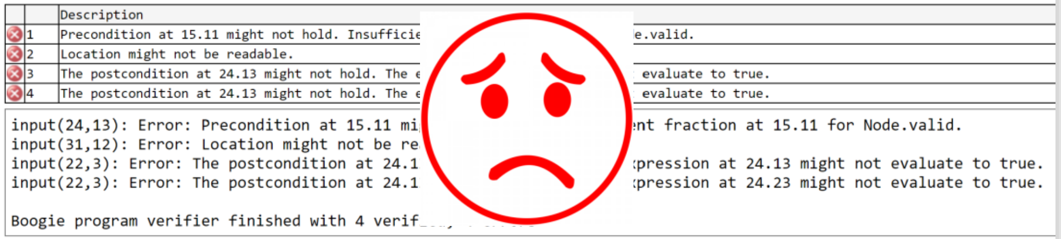
Empirically Evaluating Gradual Verification

Jenna Wise
(Carnegie Mellon University)

Mona Zhang (Columbia University), Jacob Gorenburg (Haverford College),
Jonathan Aldrich (Carnegie Mellon University),
Éric Tanter (University of Chile), Joshua Sunshine (Carnegie Mellon University)

Naïve Verification Attempt

```
int findMax(Node l)
  ensures max(result,l) && contains(result,l)
{
  int m := l.val;
  Node curr := l.next;
  while(curr != null) {
    if(curr.val > m) {
      m := curr.val;
    }
    curr := curr.next;
  }
  return m;
}
```



		Description
✖	1	Precondition at 15.11 might not hold. Insufficie e.valid.
✖	2	Location might not be readable.
✖	3	The postcondition at 24.13 might not hold. The e evaluate to true.
✖	4	The postcondition at 24.13 might not hold. The e evaluate to true.

```
input(24,13): Error: Precondition at 15.11 mi nt fraction at 15.11 for Node.valid.
input(31,12): Error: Location might not be re
input(22,3): Error: The postcondition at 24.1 xpression at 24.13 might not evaluate to true.
input(22,3): Error: The postcondition at 24.1 xpression at 24.23 might not evaluate to true.

Boogie program verifier finished with 4 verif
```

Naïve Verification Attempt: Additional Specifications

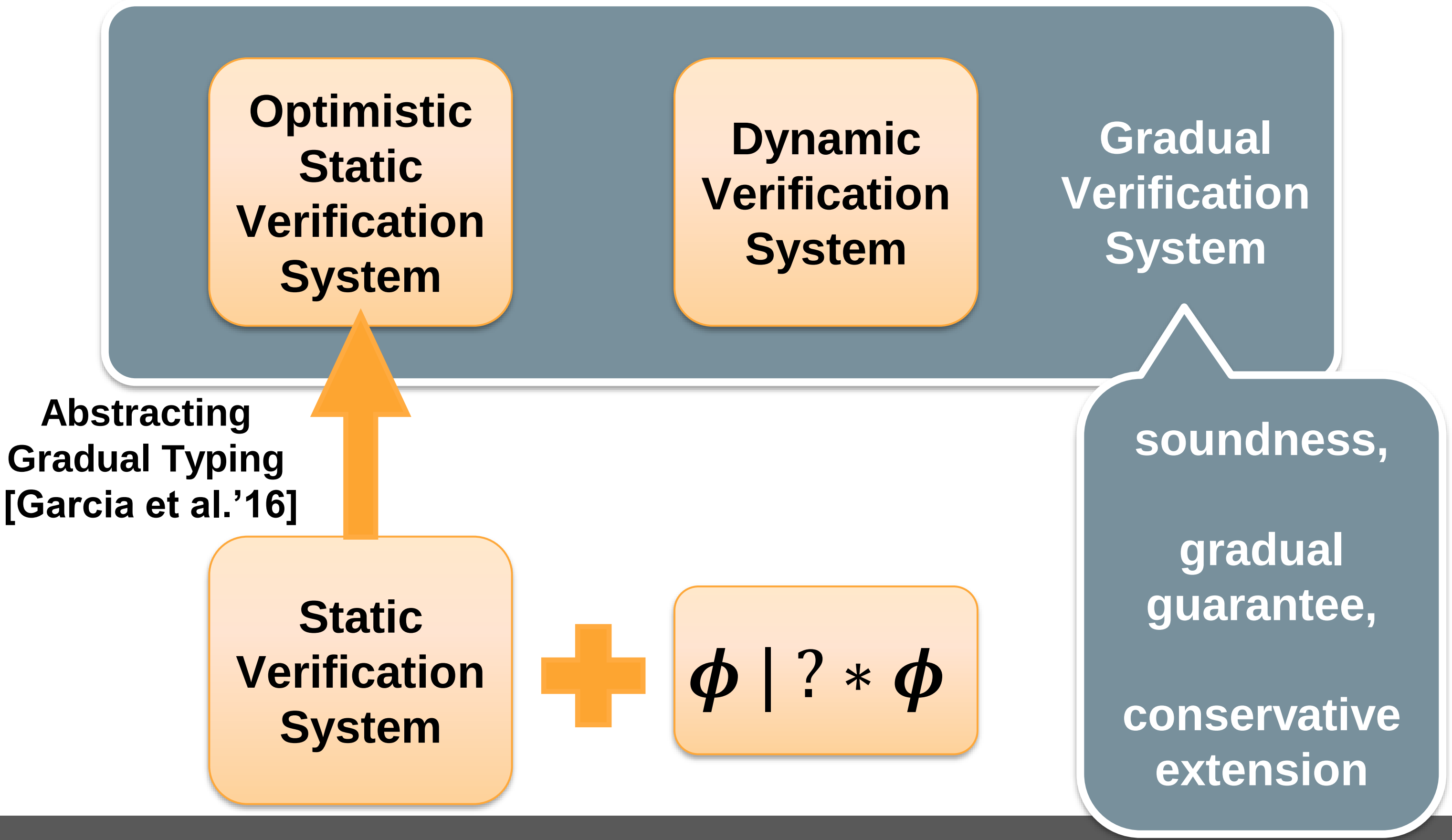
```
int findMax(Node l)
  requires l != null
  ensures max(result,l) && contains(result,l)
{
  int m := l.val;
  Node curr := l.next;
     FOLDS/UNFOLDS
  while(curr != null)  LOOP INVARIANTS   {
    if(curr.val > m) { m := curr.val; }
    curr := curr.next;
       FOLDS/UNFOLDS
          LEMMAS
  }
     FOLDS/UNFOLDS
  return m;
}
```

Gradual verification allows developers to deal with specification cost incrementally

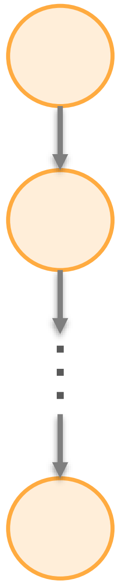
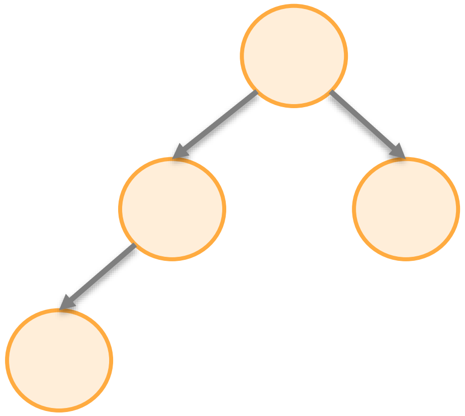
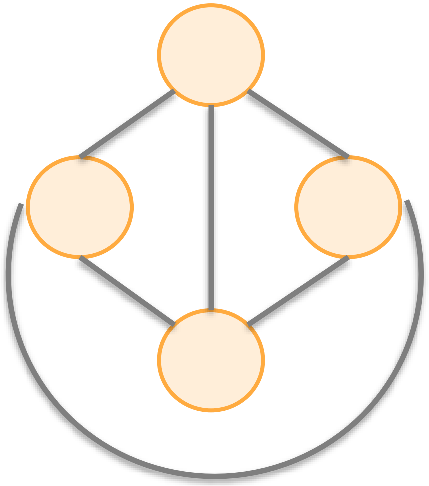
- without unnecessary effort
- with immediate feedback

by leveraging static & dynamic verification techniques

Gradual Verification Framework [Bader et al.'18]


Optimistic Static Verification System
Dynamic Verification System
Gradual Verification System
Abstracting Gradual Typing [Garcia et al.'16]
Static Verification System
$\phi \mid ? * \phi$
soundness,
gradual guarantee,
conservative extension

[Wise et al.'20] extends [Bader et al.'18] with Recursive Heap Data Structures

Lists	Trees	Graphs
		

Limitation: Abstract Theoretical Definitions

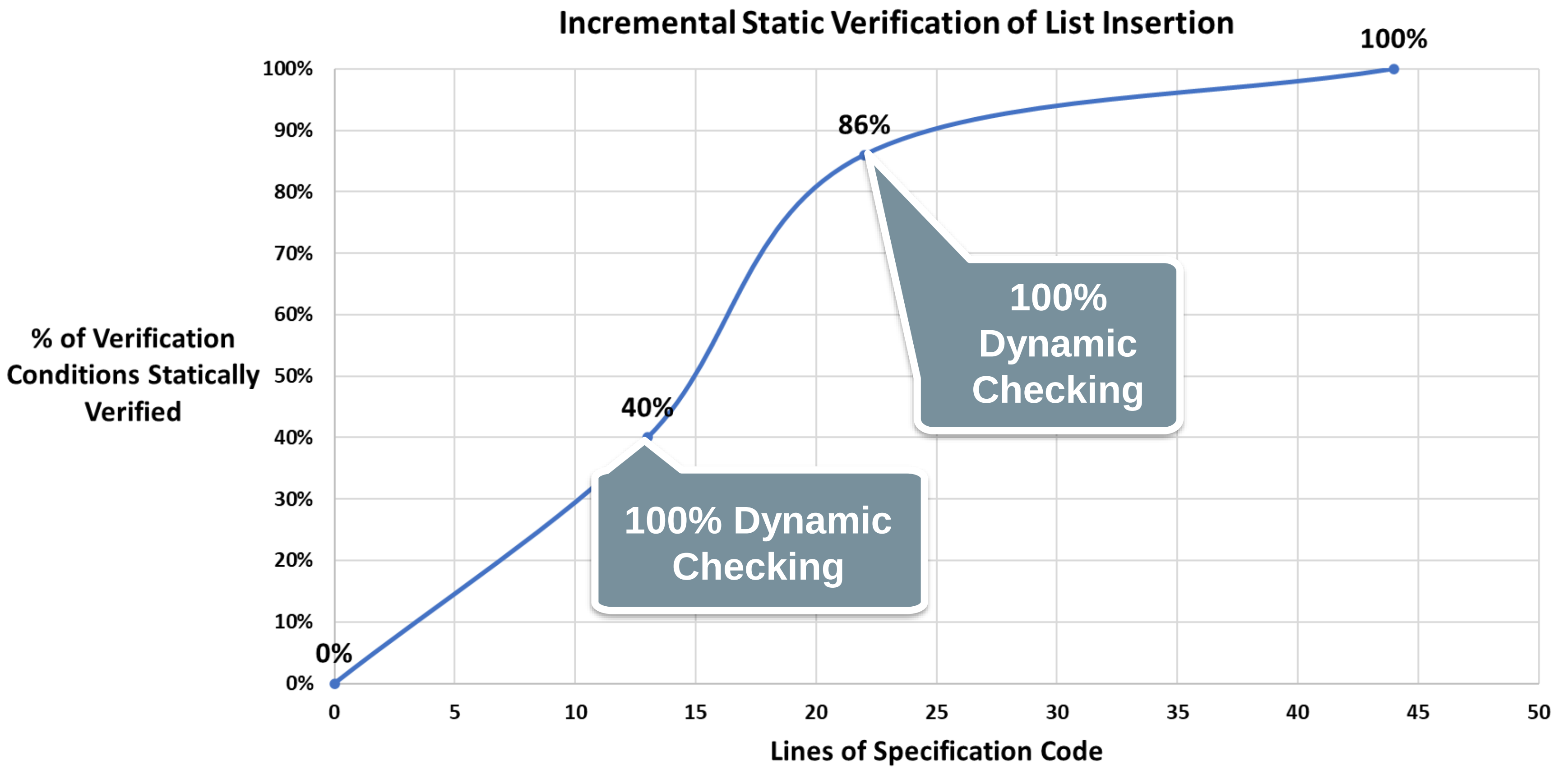
$$\widetilde{WLP}\,(\ldots,\tilde{\phi}) = \alpha(\,\{\,\boxed{max_{\Rightarrow}\{\ldots\}}\mid\ \ldots\ \})$$

$$\alpha(\bar{\phi}) = \boxed{min_{\sqsubseteq}\{\ldots\}}$$

Can we implement important abstract definitions?

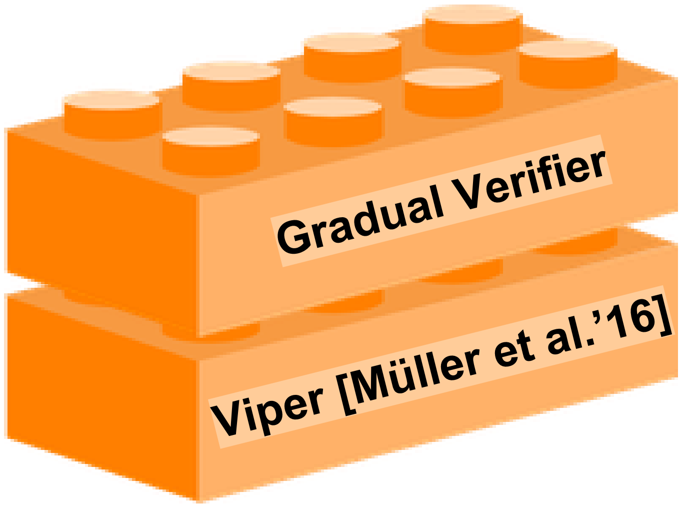
Can our implementation smoothly support the trade-off between static & dynamic checking?

Limitation: Eliminating Dynamic Checks Not Supported



Gradualizing the Viper Static Verifier

Implicit Dynamic Frames (IDF) [Smans et al.'09]



Accompanying PhD Thesis

Abstract Predicates [Parkinson et al.'05]

Symbolic Execution

Theory Research Questions

[RQ1] Is our verifier sound?

Prove our <u>verifier design</u> is *sound*.

[RQ2] Does our verifier support incrementality?

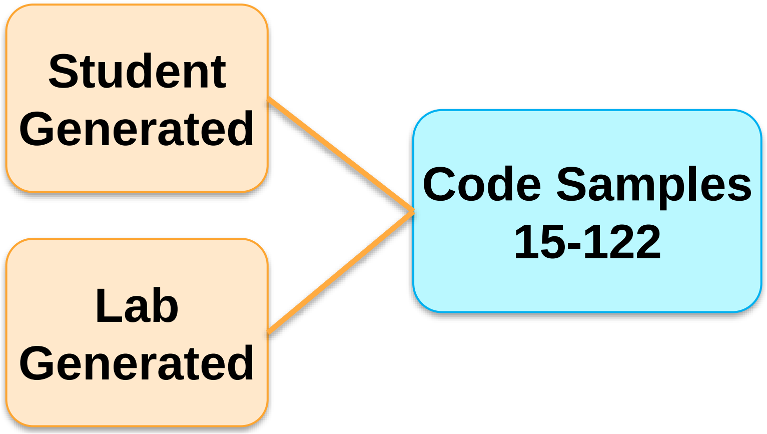
Prove our <u>verifier design</u> adheres to the *gradual guarantee.*

Empirical Research Questions: Exploring Trade-off Between Static & Dynamic Checking

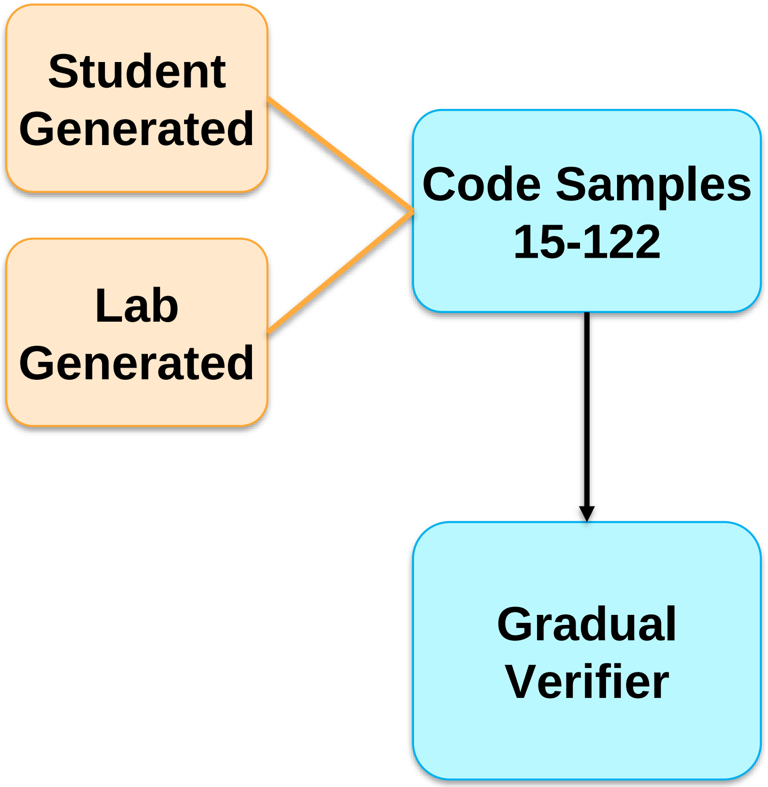
[RQ1] As the lines of correct specification code in programs containing recursive heap data structures increase/vary, what trends emerge from the percentage of VCs verified statically vs dynamically?

[RQ2] As the lines of correct specification code in programs containing recursive heap data structures increase/vary, what trends emerge from how long it takes to dynamically verify the program?

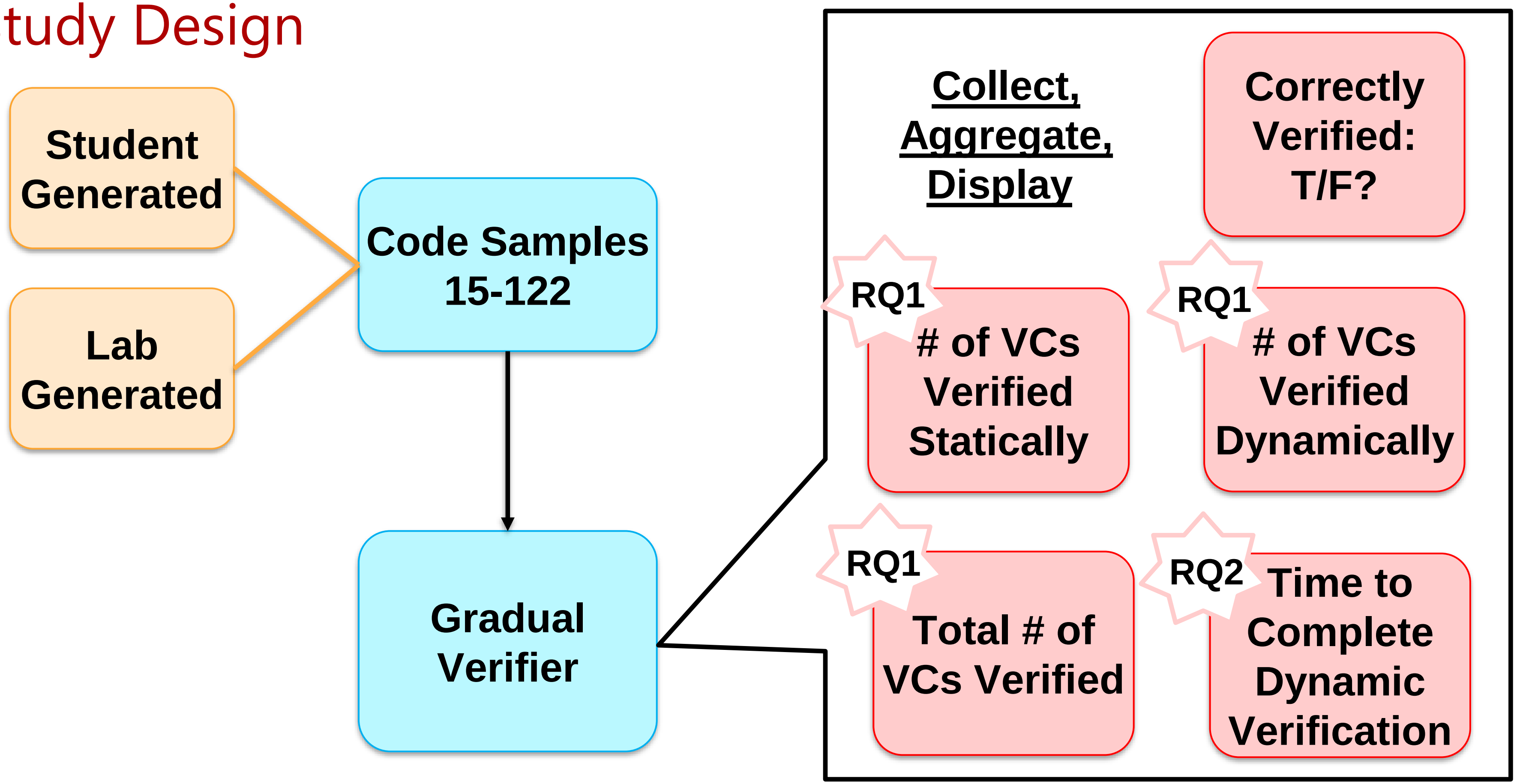
Study Design



Study Design



Study Design



Incremental static verification is made possible with Gradual Verification

Prior Work Limitations

1. Theoretical definitions

2. 100% Dynamic checking

Solution

Designing & implementing symbolic execution based gradual verifier

Current & Future Work

- Prototype implementation
- Proofs: soundness, gradual guarantee
- Empirical Study: static & dynamic trade-off