Analyzing and Securing Software via Robust and Generalizable Learning

Kexin Pei
Columbia University

- Program analysis is a crucial technique to build trustworthy software, but traditional program analysis incurs significant **manual effort** to tune for (1) **heterogeneous software components**, and (2) **various security applications**.
- ML4Code is promising, e.g., automated bug finding, program optimization, but shown **not robust** and **not generalizable** due to lack of understanding of **program semantics**.

```
...
    for (int j=0;j<array.length-1-i;j++) {
      if (array[j]>array[j+1]) {
        int temp = array[j];
        array[j] = array[j+1];
        array[j+1] = temp;
...
```

Prediction: `sort` (98.54%) ✓

```
...
    for (int j=0;j<ttypes.length-1-i;j++) {
      if (ttypes[j]>ttypes[j+1]) {
        int temp = ttypes[j];
        ttypes[j] = ttypes[j+1];
        ttypes[j+1] = temp;
...
```

Prediction: `contains` (99.97%) ✗

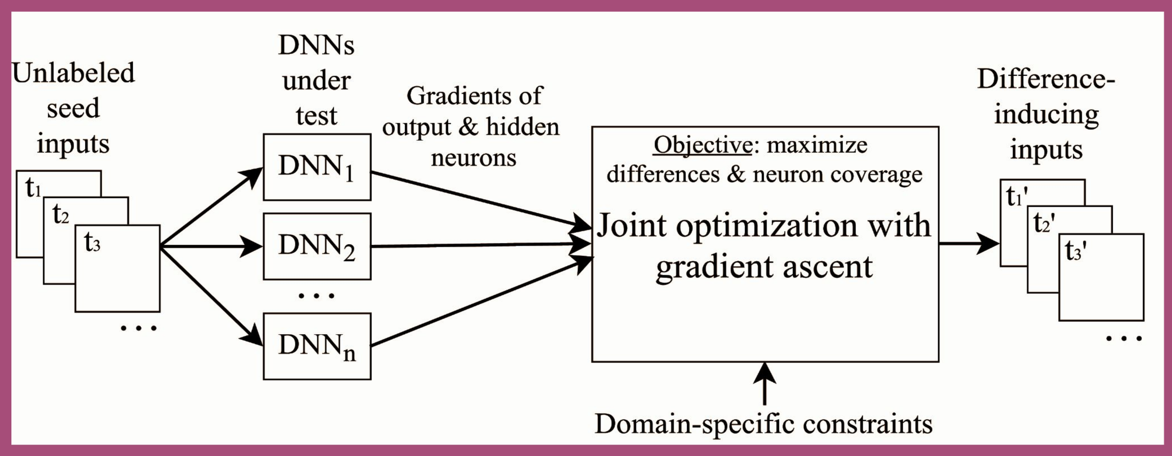
Trustworthy AI

Execution-aware program representation

[FSE'21,22,TSE'22,CCS'22,ICSE'23,ICML'23]

Testing and Formal Verification

[SOSP'17,ICSE'18,Usenix'18,Neurips'19]



for Trustworthy Software

Security-critical program analysis tasks:

Specification Inference [ICML'23]

Semantic Similarity [TSE'22]

SSL/TLS Hostname Verification [Oakland S&P'17]

Type Inference [FSE'16]

Memory Dependence [FSE'22]

Debug Symbol Recovery [FSE'21, CCS'22]

Attack Forensics [ACSAC'16]

Malware Analysis [DSN'15]

Disassembly [NDSS'21]

Fuzzing [Oakland S&P'19]

Efficient 98.1✕

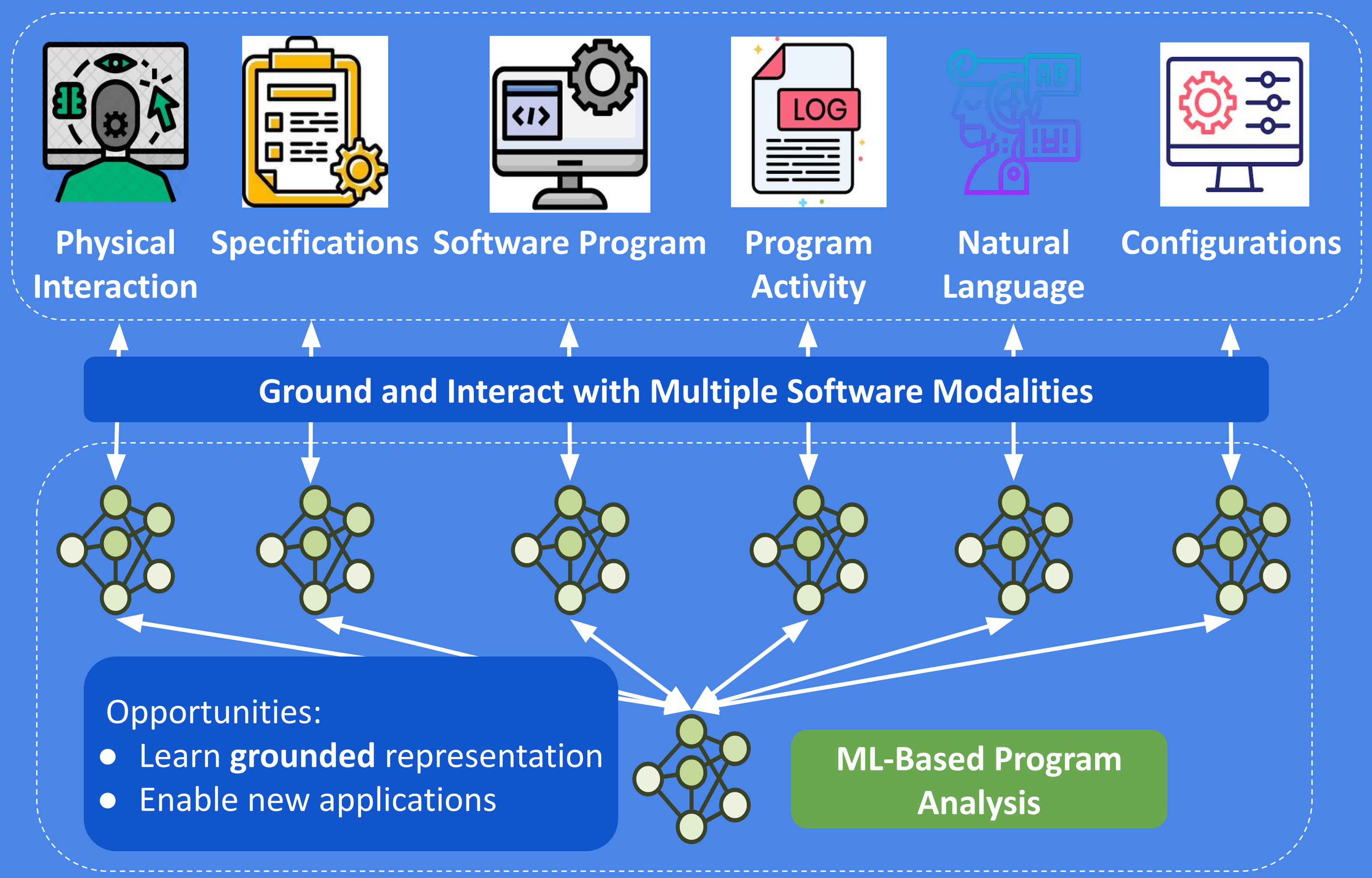
Precise 118%+

Generalizable and robust across:

- Compilers
- Architectures (x64, x86)
- Optimizations (-O0 -O1 -O2 -O3 -O Ox)
- Obfuscations

Future Work

Semantics-Grounded LLMs



Robustness by Construction