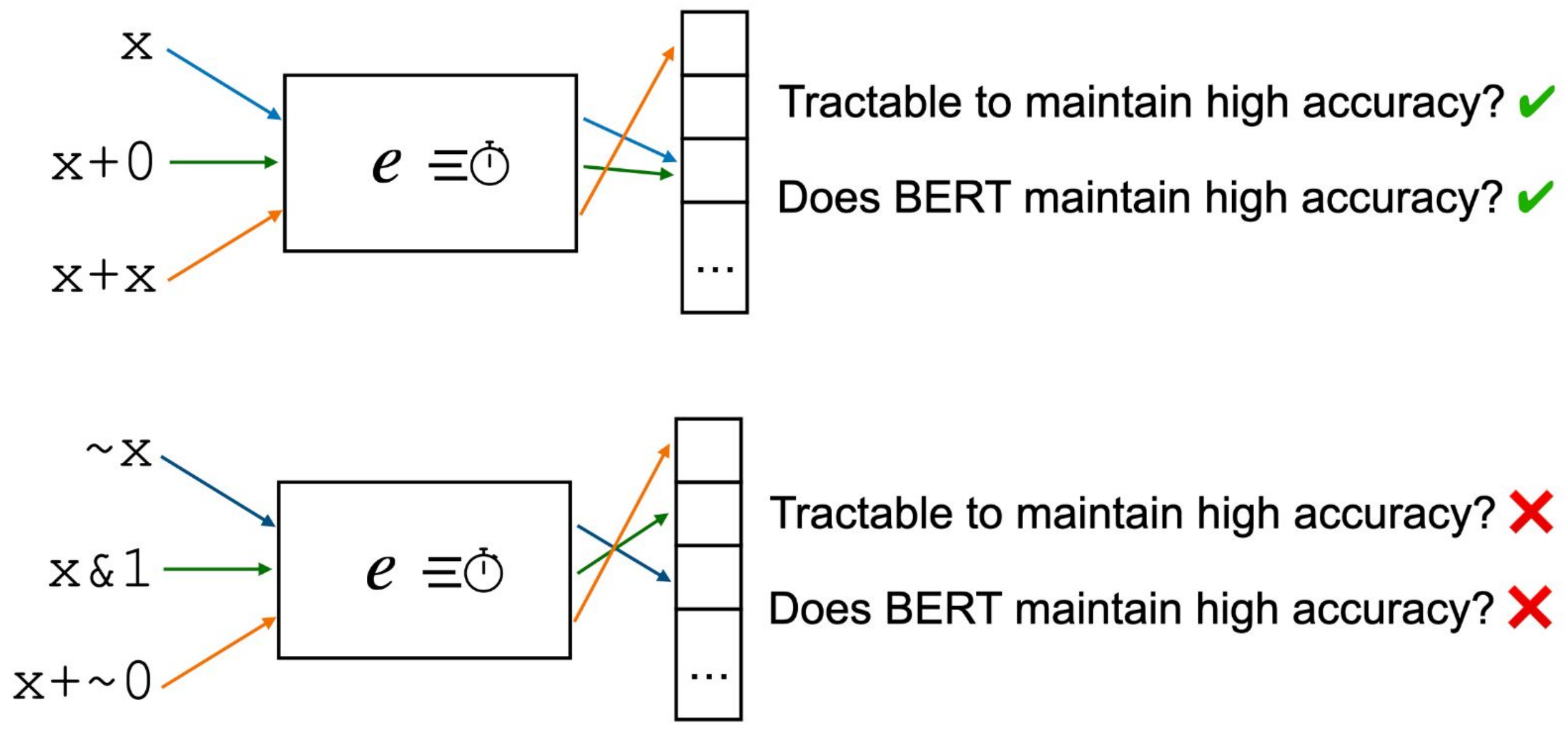
# A Theory of Equivalence-Preserving Program Embeddings

Logan Weber*, Jesse Michel*, Alex Renda, Saman Amarasinghe, Michael Carbin

We characterize when it is tractable to embed a programming language such that semantic equivalence is preserved.

## Abstract

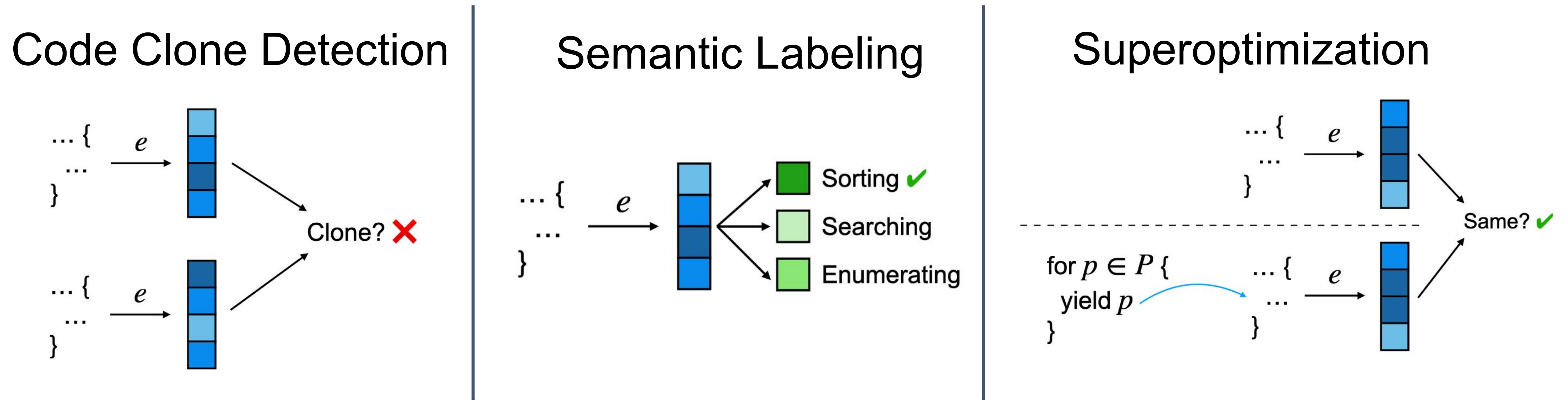
Program embeddings are increasingly used to solve program reasoning tasks. We develop a theory of program embeddings for solving tasks that require reasoning about program semantics.


x
x+0
x+x
$e\ \equiv$⏱
...
Tractable to maintain high accuracy? ✓
Does BERT maintain high accuracy? ✓
~x
x&1
x+~0
$e\ \equiv$⏱
...
Tractable to maintain high accuracy? ✗
Does BERT maintain high accuracy? ✗

We compare two languages theoretically, then show BERT-Tiny's performance on each language accords with our theory.

## Semantic Tasks

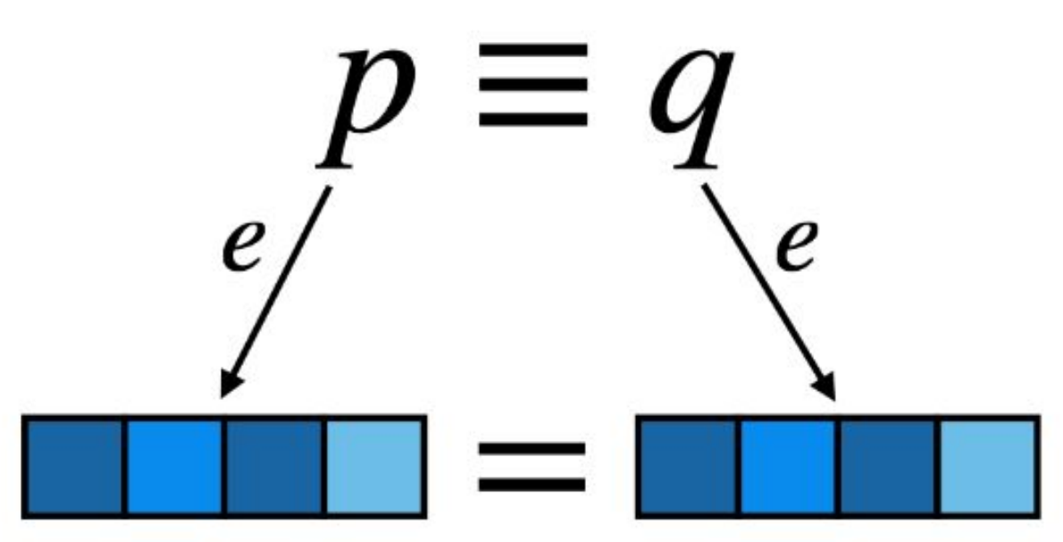
*Semantic tasks* are tasks where only the input-output behavior of a program is relevant.



In code clone detection, the goal is to find duplicates of a given program in a codebase. In semantic labeling, the goal is to identify the semantic behavior a program exhibits from a fixed collection of possible behaviors. In superoptimization, the goal is to produce a semantically equivalent program that is optimal with respect to some metric.

## Equivalence-Preserving Program Embeddings

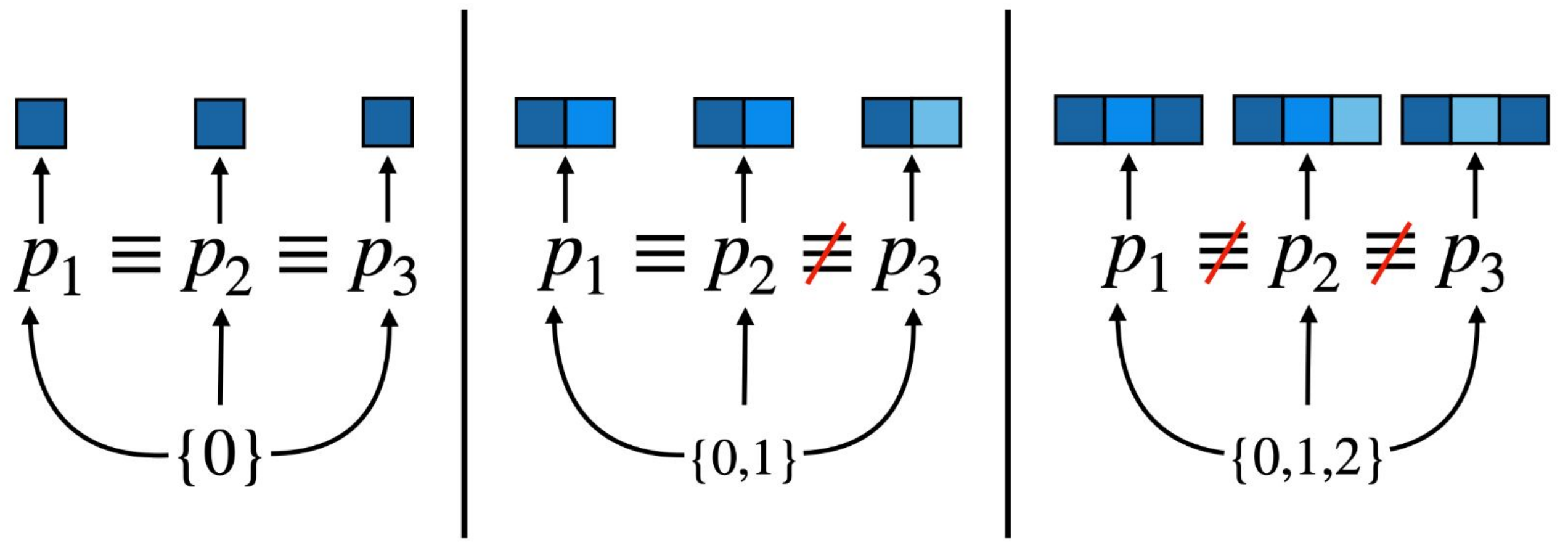
Embeddings that are identical exactly when programs are semantically equivalent perfectly solve the above tasks. We call such embeddings *equivalence-preserving embeddings*.



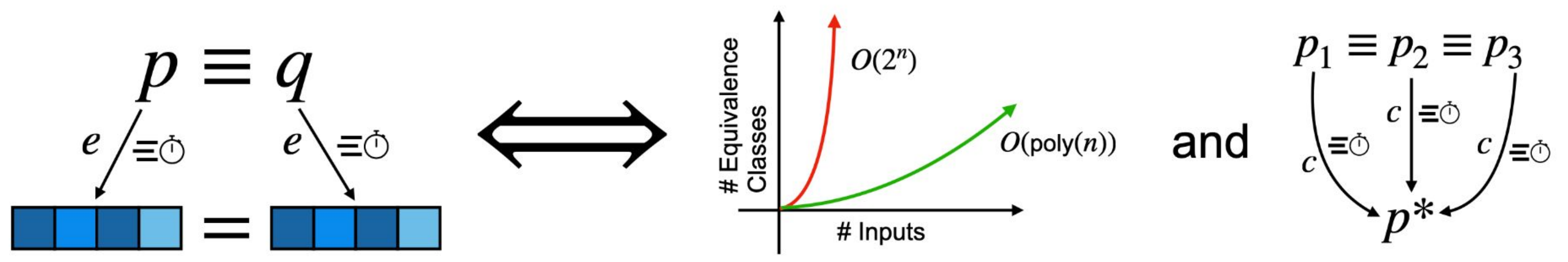
*Denotes equal contribution.

## Theoretical Results

By probing programs with an increasing number of inputs, one can determine the complexity of a language's semantics.


$p_1 \equiv p_2 \equiv p_3$
{0}
$p_1 \equiv p_2 \not\equiv p_3$
{0,1}
$p_1 \not\equiv p_2 \not\equiv p_3$
{0,1,2}

We show that constructing equivalence-preserving embeddings for a programming language is tractable exactly when the number of semantically distinct programs is polynomial in the number of probing inputs and the language can be efficiently canonicalized.

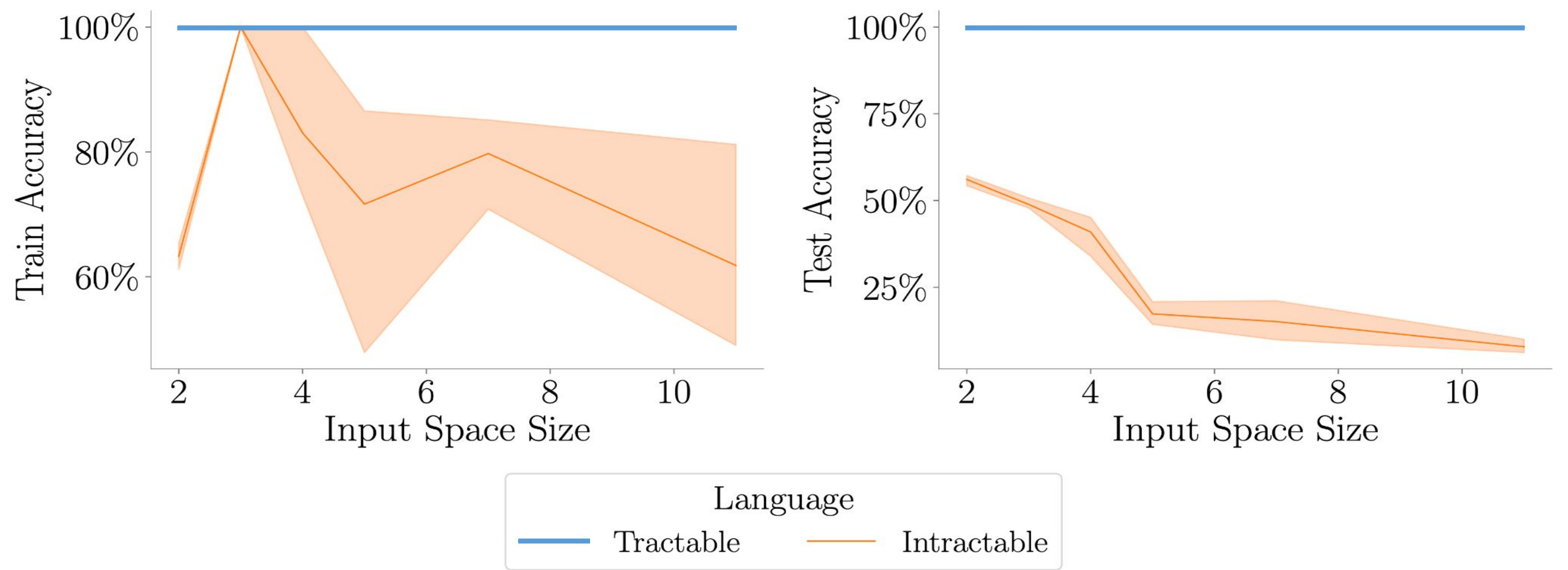


## Empirical Study

We consider a modular addition language and a larger language with bitwise operators. We prove the former can be tractably embedded while the latter cannot be, then we show BERT-Tiny's ability to learn equivalence-preserving embeddings for the intractable language degrades significantly faster than for the tractable language.

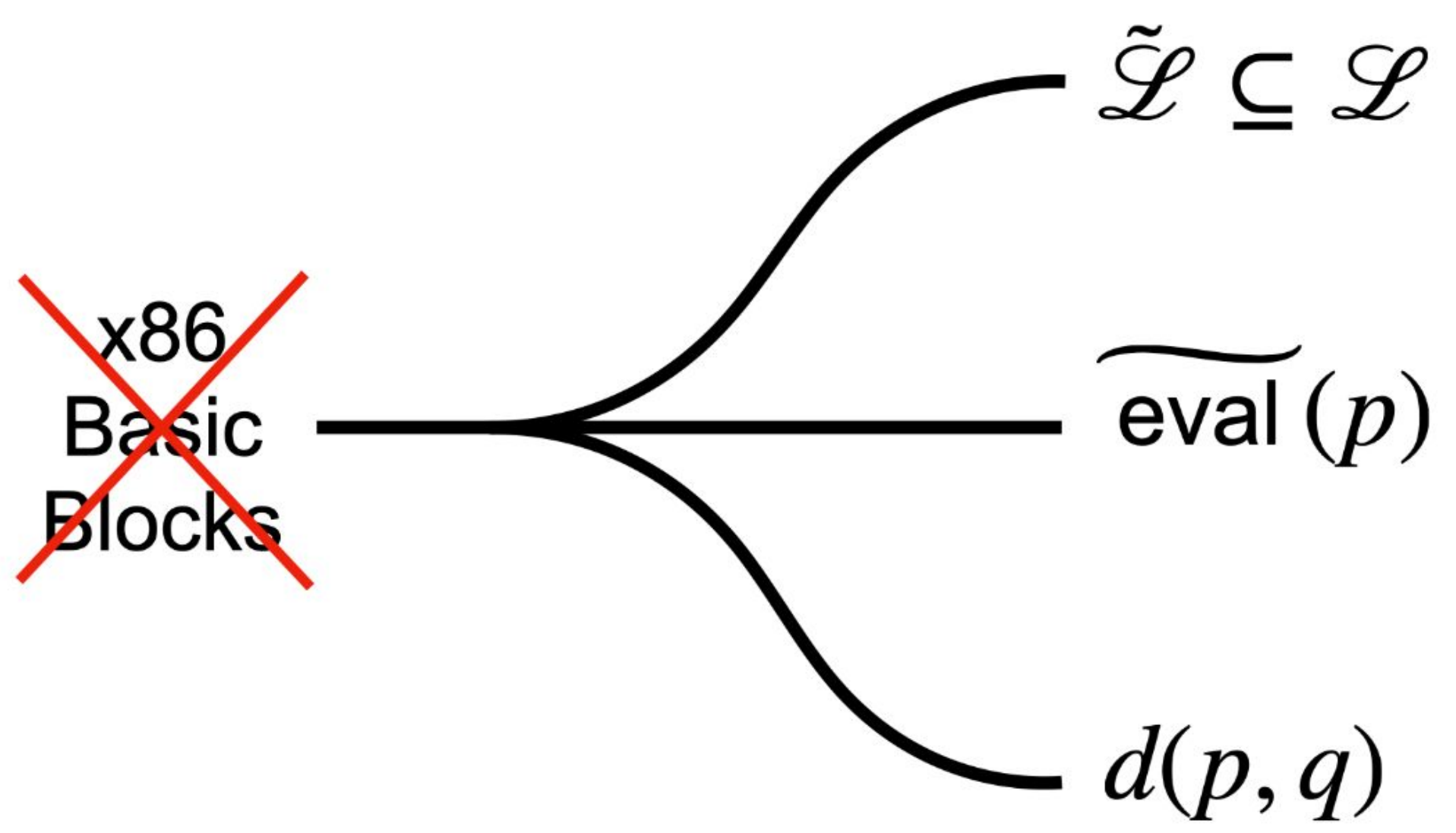
$$e \coloneqq 0 \mid 1 \mid \mathtt{x} \mid e + e$$

$$e \coloneqq 0 \mid 1 \mid \mathtt{x} \mid e + e \mid e\ \&\ e \mid e \mid e \mid \mathtt{\sim} e$$



## Discussion

Our results show subsets of basic-block assembly cannot be tractably embedded, suggesting approximation is necessary in practice. For future work, we consider identifying tractable subsets of languages, approximating the semantics of languages, and relaxations using semantic similarity.


x86 Basic Blocks
$\tilde{\mathscr{L}} \subseteq \mathscr{L}$
$\widetilde{\mathrm{eval}}\,(p)$
$d(p, q)$