MOAD

Modeling Observation-based Approximate Dependency

Seongmin Lee
KAIST

David Binkley
Loyola University Maryland

Robert Feldt
Chalmers University of Technology

Nicolas Gold
University College London

Shin Yoo
KAIST

Program Dependency Analysis

```
int main() {
    int sum = 0;
    int i = 1;
    while (i < 11) {
        sum = sum + i;
        i = i + 1;
    }
    printf(“%d\n”, sum);
    printf(“%d\n”, i);
}
```

Program

$$\forall e_1, e_2 \in E$$

Program elements

depends on

$$e_1 \longleftarrow e_2$$

Program Dependency Analysis

$$\forall e_1, e_2 \in E$$

Program elements

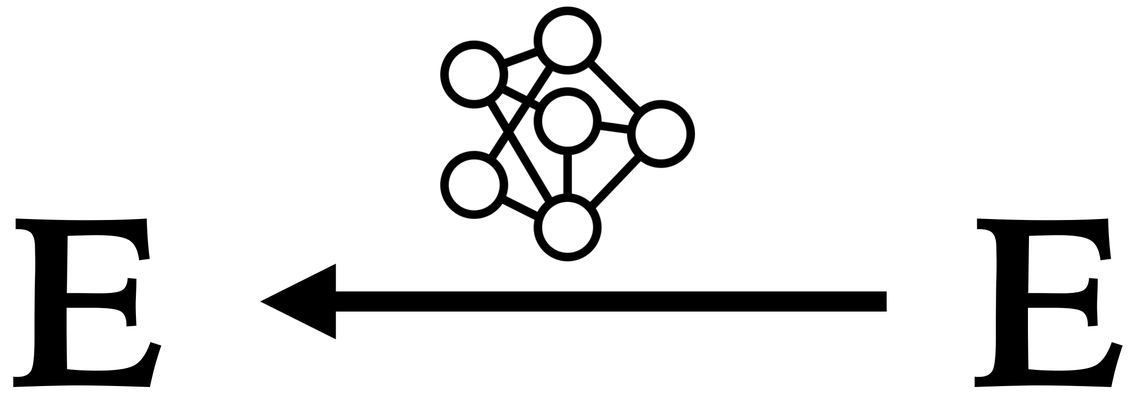
depends on

$$e_1 \longleftarrow e_2$$

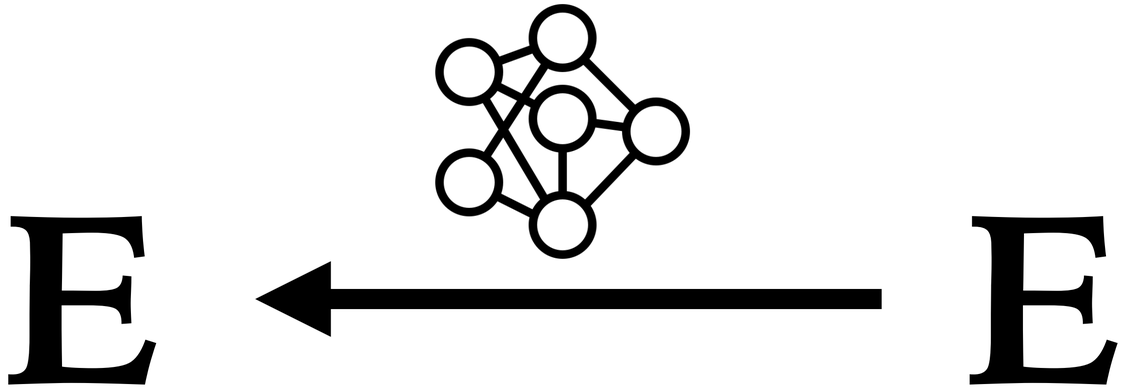
- Fault comprehension
  - *Hidden dependencies in program comprehension and change propagation*, Zhifeng Yu et al.
- Software testing
  - *Semantics guided regression test cost reduction,* Binkley et al.
- Software maintenance
  - *Using program slicing in software maintenance,* Gallagher et al.
- Security
  - *Platform-independent dynamic taint analysis for javascript*, Karim et al.
- Debugging
  - *Do programmers do change impact analysis in debugging?,* Jiang et al.

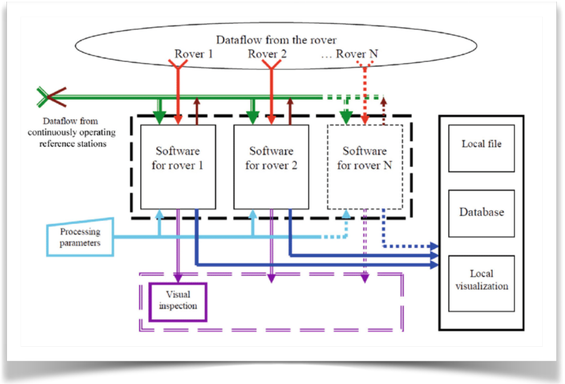
etc.

Static Analysis



Static Analysis



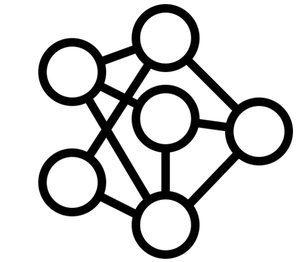


Large & complex system

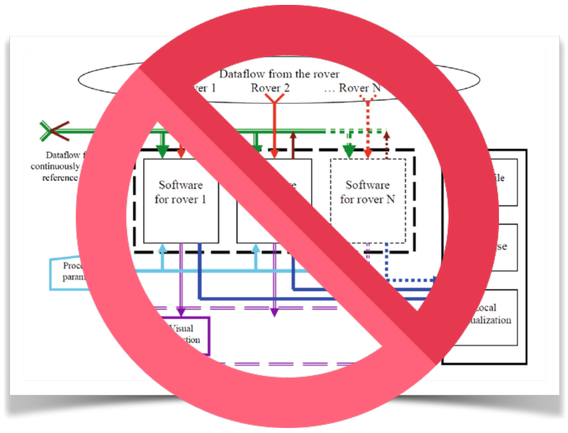
Working with external database

Multi-lingual program

Static Analysis



E ← E



Large & complex system

Working with external database

Multi-lingual program

Observation-based Slicing (ORBS)

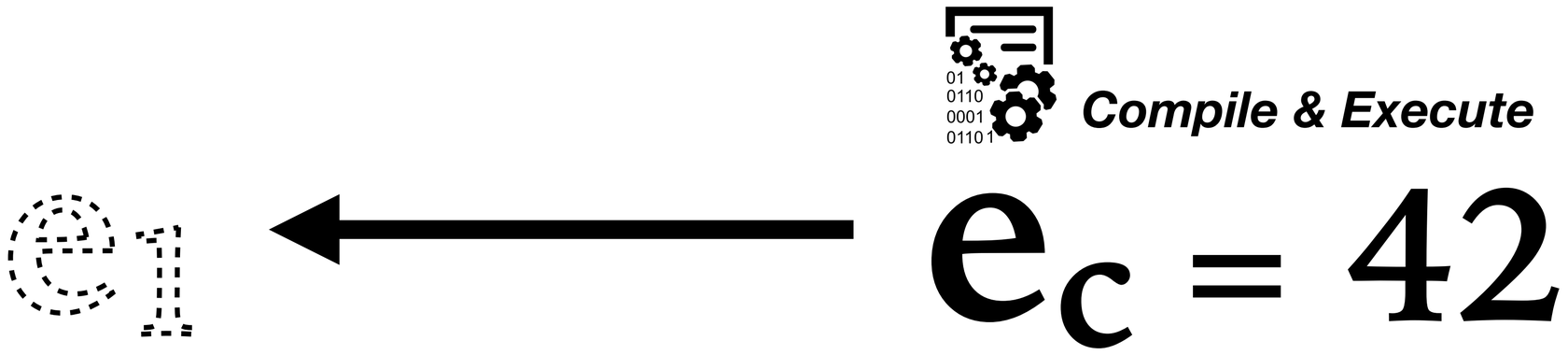
Observation-based Slicing (ORBS)

Compile & Execute

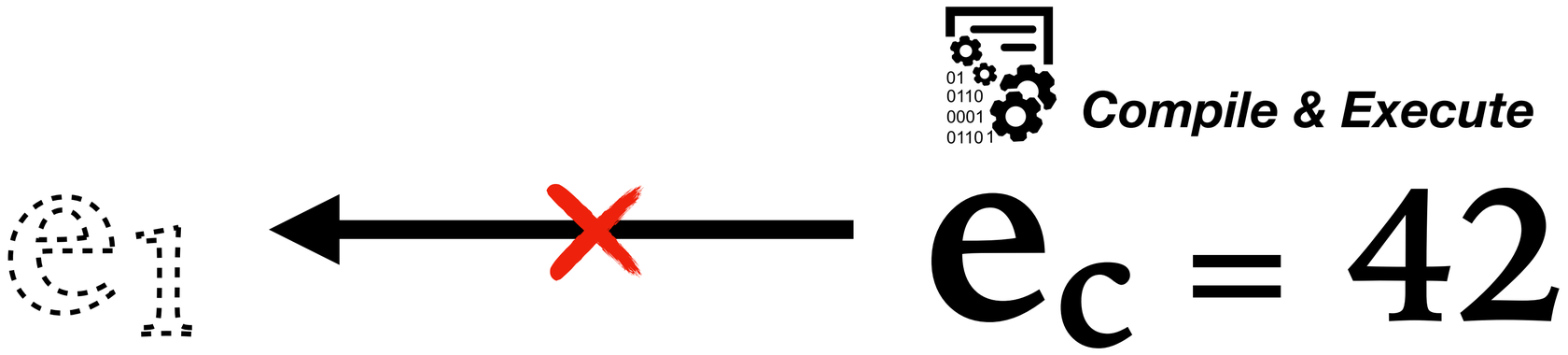
$$e_1 \longleftarrow e_c = 42$$

Observation-based Slicing (ORBS)

Observation-based Slicing (ORBS)

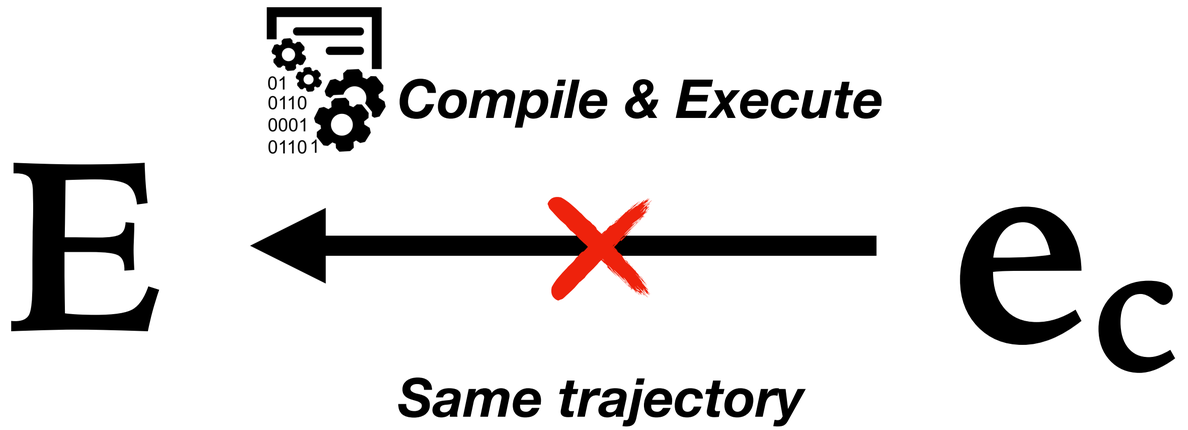


Observation-based Slicing (ORBS)

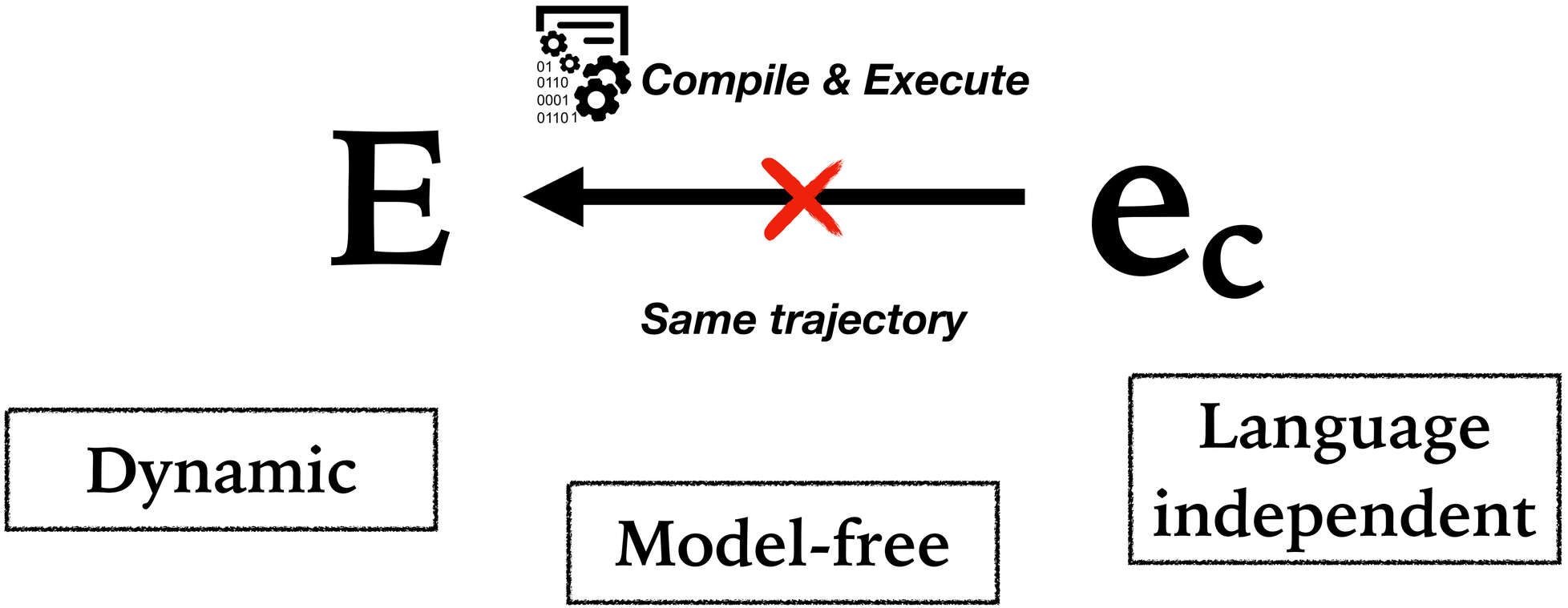


Observation-based Slicing (ORBS)

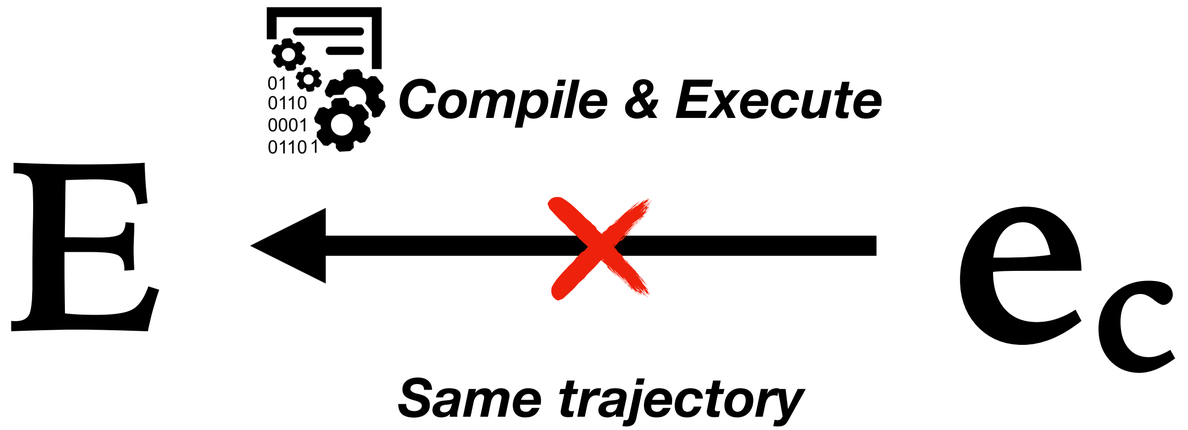
Observation-based Slicing (ORBS)



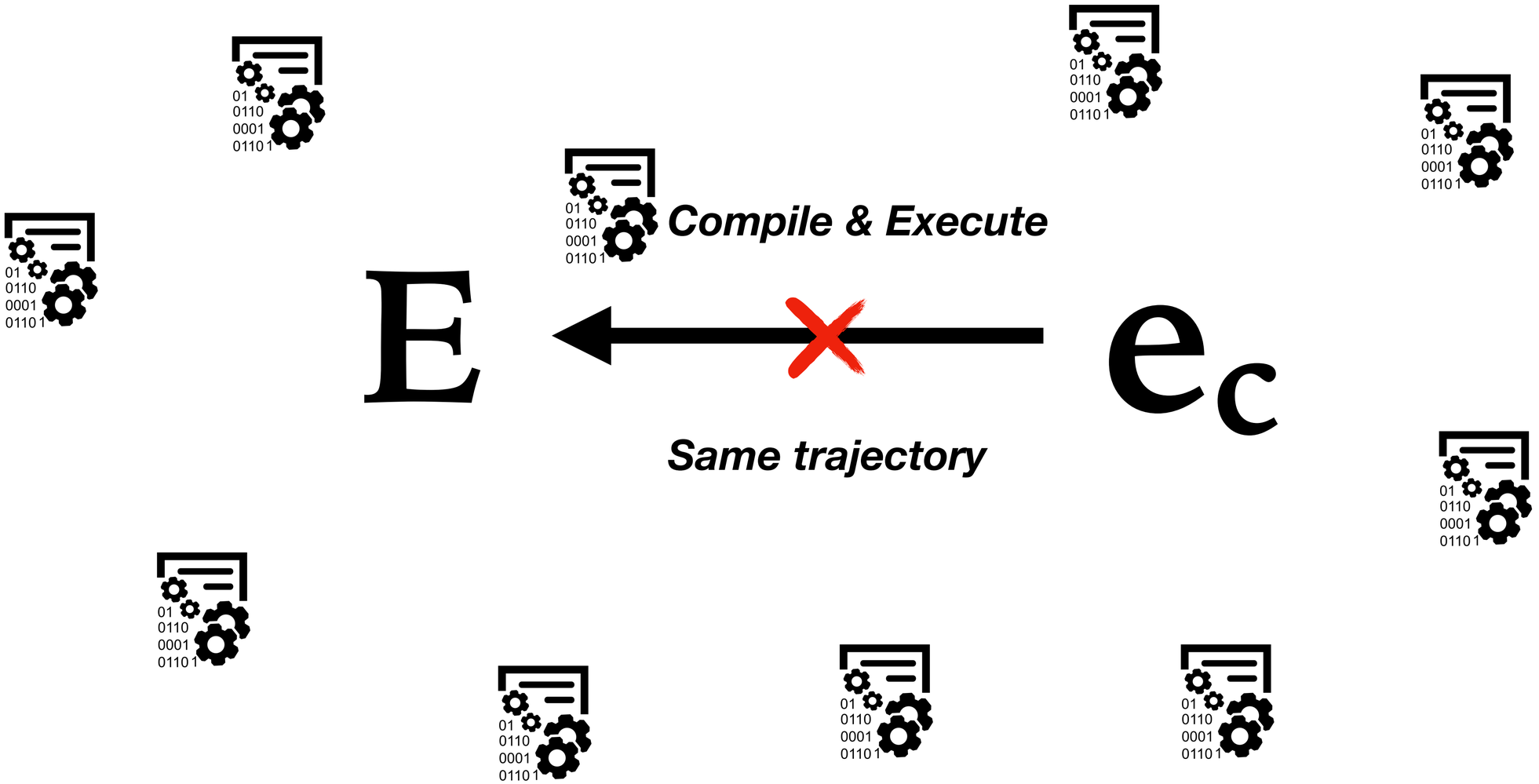
Observation-based Slicing (ORBS)



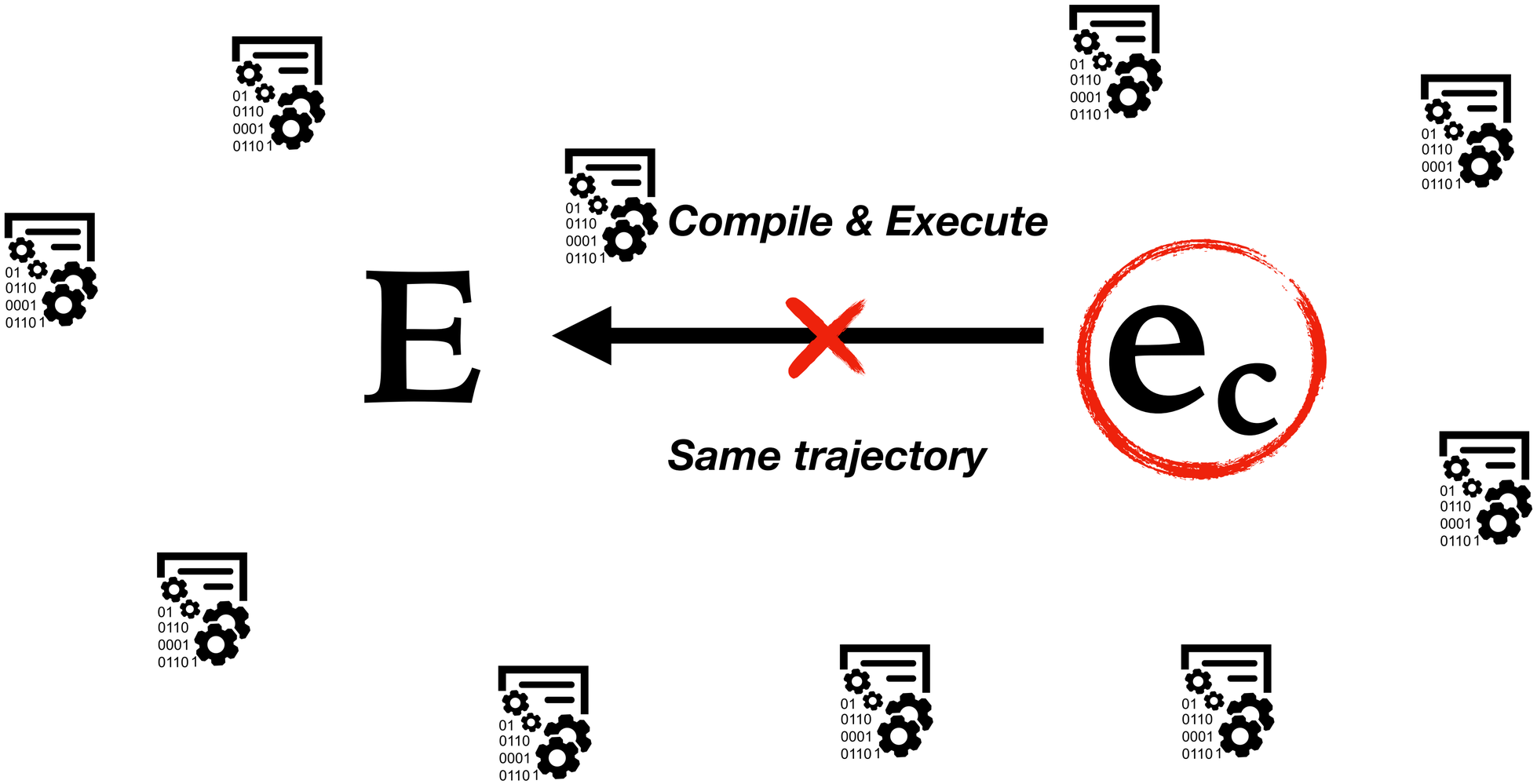
Observation-based Slicing (ORBS)



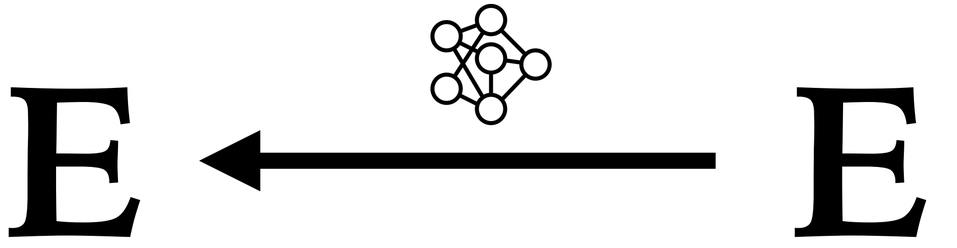
Observation-based Slicing (ORBS)



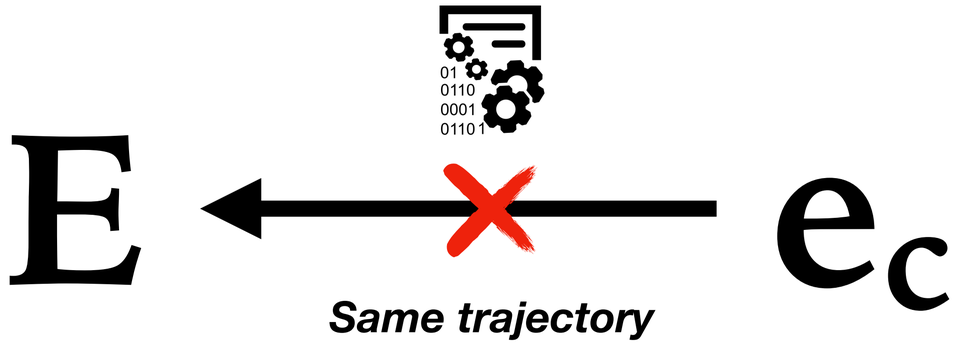
Observation-based Slicing (ORBS)



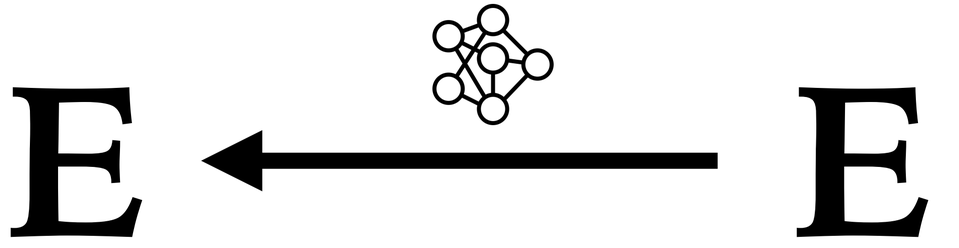
Static Analysis



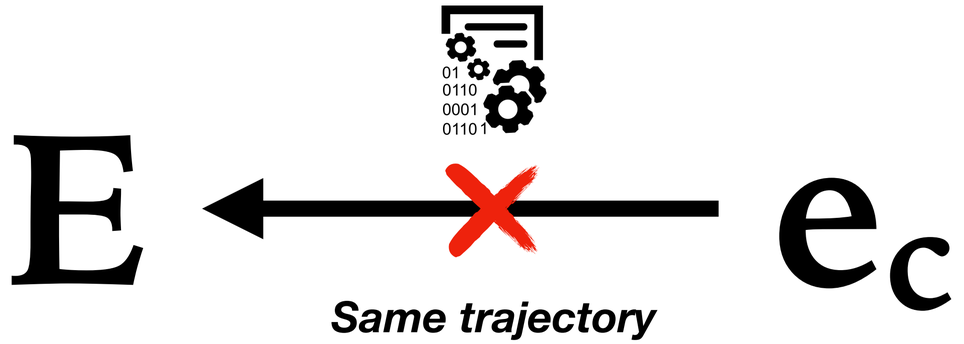
ORBS



Static Analysis



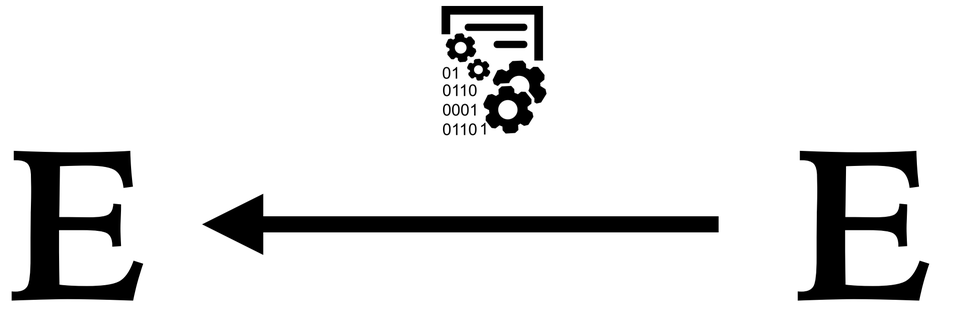
ORBS



Lightweight dynamic analysis

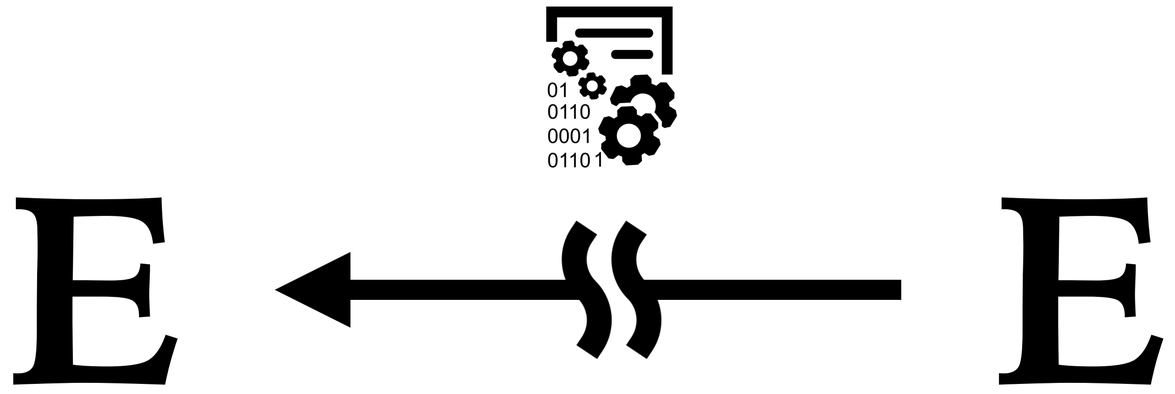
+

Modeling dependency



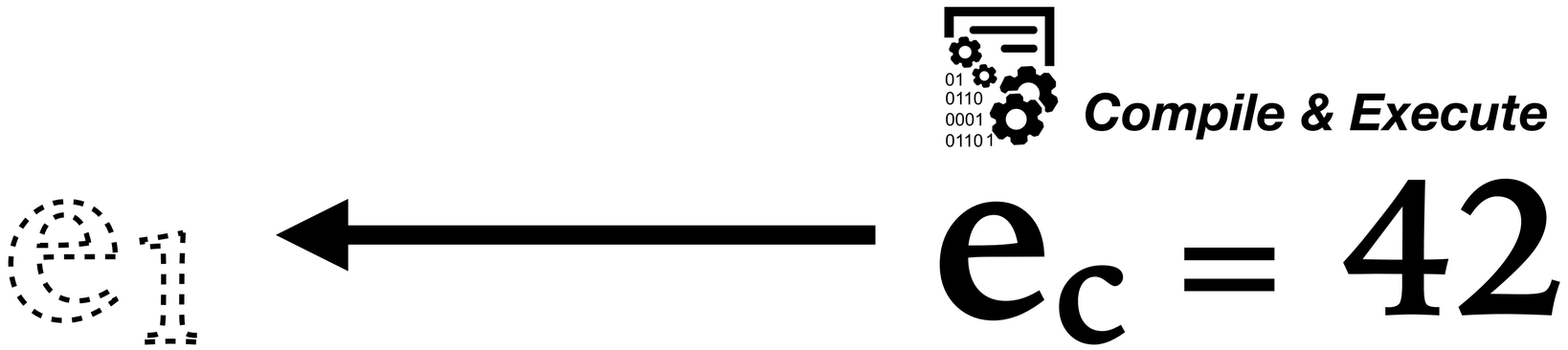
MOAD

Modeling Observation-based Approximate Dependency

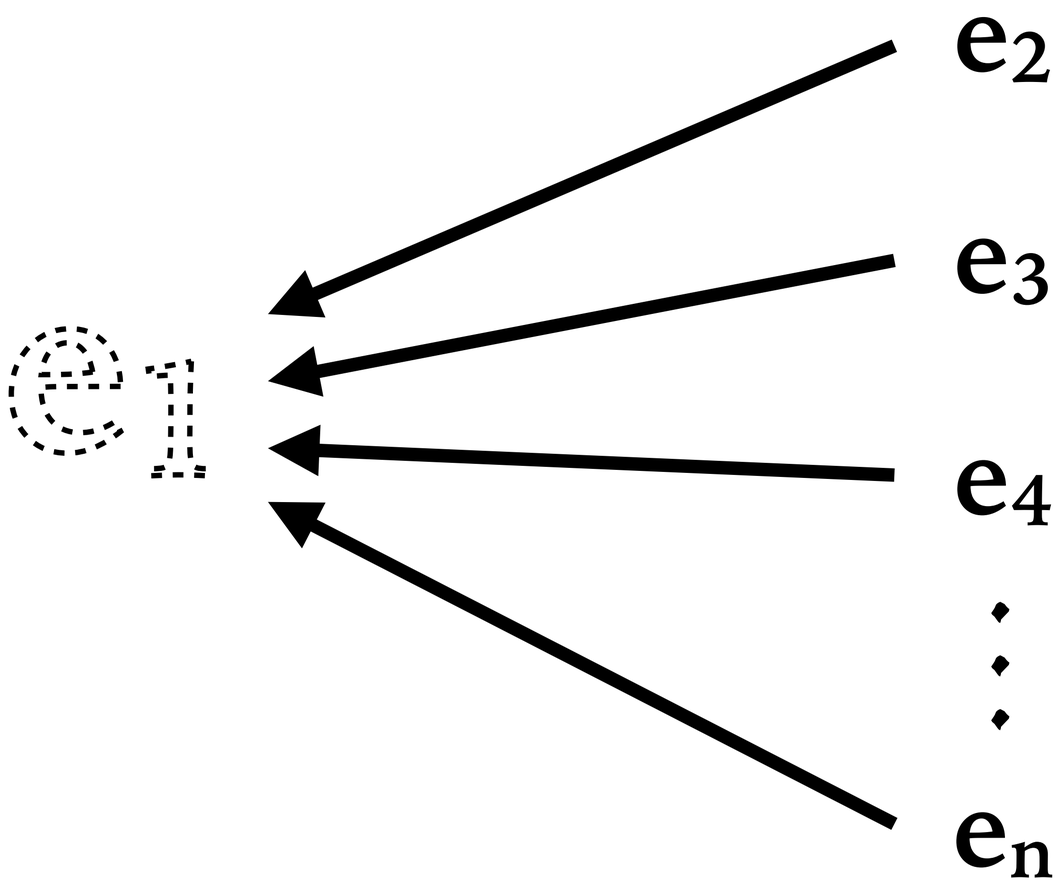


MOAD

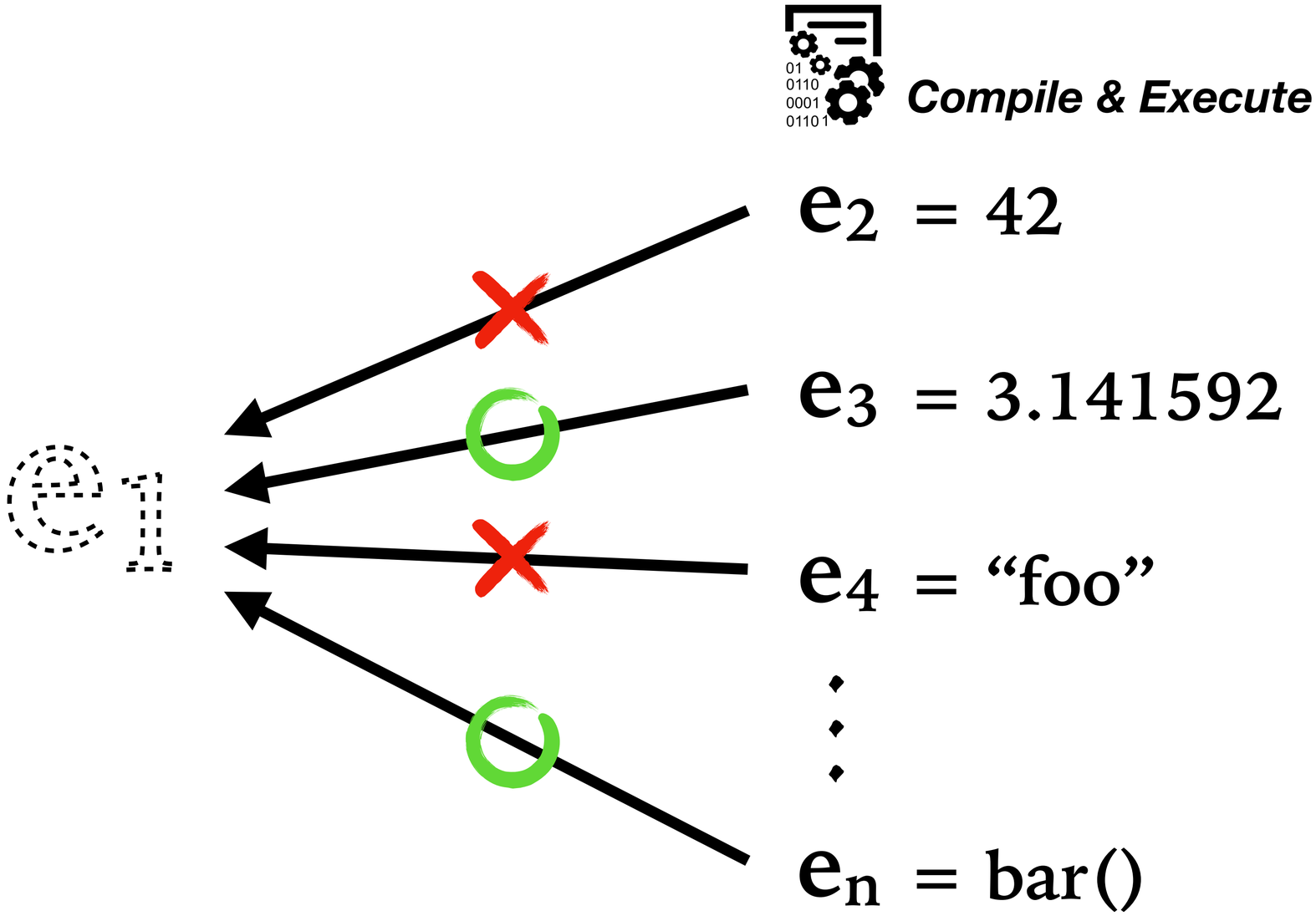
MOAD



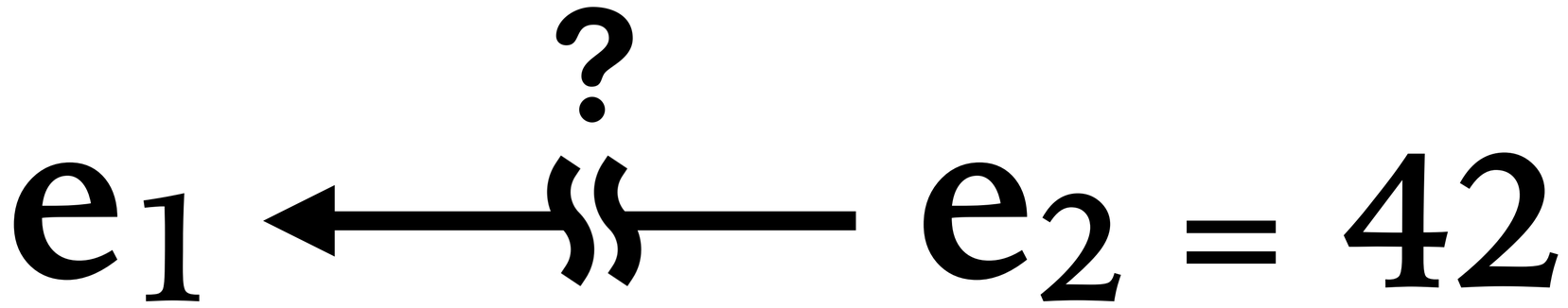
MOAD



MOAD

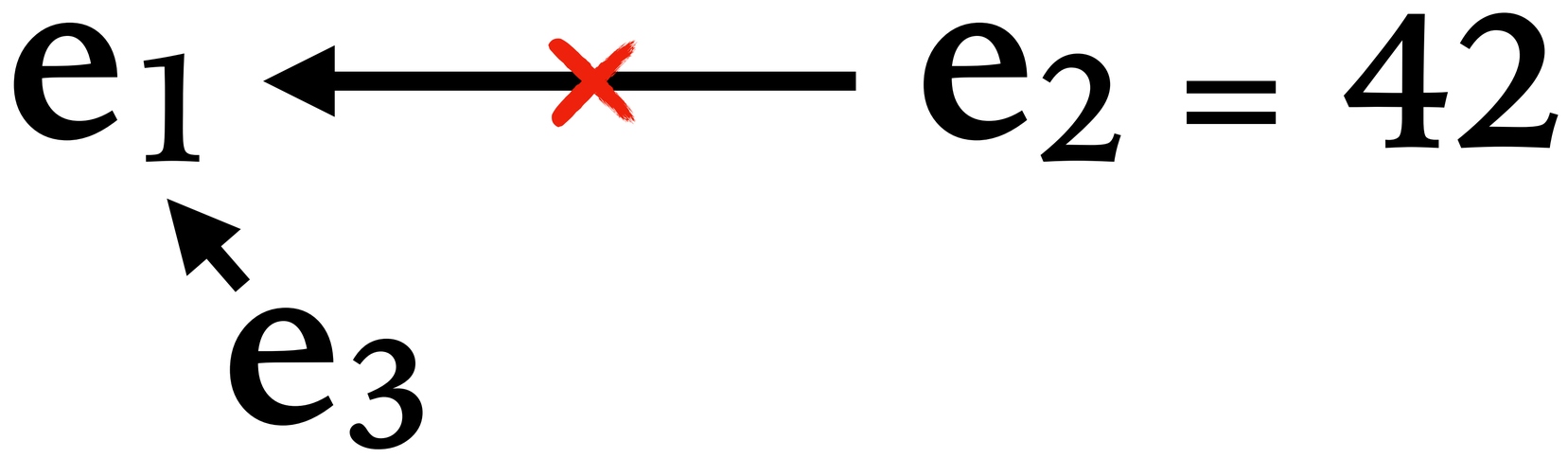


MOAD

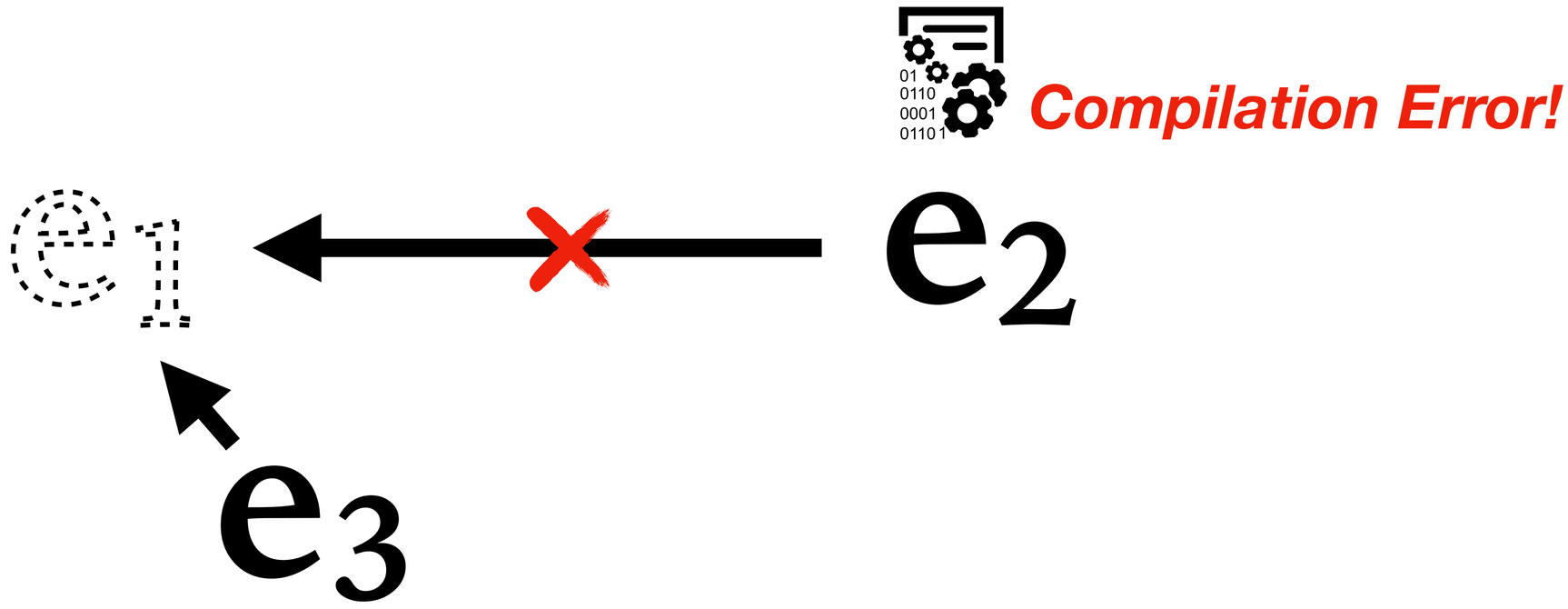


MOAD

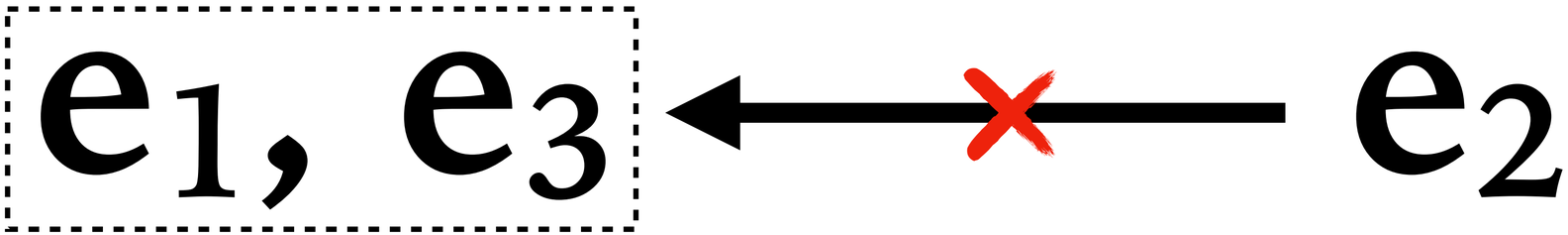
MOAD



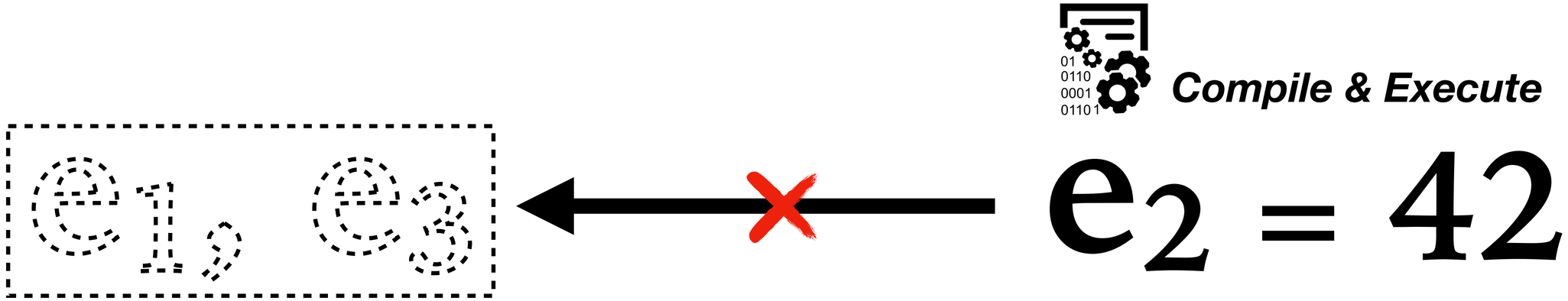
MOAD



MOAD

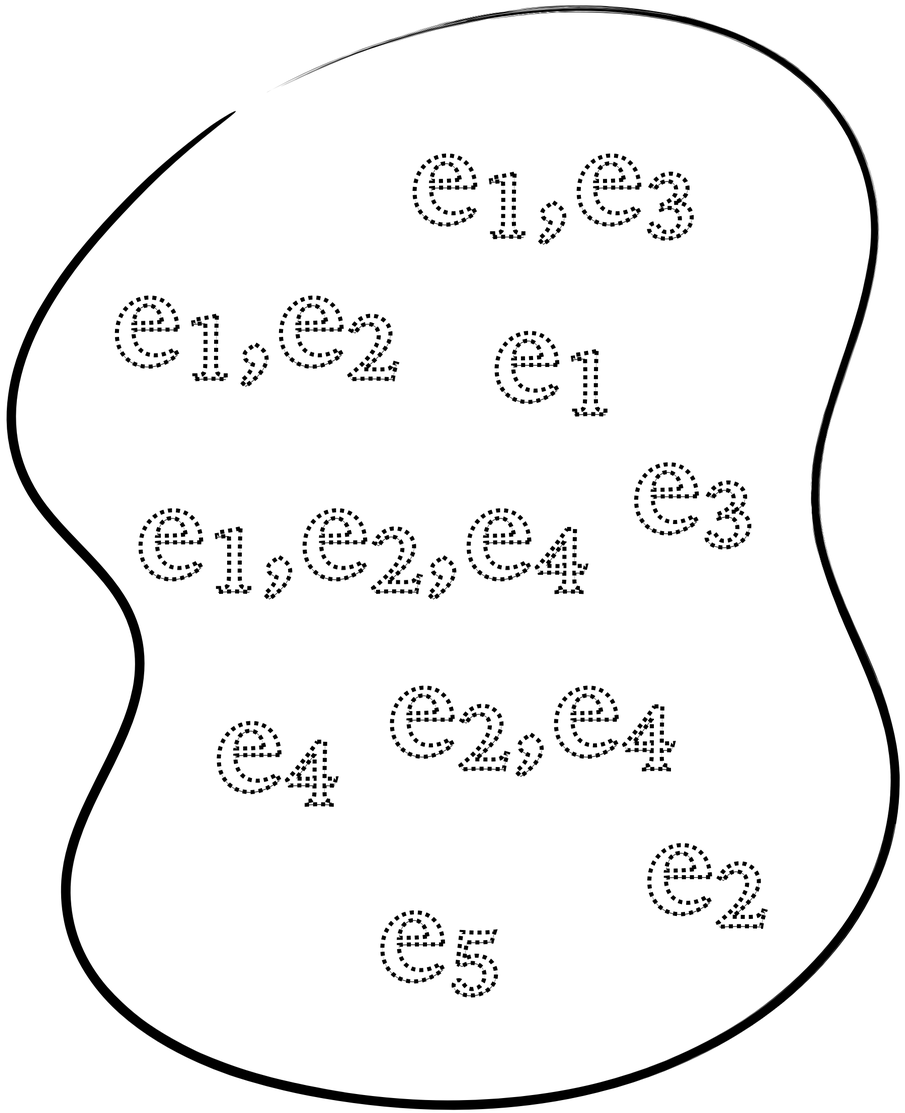


MOAD

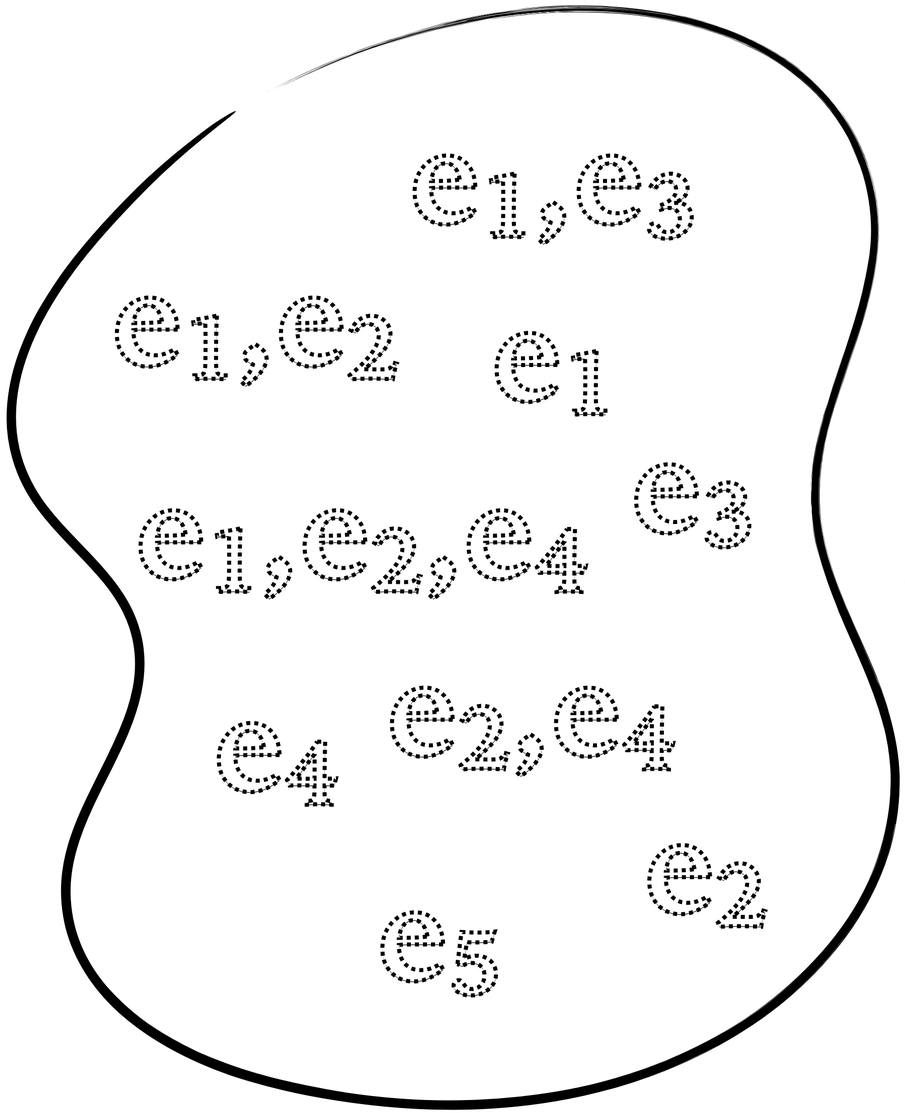


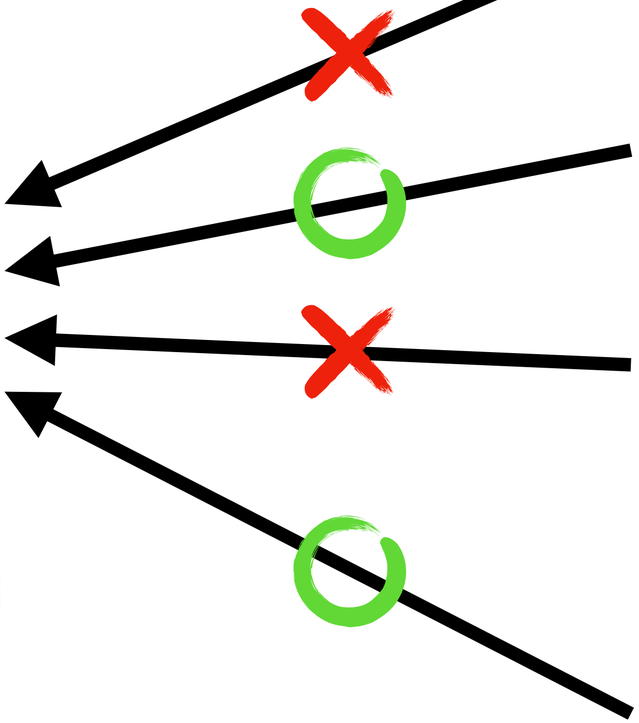
MOAD

MOAD



MOAD





Compile & Execute

$e_2 = 42$

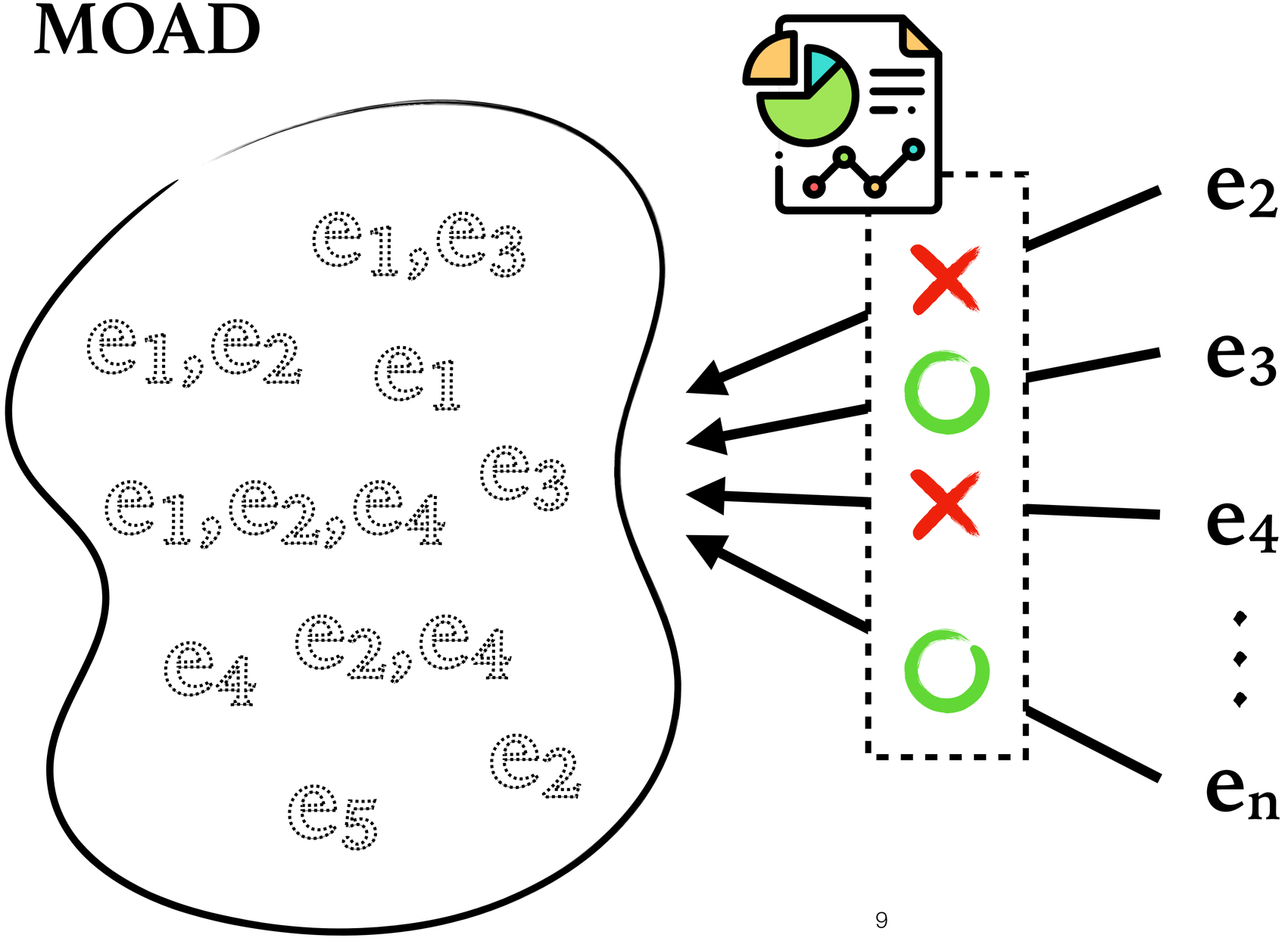
$e_3 = 3.141592$

$e_4 = \text{“foo”}$

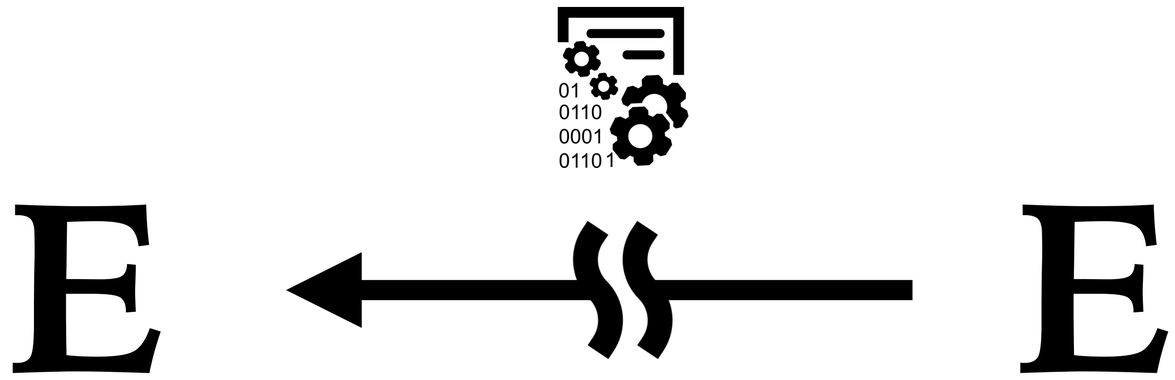
$\vdots$

$e_n = \text{bar}()$

MOAD



MOAD



ORBS

- Try iterative, sequential deletion attempts
- Relation with respect to a single criterion
- Exact (1-minimal), compilable, executable slice

MOAD

- Observe various independent partially deleted programs' behavior
- Program's overall dependency model
- An approximated dependency

ORBS

- Try iterative, sequential deletion attempts
- Relation with respect to a single criterion
- Exact (1-minimal), compilable, executable slice

```
int main() {
    int sum = 0;
    int i = 1;
    while (i < 11) {
        sum = sum + i;
        i = i + 1;
    }
    printf(“%d\n”, sum);
    printf(“%d\n”, i);
}
```

MOAD

- Observe various independent partially deleted programs’ behavior
- Program’s overall dependency model
- An approximated dependency

```
int main() {
    int sum = 0;
    int i = 1;
    while (i < 11) {
        sum = sum + i;
        i = i + 1;
    }
    printf(“%d\n”, sum);
    printf(“%d\n”, i);
}
```

ORBS

- Try iterative, sequential deletion attempts
- Relation with respect to a single criterion
- Exact (1-minimal), compilable, executable slice

```
int main() {

    int i = 1;
    while (i < 11) {

        i = i + 1;
    }

    printf("%d\n", i);
}
```

MOAD

- Observe various independent partially deleted programs' behavior
- Program's overall dependency model
- An approximated dependency

```
int main() {
    int sum = 0;
    int i = 1;
    while (i < 11) {
        sum = sum + i;
        i = i + 1;
    }
    printf("%d\n", sum);
    printf("%d\n", i);
}
```

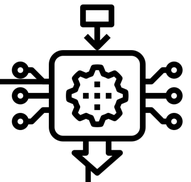
ORBS

- Try iterative, sequential deletion attempts
- Relation with respect to a single criterion
- Exact (1-minimal), compilable, executable slice

```
int main() {

    int i = 1;
    while (i < 11) {

        i = i + 1;
    }

    printf("%d\n", i);
}
```

MOAD

- Observe various independent partially deleted programs' behavior
- Program's overall dependency model
- An approximated dependency



```
int main() {
    int sum = 0;
    int i = 1;
    while (i < 11) {
        sum = sum + i;
        i = i + 1;
    }
    printf("%d\n", sum);
    printf("%d\n", i);
}
```

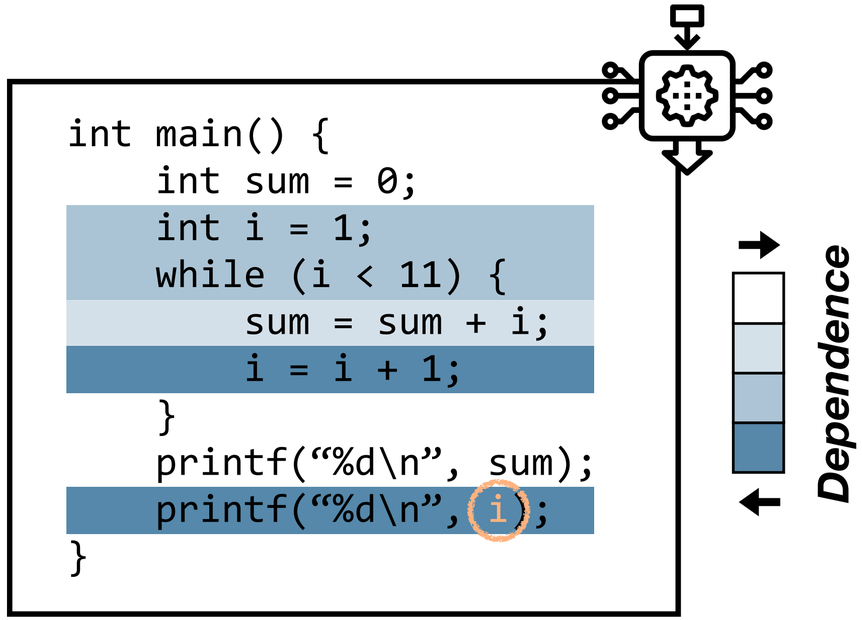
ORBS

- Try iterative, sequential deletion attempts
- Relation with respect to a single criterion
- Exact (1-minimal), compilable, executable slice

```
int main() {

    int i = 1;
    while (i < 11) {

        i = i + 1;
    }

    printf("%d\n", i);
}
```

MOAD

- Observe various independent partially deleted programs' behavior
- Program's overall dependency model
- An approximated dependency



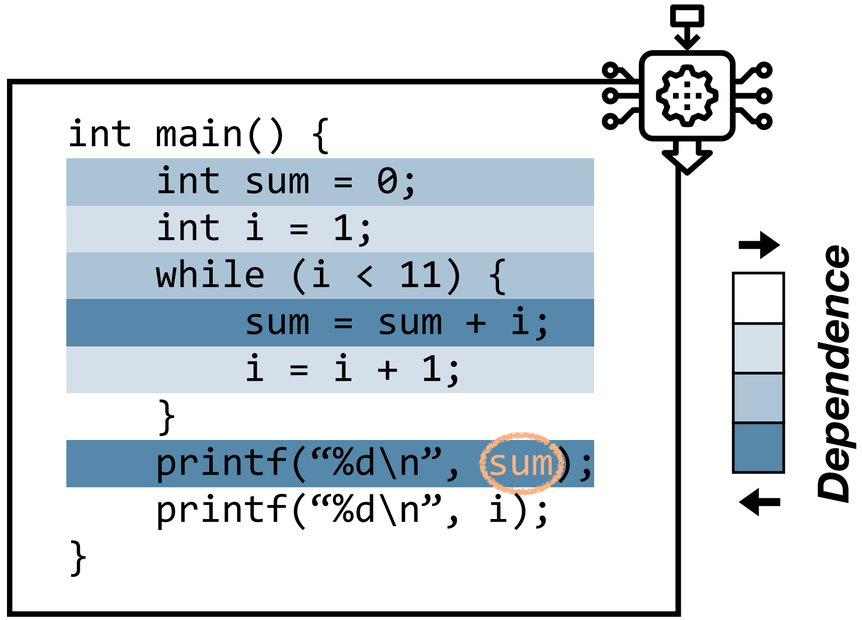
ORBS

- Try iterative, sequential deletion attempts
- Relation with respect to a single criterion
- Exact (1-minimal), compilable, executable slice

```
int main() {

    int i = 1;
    while (i < 11) {

        i = i + 1;
    }

    printf("%d\n", i);
}
```

MOAD

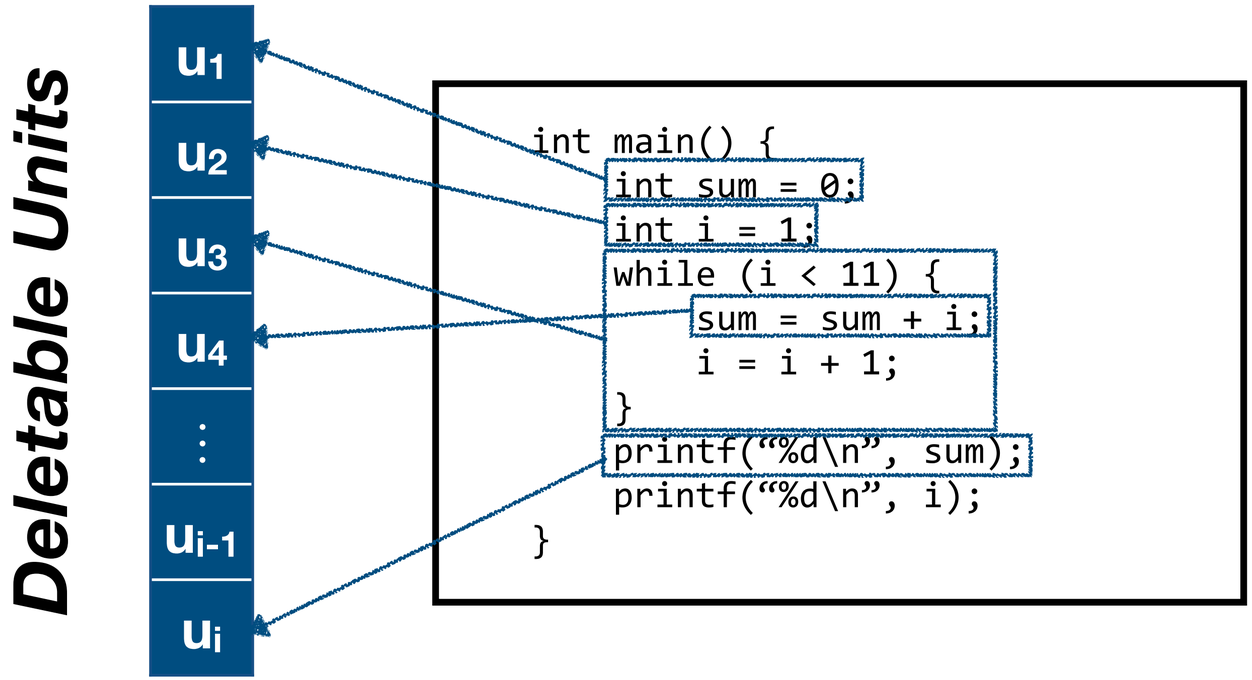
- Observe various independent partially deleted programs' behavior
- Program's overall dependency model
- An approximated dependency



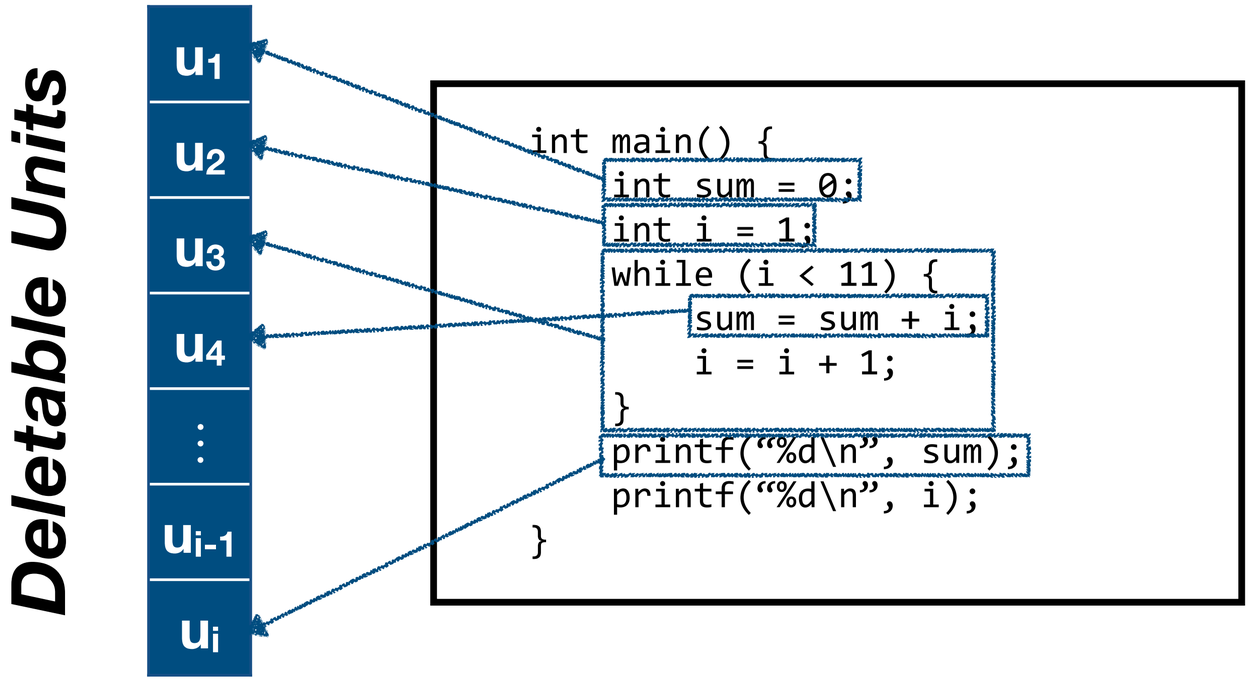
MOAD - Deletable Units

```
int main() {
    int sum = 0;
    int i = 1;
    while (i < 11) {
        sum = sum + i;
        i = i + 1;
    }
    printf(“%d\n”, sum);
    printf(“%d\n”, i);
}
```

MOAD - Deletable Units



MOAD - Deletable Units



Original program

u_1	u_2	u_3	u_4	…	u_{i-1}	u_i
0	0	0	0	…	0	0

- 1 : Unit deleted
- 0 : Unit remains

MOAD - Deletion Generation Scheme

MOAD - Deletion Generation Scheme

1) 1-hot

u_1	u_2	u_3	u_4	…	u_{i-1}	u_i
0	0	0	0	…	0	0
1	0	0	0	…	0	0
0	1	0	0	…	0	0
0	0	1	0	…	0	0
0	0	0	1	…	0	0
0	0	0	0	…	0	0
…	…	…	…	…	…	…
0	0	0	0	…	0	1

MOAD - Deletion Generation Scheme

1) 1-hot

u_1	u_2	u_3	u_4	…	u_{i-1}	u_i
0	0	0	0	…	0	0
1	0	0	0	…	0	0
0	1	0	0	…	0	0
0	0	1	0	…	0	0
0	0	0	1	…	0	0
0	0	0	0	…	0	0
…	…	…	…	…	…	…
0	0	0	0	…	0	1

+

2) 2-hot

u_1	u_2	u_3	u_4	…	u_{i-1}	u_i
…	…	…	…	…	…	…
1	1	0	0	…	0	0
1	0	1	0	…	0	0
1	0	0	1	…	0	0
…	…	…	…	…	…	…
0	1	0	1	…	0	0
…	…	…	…	…	…	…
0	0	0	0	…	1	1

Partially deleted prog.

u_1	u_2	u_3	u_4	…	u_{i-1}	u_i
0	0	0	0	…	0	0
1	0	0	0	…	0	0
0	1	0	0	…	0	0
0	0	1	0	…	0	0
0	0	0	1	…	0	0
			⋮			
0	0	1	0	…	0	1
0	1	1	1	…	0	0
1	0	0	0	…	1	0

Partially deleted prog.

u_1	u_2	u_3	u_4	…	u_{i-1}	u_i
0	0	0	0	…	0	0
1	0	0	0	…	0	0
0	1	0	0	…	0	0
0	0	1	0	…	0	0
0	0	0	1	…	0	0
			⋮			
0	0	1	0	…	0	1
0	1	1	1	…	0	0
1	0	0	0	…	1	0

Response

v_1	v_2	v_3	…	v_j
1	1	1	…	1

- **1 : Same behavior**
- **0 : Compile error or different behavior**

Response

u_1	u_2	u_3	u_4	…	u_{i-1}	u_i	v_1	v_2	v_3	…	v_j
0	0	0	0	…	0	0	1	1	1	…	1
1	0	0	0	…	0	0	0	0	0	…	0
0	1	0	0	…	0	0	1	1	0	…	1
0	0	1	0	…	0	0	1	1	1	…	0
0	0	0	1	…	0	0	1	1	0	…	1
			⋮						⋮		
0	0	1	0	…	0	1	1	1	1	…	0
0	1	1	1	…	0	0	0	0	0	…	0
1	0	0	0	…	1	0	0	0	0	…	0

Partially deleted prog.

Observed behavior

Training data

u_1	u_2	u_3	u_4	…	u_{i-1}	u_i	v_k
0	0	0	0	…	0	0	1
1	0	0	0	…	0	0	0
0	1	0	0	…	0	0	0
0	0	1	0	…	0	0	1
0	0	0	1	…	0	0	0
0	0	1	0	…	0	1	1
…	…	…	…	…	…	…	…
1	0	0	0	…	1	0	0

Training data

u_1	u_2	u_3	u_4	…	u_{i-1}	u_i	v_k
0	0	0	0	…	0	0	1
1	0	0	0	…	0	0	0
0	1	0	0	…	0	0	0
0	0	1	0	…	0	0	1
0	0	0	1	…	0	0	0
0	0	1	0	…	0	1	1
…	…	…	…	…	…	…	…
1	0	0	0	…	1	0	0

M :

u_1	u_2	u_3	u_4	…	u_{i-1}	u_i
0	*0*	*1*	*0*	*…*	*0*	*1*

v_k
1

Statistical model

M :

u_1	u_2	u_3	u_4	…	u_{i-1}	u_i
0	*0*	*1*	*0*	*…*	*0*	*1*

→

v_k
1

Statistical model

$$Dep(M, v_k) = \{u_i\}$$

$\mathbf{u_i}$ ← $\mathbf{v_k}$

Infer dependency

MOAD - Inference Algorithm

- **Once-success ($\mathbb{O}$)**

u_i	v_k
1	1

If the behavior of v_k is preserved at least once when u_i is deleted, then v_k Is independent from u_i.

MOAD - Inference Algorithm

- Logistic regression ($\mathbb{L}$)

If β_i, the coefficient for u_i of the logistic regression for v_k, is larger than 0, then v_k Is independent from u_i.

u_1	u_2	u_3	u_4	…	u_{i-1}	u_i	v_k
0	0	0	0	…	0	0	1
1	0	0	0	…	0	0	0
0	1	0	0	…	0	0	1
…	…	…	…	…	…	…	…
1	0	0	0	…	1	0	0

Observed data

Coefficients represent the relative impact on dependence

$$\log \frac{v_k}{1 - v_k} = \beta_0 + \beta_1 u_1 + \beta_2 u_2 + \cdots + \beta_i u_i$$

$$u_i \xleftarrow[\text{if } \beta_i > 0]{\text{if } \beta_i \leq 0} v_k$$

MOAD - Inference Model

- **Bayesian inference ($\mathbb{B}$)**

If the $P(v_k$ behaves the same $|\ u_i$ has been deleted$)$ is larger than the mean, then v_k Is independent from u_i.

$$
\begin{aligned}
P\left(v_k \mid u_i\right) &= P\left(v_k \text{ behaves the same } \mid \mathrm{u_i} \text{ has been deleted } \right) \\
&= P(v_k = 1 \mid u_i = 1) \\
&= \frac{P(v_k = 1,\ u_i = 1)}{P(u_i = 1)}
\end{aligned}
$$

$$
\begin{aligned}
\hat{P}\left(v_k \mid u_i\right) &= \frac{\#(v_k = 1 \text{ and } \mathrm{u_i} = 1)/|O|}{\#(u_i = 1)/|O|} \\
&= \frac{\#(v_k = 1 \text{ and } \mathrm{u_i} = 1)}{\#(u_i = 1)}
\end{aligned}
$$

Estimate with the frequency of behavior preservation

$$
\mathbf{u_i} \xleftarrow[\text{if } \hat{P}\left(v_k \mid u_i\right) > \mu]{\text{if } \hat{P}\left(v_k \mid u_i\right) \le \mu} \mathbf{v_k}
$$

μ **: Average of the probability over units**

MOAD

2 deletion generation schema X 3 inference algorithms

MOAD

2 deletion generation schema X 3 inference algorithms

V.S.

Program Slicing

- ***For all numeric variables***

- ***Number of observations needed***
- ***Size of the slices***
- ***Difference of the slices***

ORBS

Statement level

MOAD

2 deletion generation schema X 3 inference algorithms

V.S.

Program Slicing

- For all numeric variables

- ***Number of observations needed***
- ***Size of the slices***
- ***Difference of the slices***

ORBS

Line of text level

Statement level

MOAD

---

2 deletion generation schema X 3 inference algorithms

V.S.

Program Slicing

- *For all numeric variables*

- *Number of observations needed*
- *Size of the slices*
- *Difference of the slices*

ORBS

Line of text level

T-ORBS

Statement level

Gold et al., "Generalized Observational Slicing for Tree-Represented Modelling Languages", *FSE'17*

Statement level

MOAD

2 deletion generation schema X 3 inference algorithms

V.S.

Program Slicing

- For all numeric variables

- ***Number of observations needed***
- ***Size of the slices***
- ***Difference of the slices***

W-ORBS

Line of text level

T-ORBS

Statement level

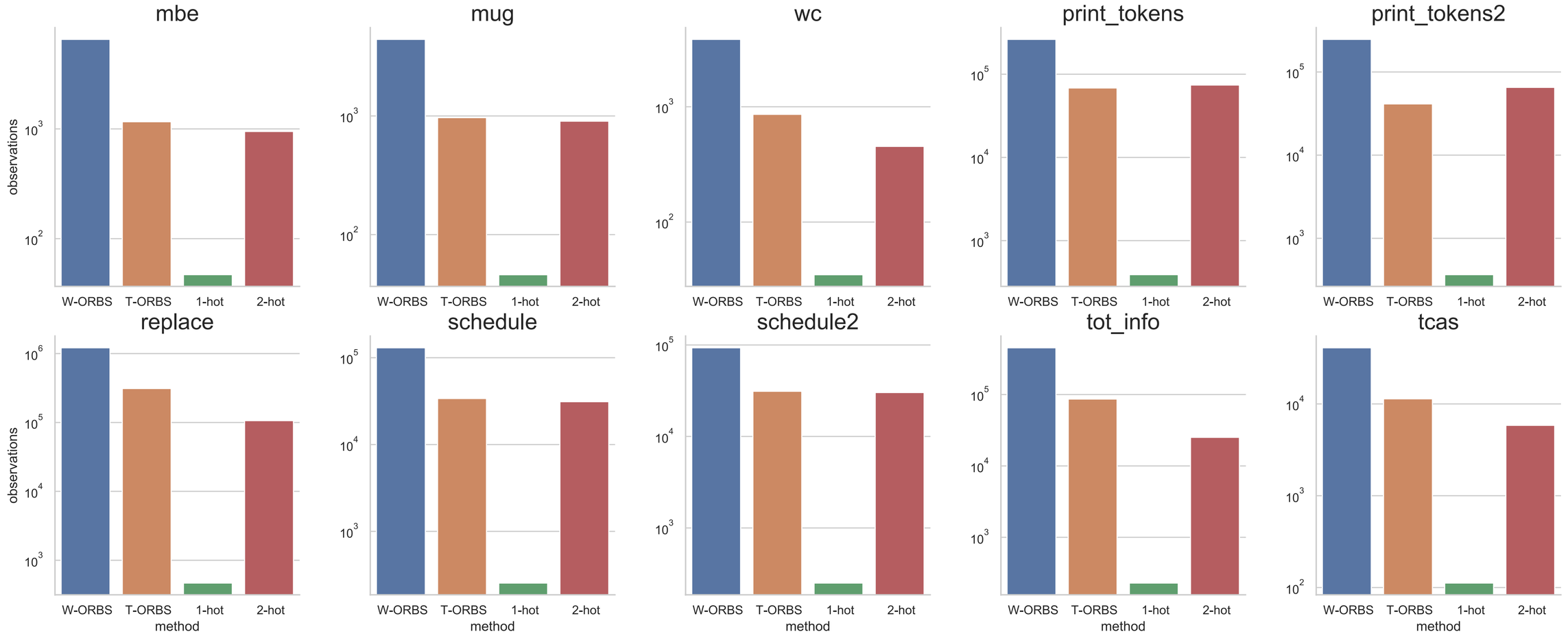
Gold et al., "Generalized Observational Slicing for Tree-Represented Modelling Languages", *FSE'17*

* Binkley et al., "ORBS and the limits of static slicing”, *SCAM'15*

MOAD - Subjects

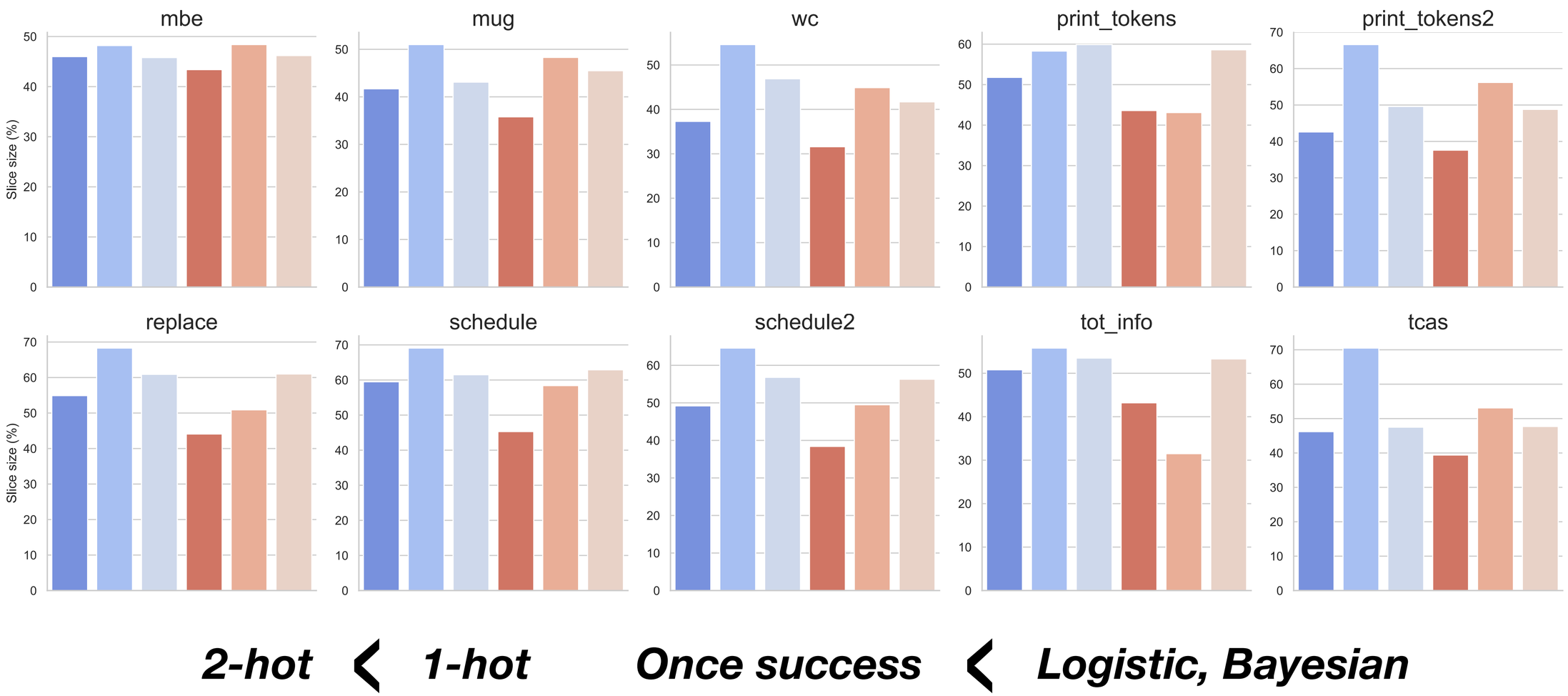
Subject	SLoC	# of statements	# of numeric variables
mbe *	64	45	16
mug *	61	44	13
wc *	46	33	17
print_tokens	410	388	98
print_tokens2	387	364	75
replace	508	465	253
schedule	283	252	75
schedule2	276	248	81
tot_info	314	227	210
tcas	152	110	62

of observations (log scale)



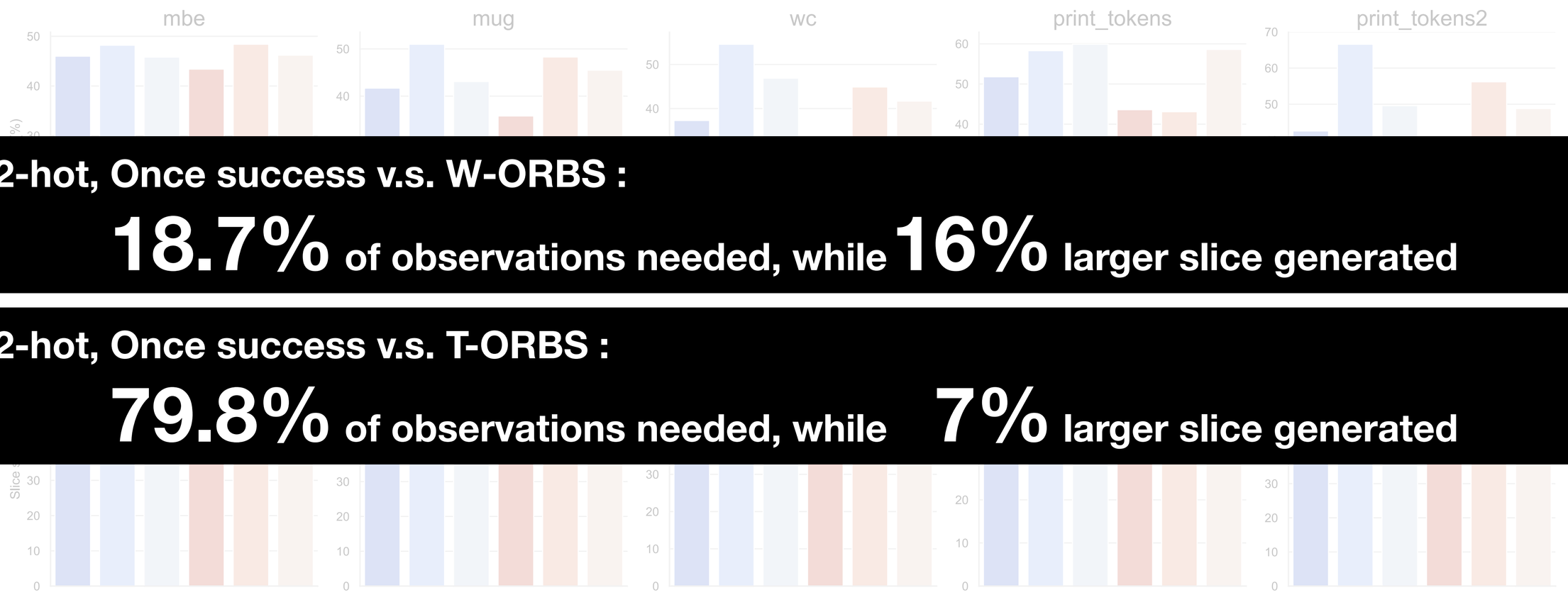
$$\frac{\text{MOAD, 1-hot}}{\text{W-ORBS}} = 0.37\% \qquad \frac{\text{MOAD, 2-hot}}{\text{W-ORBS}} = 18.7\% \qquad \frac{\text{MOAD, 2-hot}}{\text{T-ORBS}} = 79.8\%$$

Mean slice size / Original program size (%)



2-hot < 1-hot Once success < Logistic, Bayesian

Mean slice size / Original program size (%)



2-hot, Once success v.s. W-ORBS :

18.7% of observations needed, while 16% larger slice generated

2-hot, Once success v.s. T-ORBS :

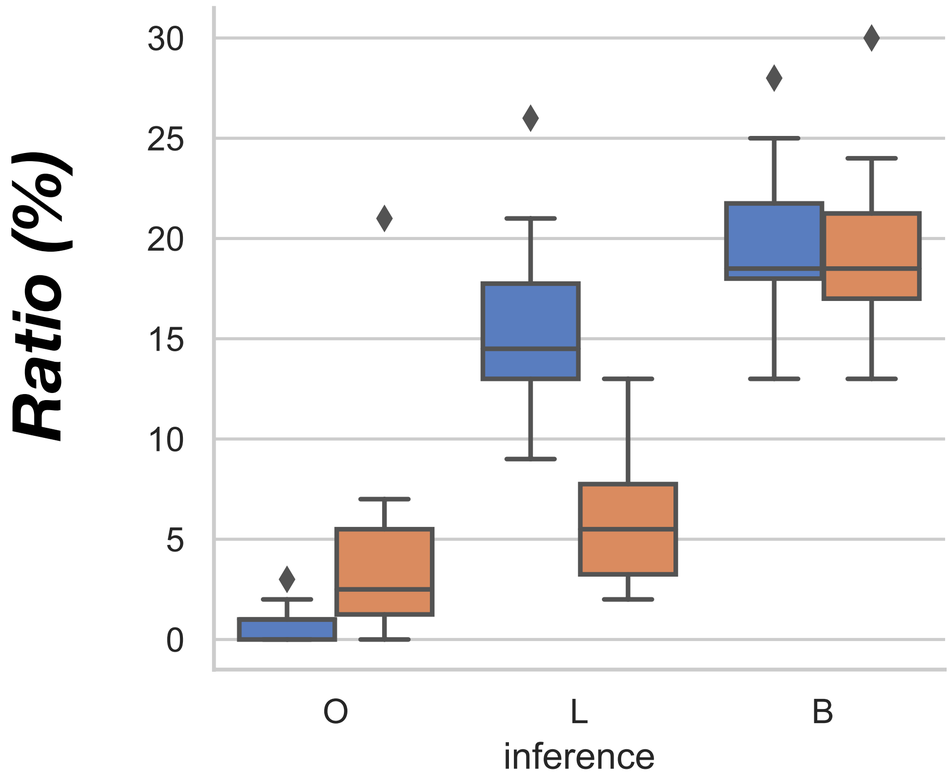
79.8% of observations needed, while 7% larger slice generated

2-hot < 1-hot Once success < Logistic, Bayesian

O: Once-success
L: Logistic
B: bayesian

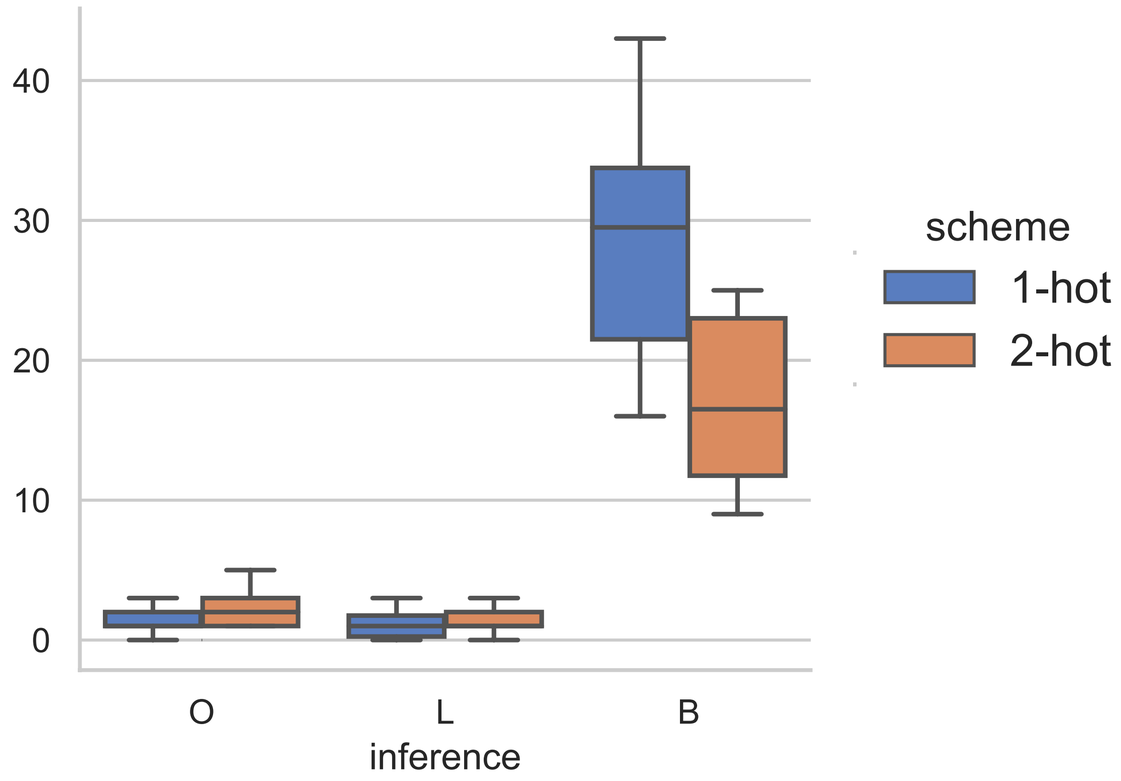
Miss

The number of statements MOAD fails to delete



Excess

The number of statements MOAD excessively deletes



Future work

Enhance MOAD

- Advanced, adaptive deletion generation scheme
- Alternative inference algorithm
  - Bayesian Networks, Markov Random Fields, Gaussian processes
- Parallelization

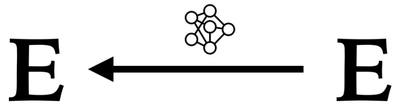
Future work

Enhance MOAD

- Advanced, adaptive deletion generation scheme
- Alternative inference algorithm
  - Bayesian Networks, Markov Random Fields, Gaussian processes
- Parallelization

Apply MOAD to various other SE tasks

Static Analysis



ORBS

Lightweight dynamic analysis

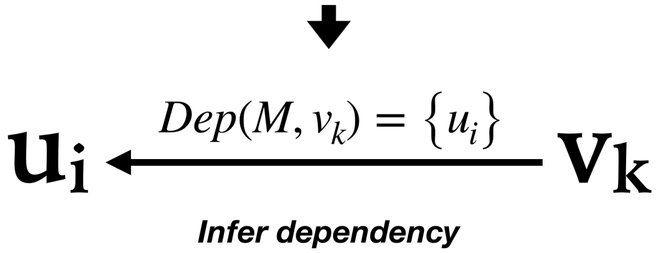
\+

Modeling dependency

M :

u_1	u_2	u_3	u_4	...	u_{i-1}	u_i	→	v_k
0	0	1	0	...	0	1		1

Statistical model



Infer dependency

MOAD — Statement level

2 deletion generation schema X **3** inference algorithms

Program Slicing
- *For all numeric variables*

V.S.

- *Number of observations needed*
- *Size of the slices*
- *Difference of the slices*

W-ORBS — Line of text level

T-ORBS — Statement level

Mean slice size / Original program size (%)

2-hot, Once success v.s. W-ORBS :

18.7% less observations needed, while **16%** larger slice generated

2-hot, Once success v.s. T-ORBS :

79.8% less observations needed, while **7%** larger slice generated

2-hot << 1-hot *Once success << Logistic, Bayesian*

Appendix A. Success rate

Subject	Deletion Gen. Scheme	Success Rate $\mathbb{O}$	$\mathbb{L}$	$\mathbb{B}$
mbe	1-hot	100%	100%	100%
	2-hot	100%	100%	100%
mug	1-hot	100%	100%	100%
	2-hot	100%	100%	100%
wc	1-hot	100%	100%	100%
	2-hot	88%	76%	100%
prttok	1-hot	03%	04%	11%
	2-hot	03%	03%	11%
prttok2	1-hot	72%	19%	77%
	2-hot	63%	13%	67%
replace	1-hot	7%	31%	28%
	2-hot	3%	13%	31%
sched	1-hot	48%	47%	41%
	2-hot	39%	35%	43%
sched2	1-hot	30%	26%	28%
	2-hot	17%	26%	28%
totinfo	1-hot	52%	50%	62%
	2-hot	32%	10%	65%
tcas	1-hot	48%	90%	48%
	2-hot	26%	68%	48%

TABLE II: MOAD's success rate on the ten test subjects

Appendix B. Sampling Effect

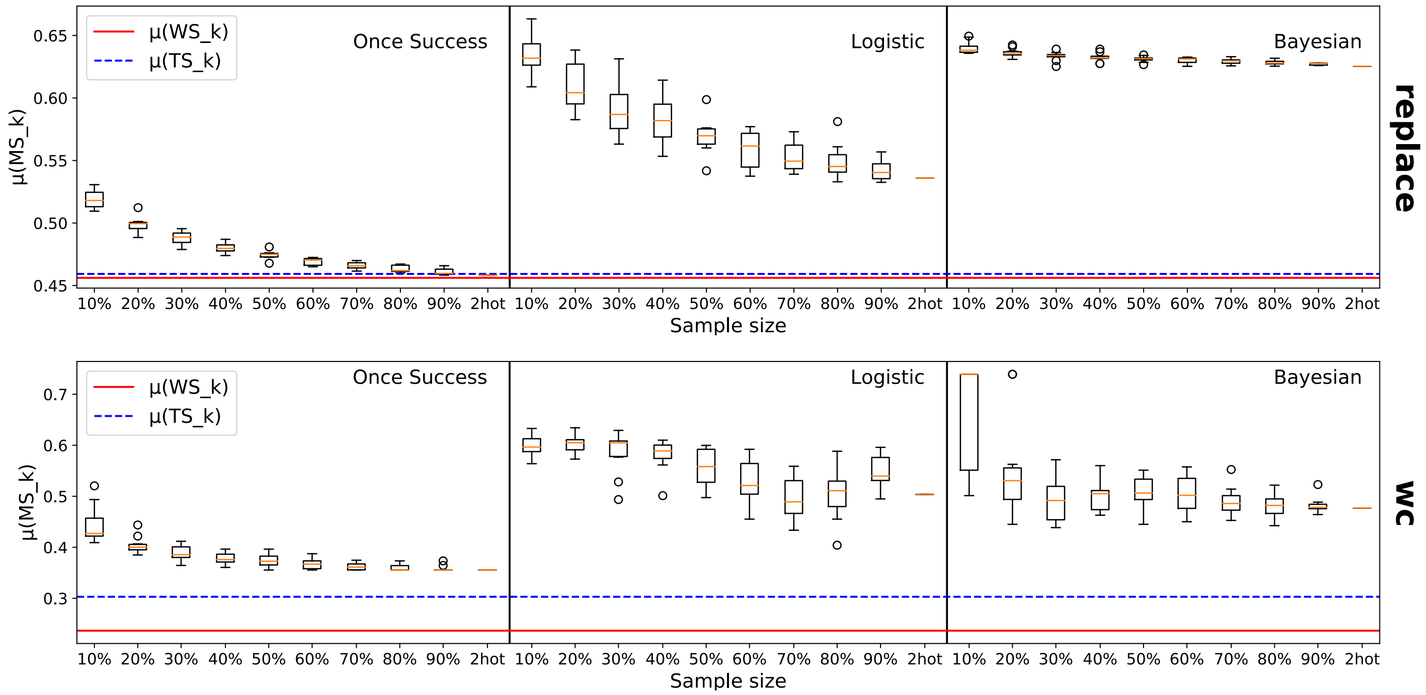


Fig. 1: The figure presents $\mu\,(MS_k)$ which represents the mean slice size given as a percentage of the original program's size, generated by MOAD using each size of the sample from 2-hot data. The boxplot shows the results of a trained model from 10 different random samplings. The red and blue line represents the ratio of the W-ORBS and T-ORBS slice size to the original program size, averaging by all slicing criteria of all subjects.