My First NSF Proposal(s)

Shan Lu

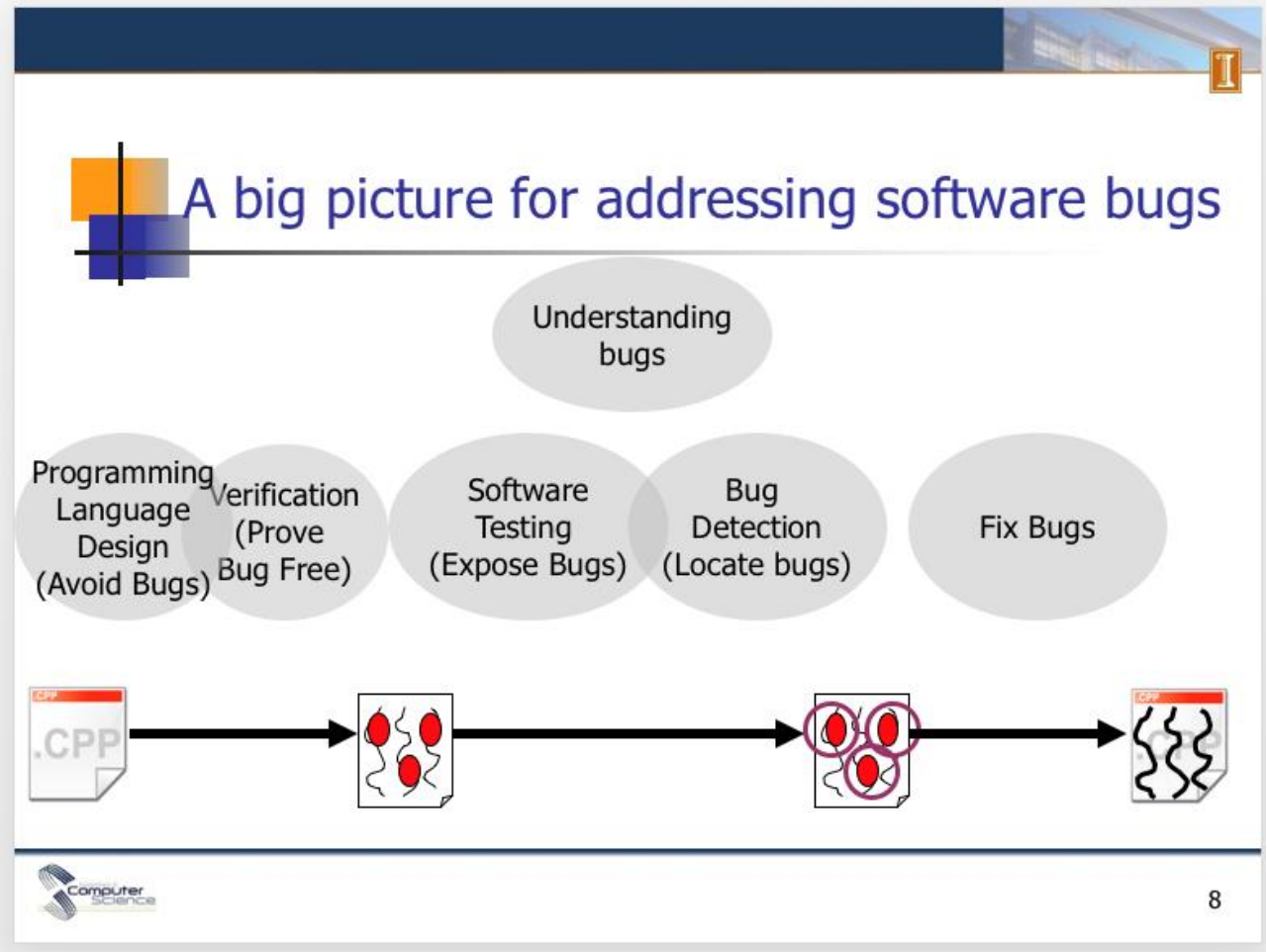
THE UNIVERSITY OF CHICAGO

My Time line

- **Dec. 2008**, thesis defense @ Univ. of Illinois
  - "Understanding, Detecting, and Exposing Concurrency Bugs"
- **Jan. 2009**, starts @ Univ. of Wisconsin
- **Jul. 2009**, should I write a proposal or paper or both?
- **Nov. 2009**, paper accepted by ASPLOS
  - "ConMem: Detecting Severe Concurrency Bugs through an Effect-Oriented Approach"
- **Dec. 2009**, submit my first NSF proposal (small, single PI)
  - "Fighting Concurrency Bugs through Effect-Oriented Approaches"
- **July 2010**, submit my CAREER proposal
  - "CAREER: Combating Performance Bugs in Software Systems"

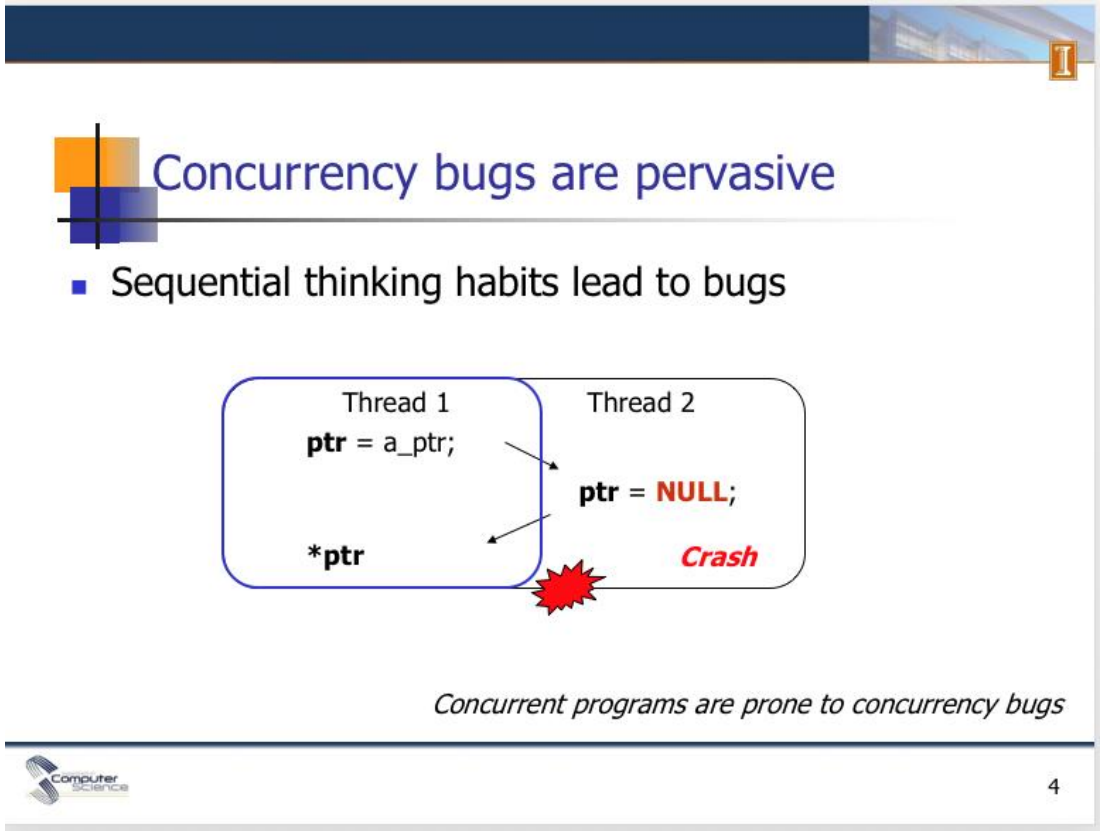
My thesis

- Understanding, Detecting, and Exposing Concurrency Bugs



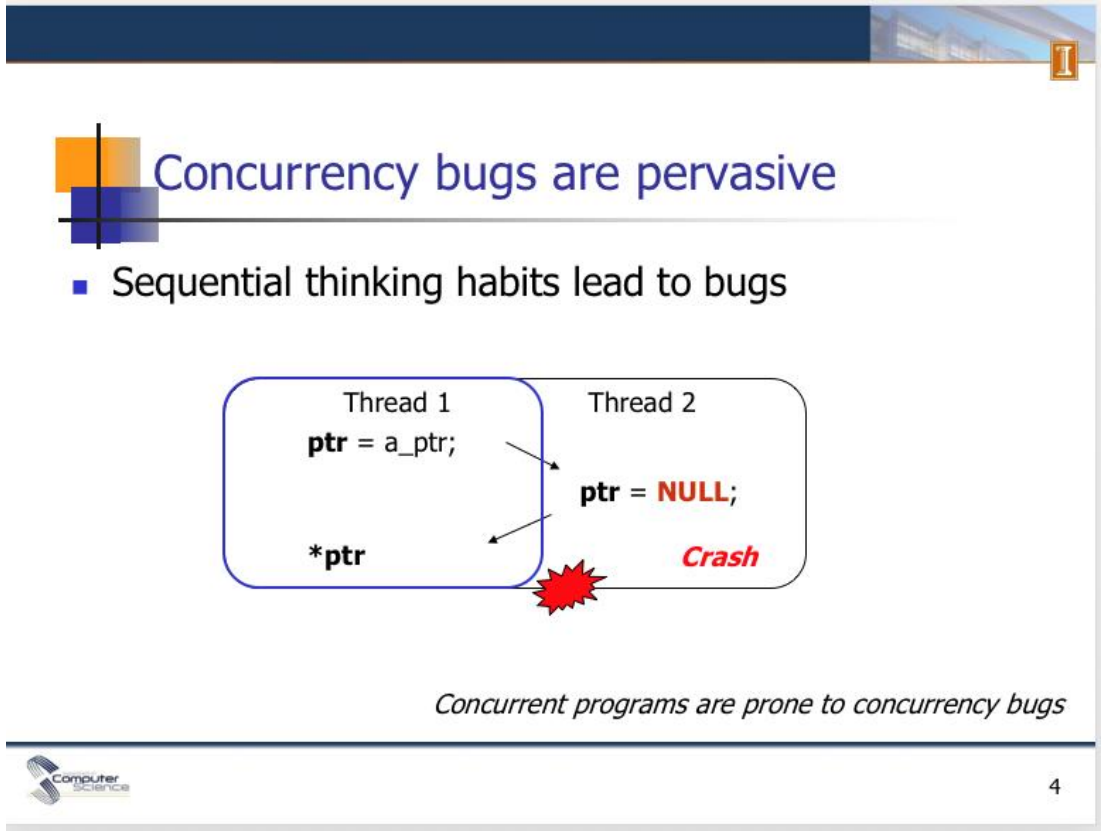
My thesis

- Understanding, Detecting, and Exposing Concurrency Bugs
  - What are concurrency bugs? data races, atomicity violations, …
  - Which code regions need to be atomic?



What should I do? Spring, 2009

- "I should do something different from what I did as a student"
- "should I?"



Proposal or paper? Summer, 2009

My paper submission August, 2009

ConMem: Detecting Severe Concurrency Bugs through an Effect-Oriented Approach

Wei Zhang Chong Sun Shan Lu
Computer Sciences Department, University of Wisconsin– Madison
{wzh,chong,shanlu}@cs.wisc.edu

My paper submission August, 2009

ConMem: Detecting Severe Concurrency Bugs through an Effect-Oriented Approach

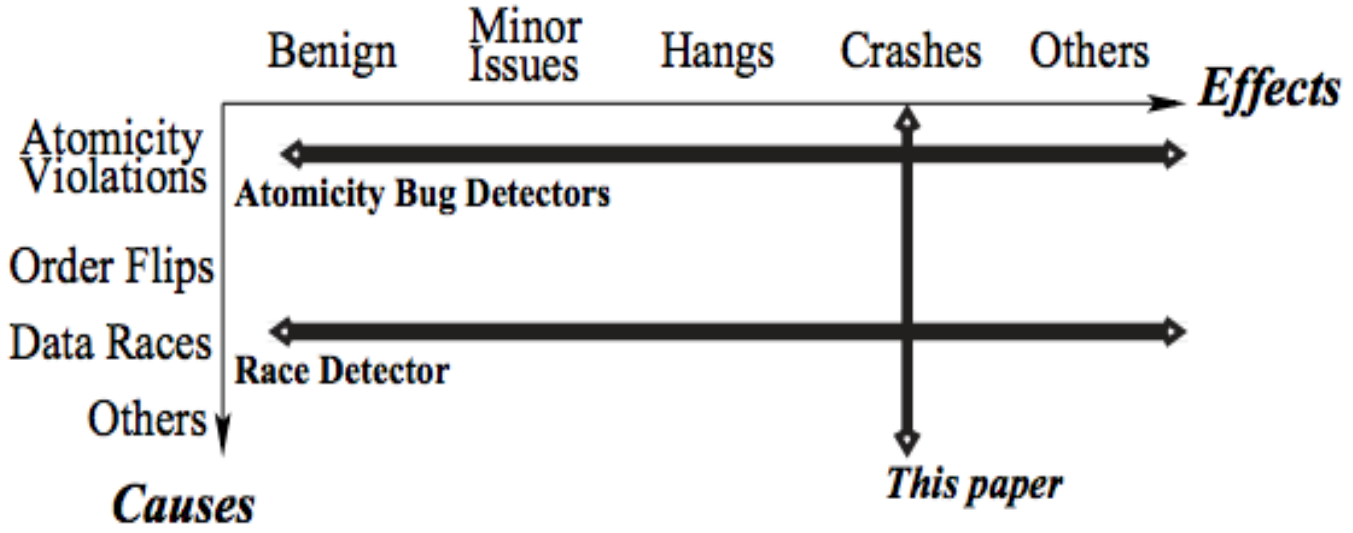


Figure 1: A conceptual two-dimension interleaving space

ASPLOS'10, March 13–17, 2010, Pittsburgh, Pennsylvania, USA.

Proposal or not?

December, 2009

So what?

My first proposal (small) Dec. 2009

- It starts from my ASPLOS work

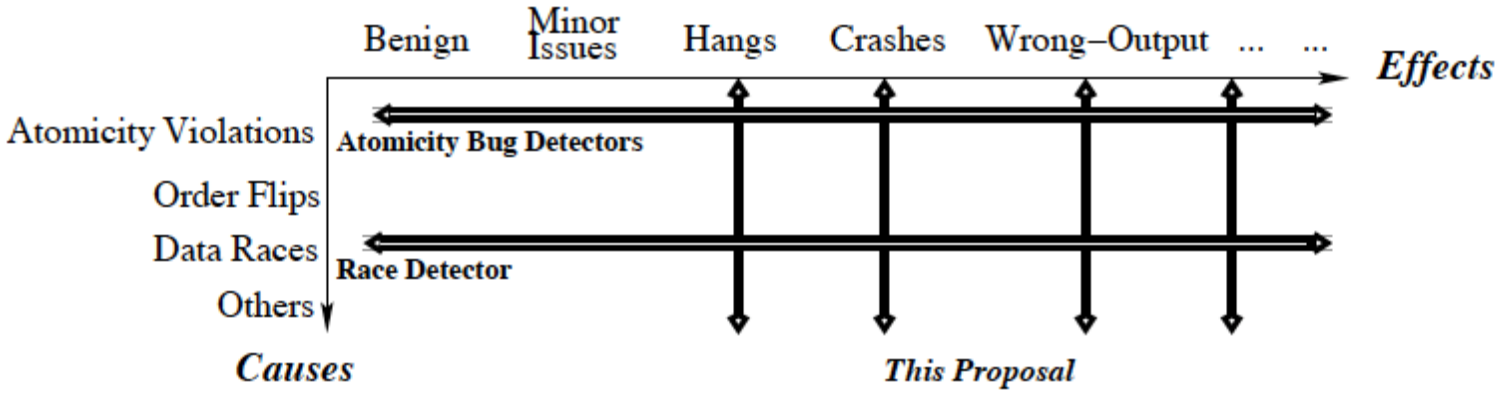


Figure 5: A conceptual two-dimension interleaving space: the proposal explores vertical approaches

How to differentiate from my published work?

- It is much broader than my ASPLOS work

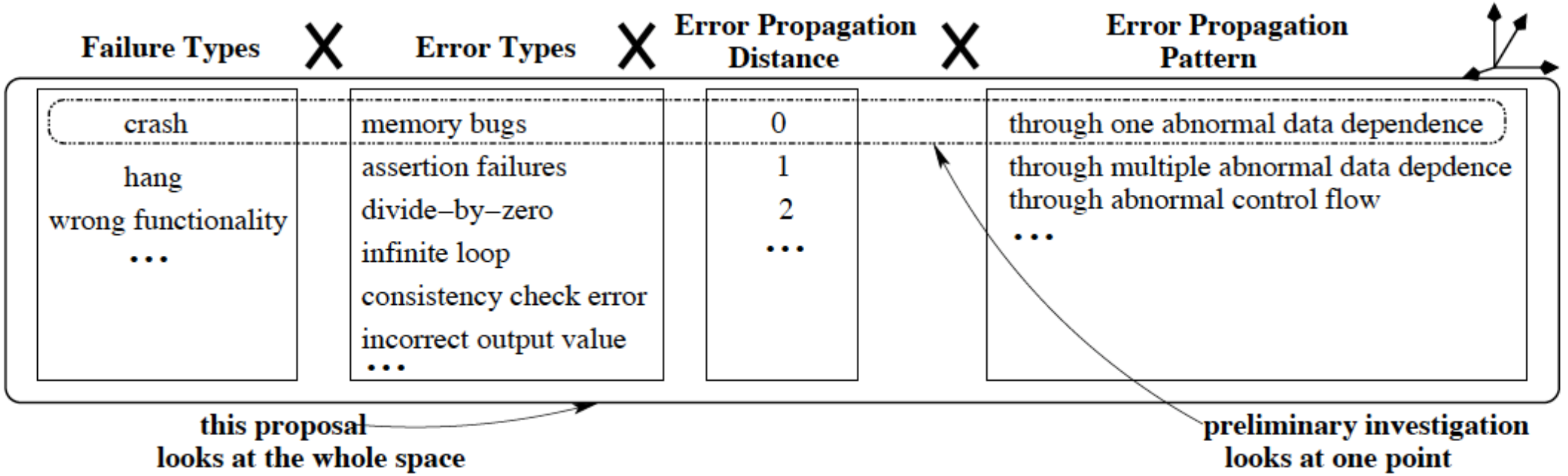


Figure 7: The space that this proposal will explore (the taxonomy will be defined and discussed in Section 3)

What tasks should I put there?

- A combination of ”it definitely will work” + “it might work”

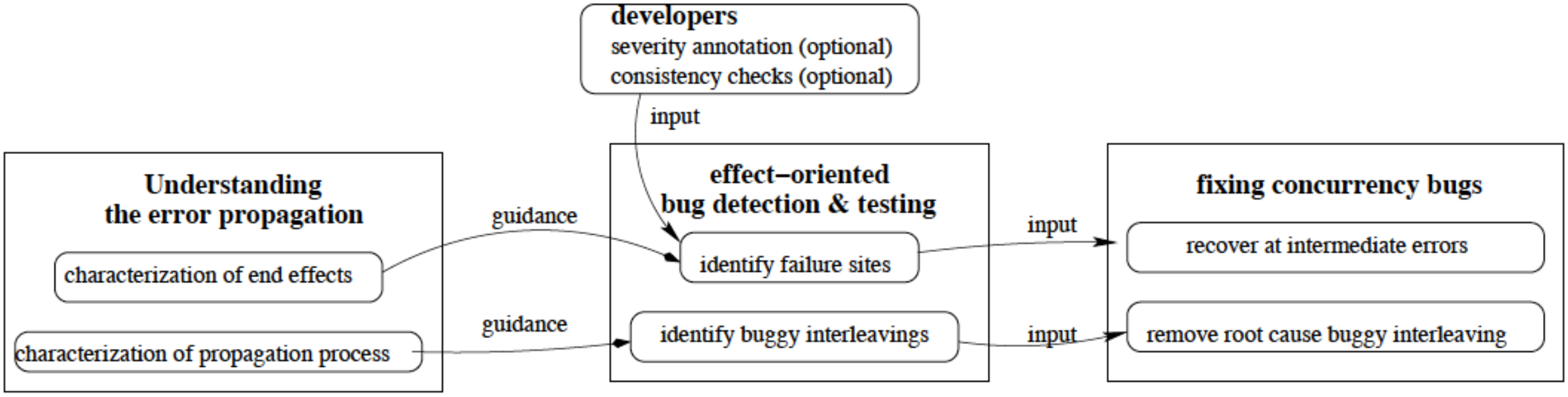


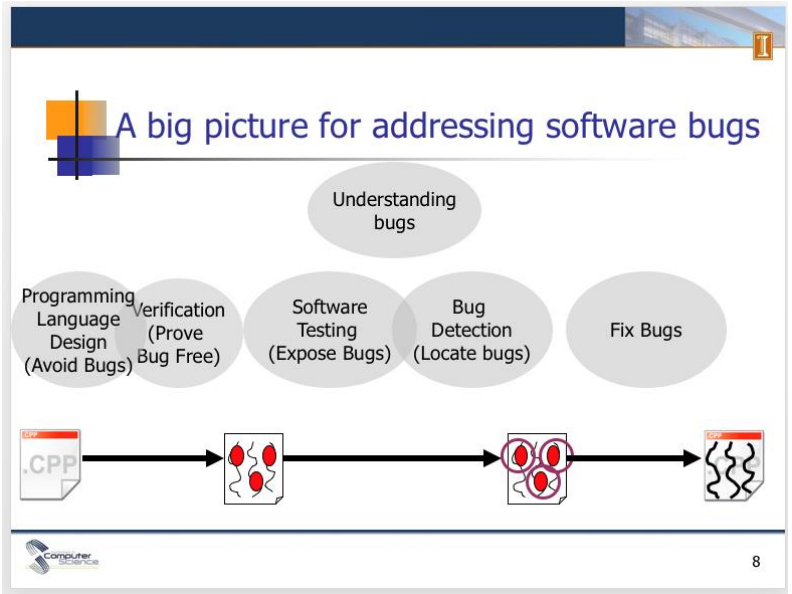
Figure 6: The overview of our proposal

What systems should I use for evaluation?

owned computing resources. To evaluate the proposed research, we will leverage open-source software, such as Apache, Mozilla, and their well maintained on-line bug databases. Especially, the proposed research will start with two sets of real-world concurrency bugs. One includes 105 bugs collected by the PI [36] from 4 C/C++ multi-threaded applications and one includes 93 concurrency bugs in Linux device drivers [60]. These two sets are among the most comprehensive benchmarks in the research community and have been used to evaluate many tools [51, 60, 61, 78]. We will further extend the application set to include Java

What should I write for CAREER?

- Leverage past strength
- Different from Ph.D. work
- It is a **5**-year project



What should I write for CAREER? Jul. 2010

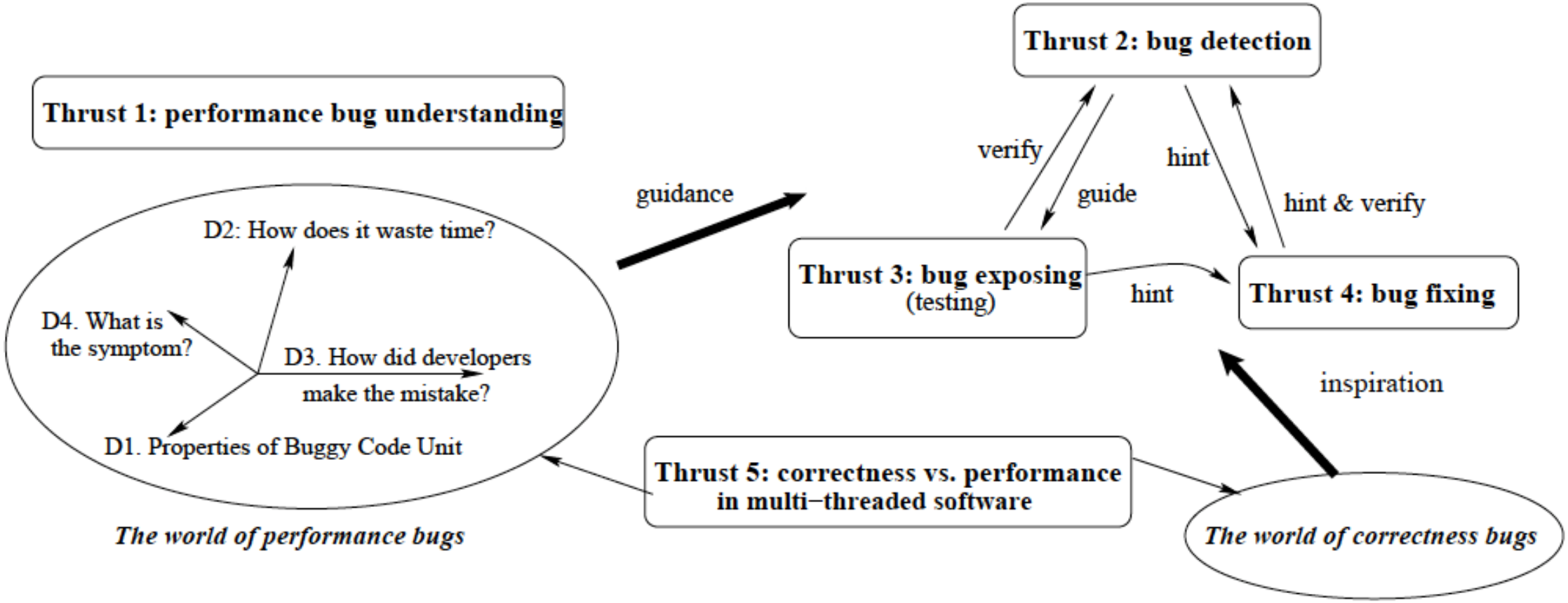


Figure 3: The overview of our proposal

What preliminary work should I do?

Preliminary study. We have conducted a preliminary study on **60 real-world performance bugs** randomly selected from widely used C/C++ and Java open-source applications (15 bugs from Apache, 15 bugs from MySQL, and 30 bugs from Mozilla). This preliminary study, together with the PI's previous empirical studies of correctness bugs [41, 46, 48], has helped us design the taxonomy for performance bugs. It also revealed many interesting examples and inspirations for our research.

What preliminary work should I do?

Characteristic-1: Most (52 out of 60 cases) buggy code units are time-consuming: some involve procedures (referred to as *heavy procedures*) that conduct heavy operations such as I/Os; and, some involve procedures (referred to as *frequent procedures*) that will execute many times, linear/quadratic to the input scale.
Characteristic-2: Most buggy code units waste computation resource in two ways: (1) generating un-needed

Characteristic-4: Most examined performance bugs did not catch developers' attention until they showed intolerable latency, or unscalable behavior, or much worse performance comparing with other software.
Characteristic-5: Most performance bugs (50 out of 60) caught developers attention when the application is under an input that can execute the buggy code region many times. This is different from most bug-triggering

Characteristic-7: There are two most common strategies for fixing performance bugs. Work skipping (23 out of 60): bugs were fixed by conditionally skip a code region. Simple API change (29 out of 60): bugs were fixed by small change to procedure-parameter (4 cases), replacing one or a sequence of procedures with an existing (16 cases) or newly added (9 cases) procedure.

What systems should I use for evaluation?

8 Evaluation Plan

To evaluate the proposed research, we will leverage open-source software, such as Apache, Mozilla, MySQL and their well maintained bug databases. Especially, we will start with the 60 real-world performance bugs in these three applications that the PI has already manually checked. We will further extend the bug set

The outcome of my proposals

- My concurrency bug proposal
  - ASPLOS'11, PLDI'11, OSDI'12, ASPLOS'13, ASPLOS'14, FSE'14

- My CAREER
  - PLDI'12, ICSE'13, CAV'13, OOPSLA'14, ICSE'15, ICSE'17, ASPLOS'18

- Span out to more projects funded by NSF CNS & CCF

Summary

- Do good research ←→ Write good proposal
- Have good graphs
- Have a mix of thrusts in the proposal
- Get advice from other people
- Don’t be scared