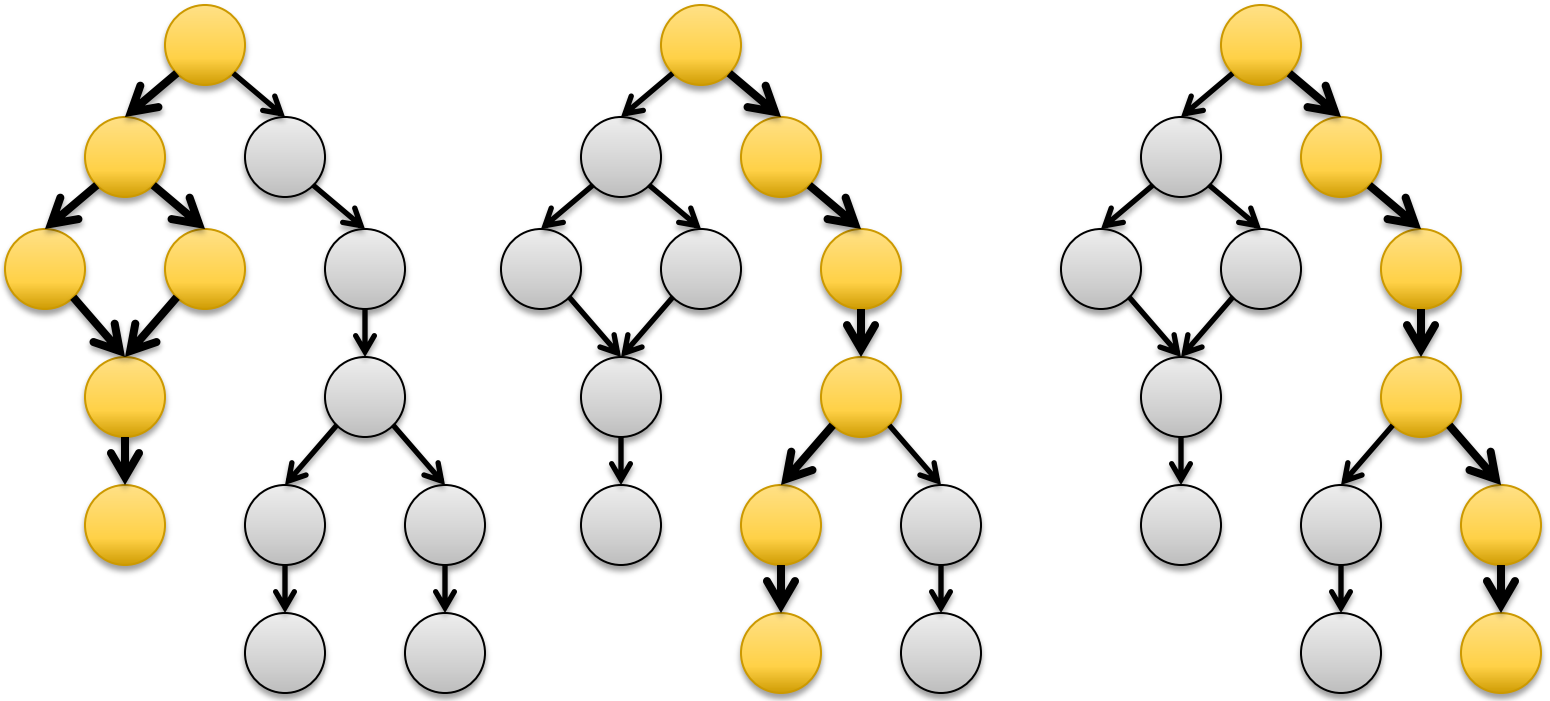
Sampling DIFT

- Must sample dataflows instead of instructions



Sampling DIFT

- Must sample dataflows instead of instructions



- 0.1-10% of faults: 10% overhead

Advanced Sampling Techniques

- Cold region Hypothesis
  - Sample “cold” code at higher rate
- New atomicity violation detection
  - Trace atomic regions and correlate crashes

Future Research Directions

- Sampling for more dynamic analyses
- New types of analyses because of sampling
- Studies on users slowdown acceptance

BACKUP SLIDES

Needed Engineering Efforts

- Push-button sampling mechanisms for Valgrind, DynamoRIO, Pin, etc.
- Libraries to integrate bug reporting into analysis tools
- Back-end infrastructure for handling distributed, sampled bug reports