Foundationally Sound Annotation Verifier

via Control Flow Splitting

Litao Zhou, Shanghai Jiao Tong University

Dec 7, 2022, Auckland, New Zealand

Background: How programs are verified

- **Interactive Program Verifiers**:
  write formal specifications and proofs in a theorem prover
  - ✓ Foundationally sound
  - ✓ Rich assertion language
  - ✓ Flexible proof strategies
  - ✗ Correctness properties are not clear from proof script

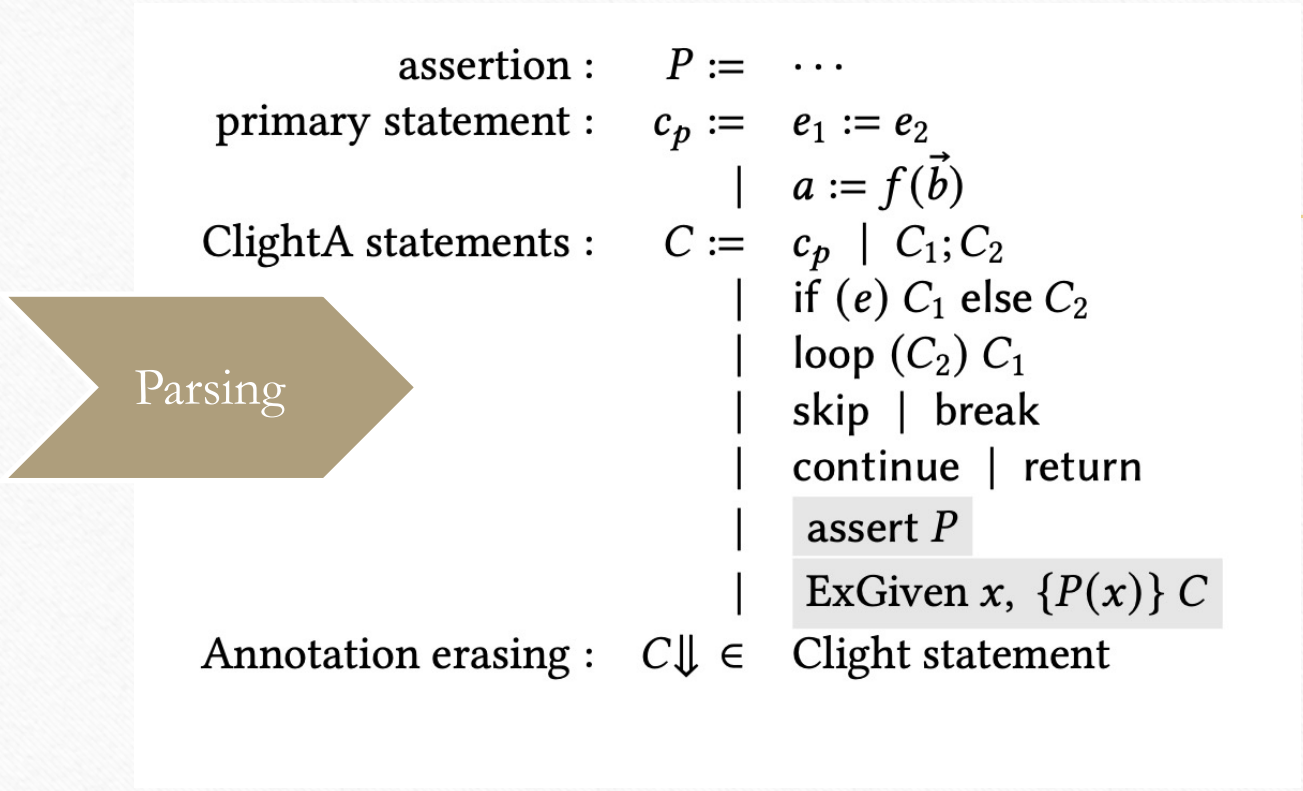
Background: How programs are verified

- **Annotation verifiers**:
  by writing annotations in the source code
  - ✓ More automation
  - ✓ Compared with formal proof scripts, **annotating programs with assertions** is a more straightforward way to demonstrate a program is correct
  - ✗ Foundational soundness proof is often lacked
- **VST-A:** a foundationally sound annotation verifier

VST-A workflow

- 1. Users annotate C programs with function specifications and assertions as **comments**

```
1   struct list {
2     unsigned head;
3     struct list *tail;
4   };
5
6   struct list *reverse (struct list *p) {
7   /*@ With  l,
8         Require  ll ([[p]], l)
9         Ensure  ll ([[ret]], rev(l)) */
10    struct list *w, *t, *v;
11    w = NULL; v = p;
12    while (v) {
13  /*@ Assert ∃ l1 c x l2'.
14          l = rev(l1) x l2'  ∧  [[v]] ↦ (x, c)
15          * ll ([[w]], l1)  *  ll (c, l2')  */
16      t = v->tail; v->tail = w;
17      w = v; v = t;
18    }
19    return w;
20  }
```

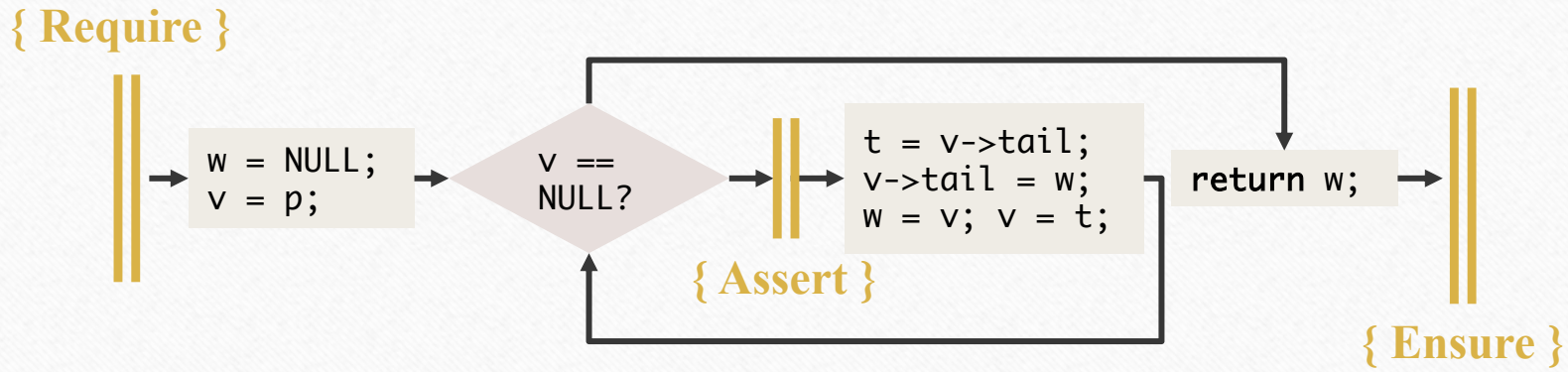


VST-A workflow

- 2. Annotated programs are parsed into **ClightA AST** definitions in Coq

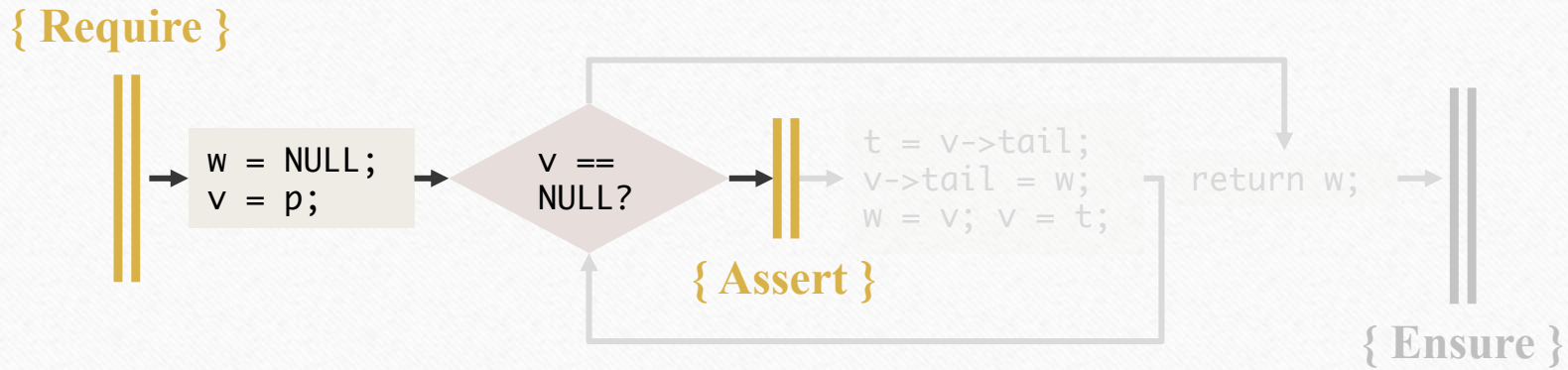
The split function

- Reduce the verification problem of the whole program into smaller **straight-line Hoare triples**



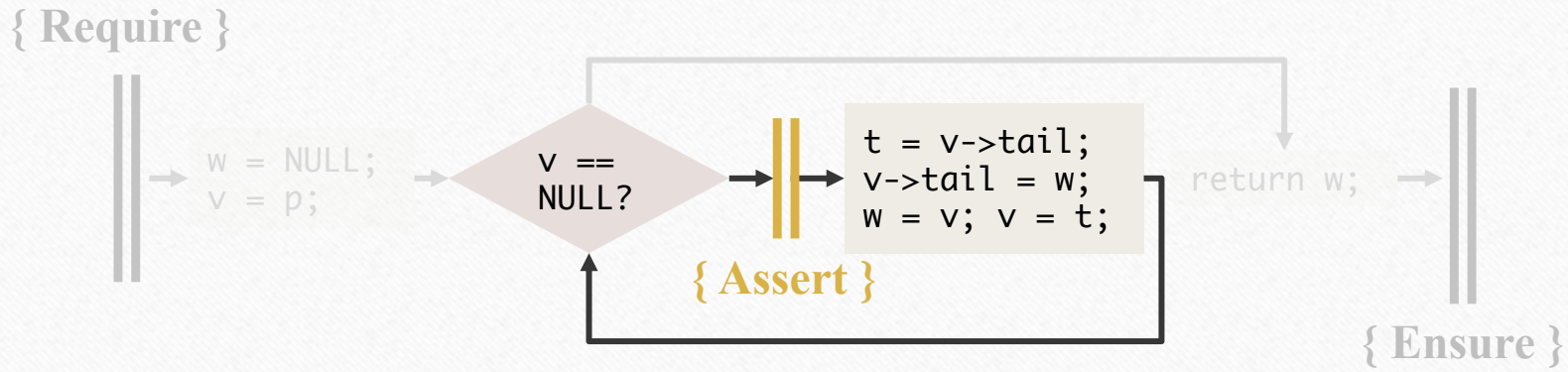
The split function

- To verify the whole program's Hoare triple, it is enough to verify the following **(1/4)** straightline Hoare triples.



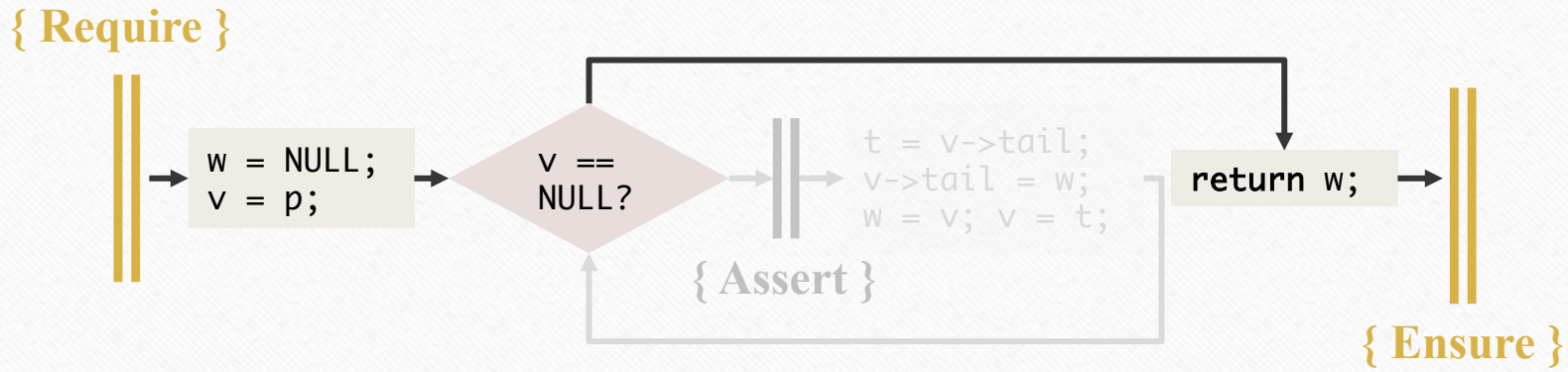
The split function

- To verify the whole program's Hoare triple, it is enough to verify the following **(2/4)** straightline Hoare triples.



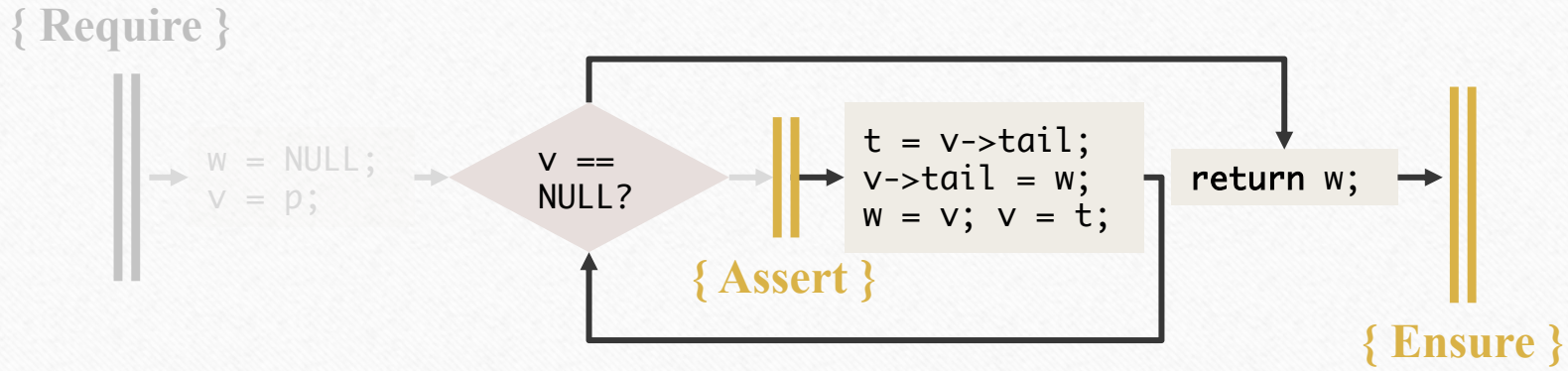
The split function

- To verify the whole program's Hoare triple, it is enough to verify the following **(3/4)** straightline Hoare triples.



The split function

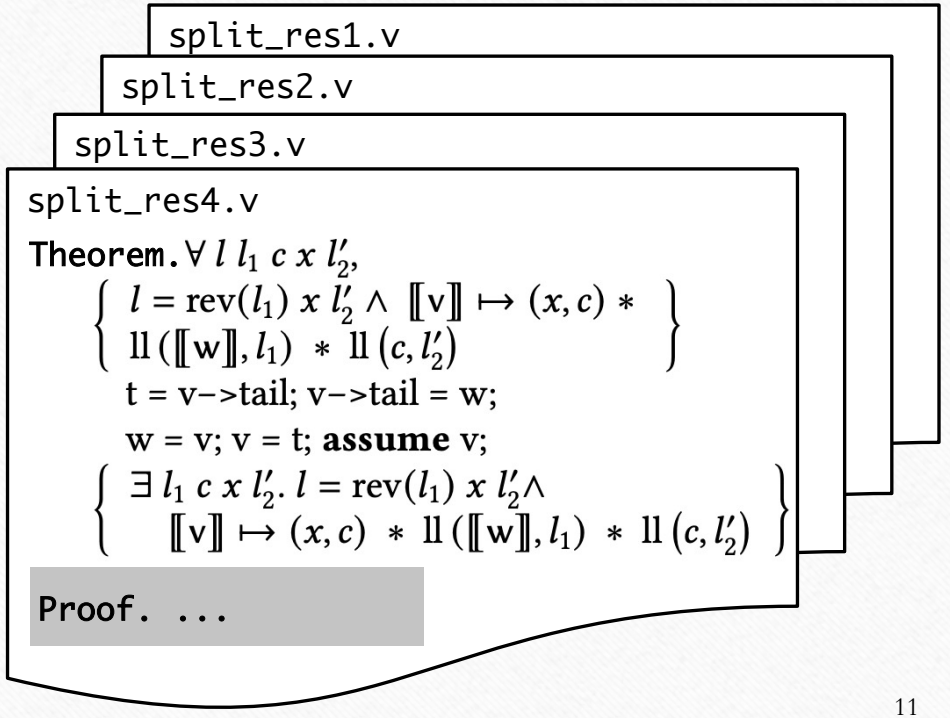
- To verify the whole program's Hoare triple, it is enough to verify the following **(4/4)** straightline Hoare triples.
- **Control flow paths separated by assertions**



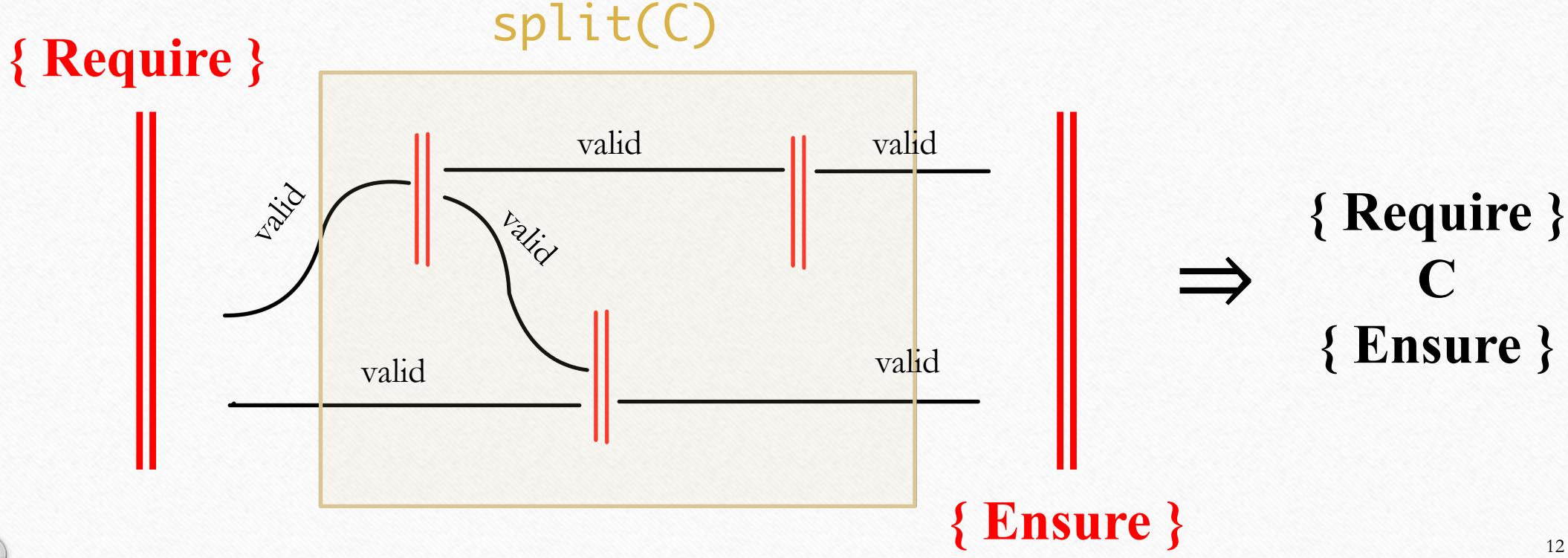
VST-A workflow

- 3. A set of **straightline Hoare triples** are automatically computed and printed into separate Coq files
- 4. Users are left to prove residual proof goals that are not checked automatically.

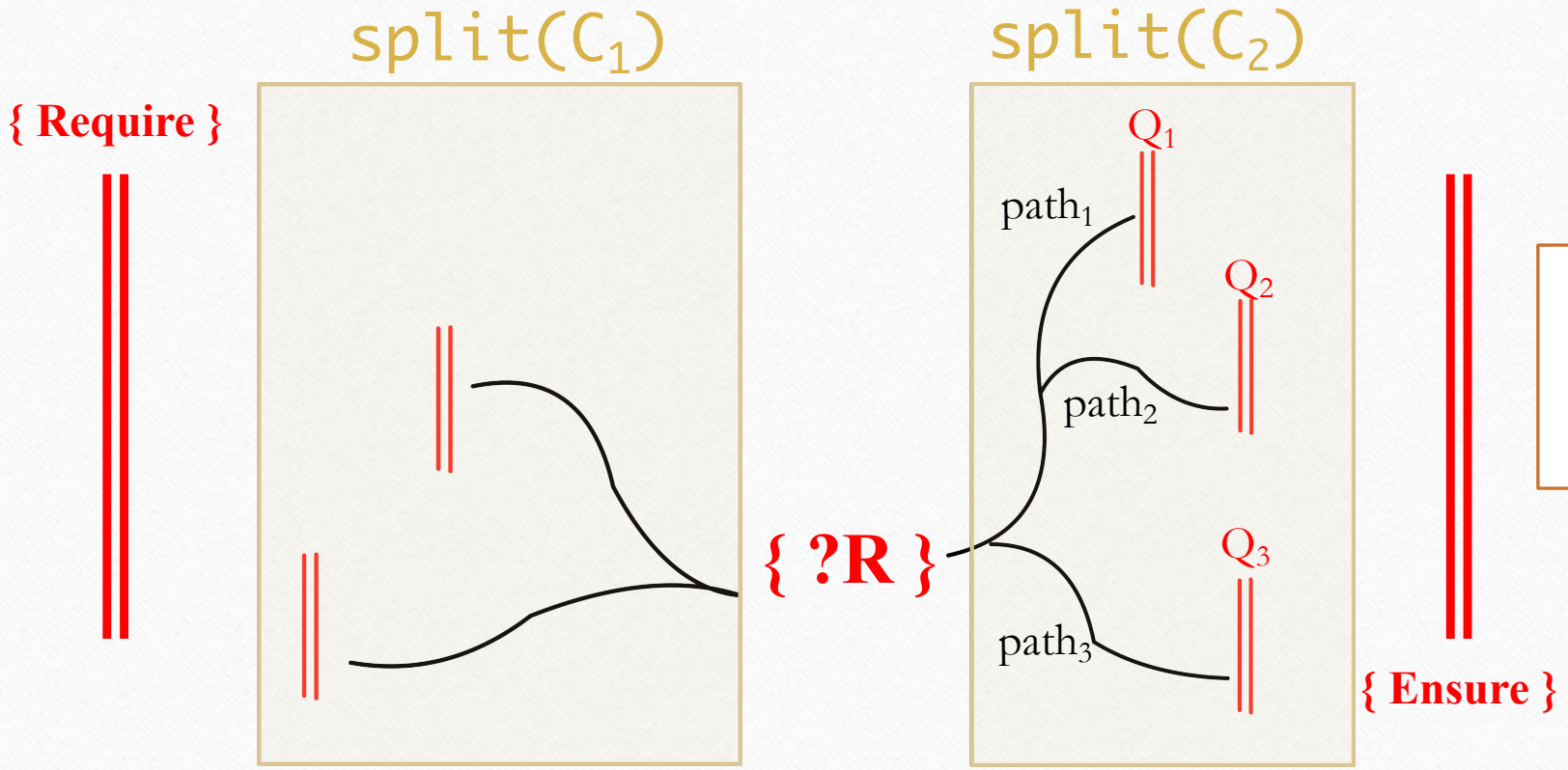
Proved sound split function



Soundness of split



Soundness of split (sequential case)



$$R = wp(path_1, Q_1) \wedge wp(path_2, Q_2) \wedge wp(path_3, Q_3)$$

Features of VST-A

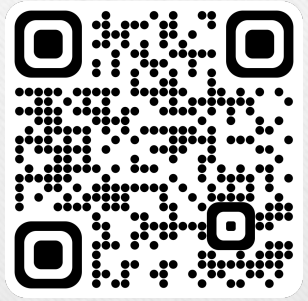
- ✓ Correctness proofs are described intuitively by inserting assertions
- ✓ Rich assertion languages and foundational soundness of VST
- ➢ Assertions can be inserted in a flexible way
  e.g. annotating loop structures with invariants is not compulsory
- ➢ Incremental verification for incremental program development
  i.e. changing part of the program only requires proof recompilation for the changed paths
- ✗ Currently only supports sequential programs and requires precise specification for callee functions
  due to the need for conjunction rule in the soundness proof

Summary

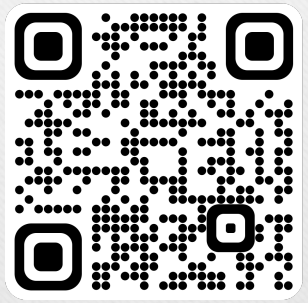
- **A novel framework for program verification**, based on the idea of reducing large program proofs to simpler verification goals
- **A formal language for annotated programs, ClightA**, that not only introduces assertions but also addresses logical variables in the verification context
- **A control-flow-based verification splitting algorithm**, implemented in Coq and proved sound w.r.t. the VST program logic

Thank you!

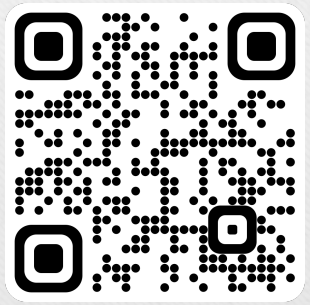
Litao Zhou, Shanghai Jiao Tong University, China



[poster]



[artifact]



[report]