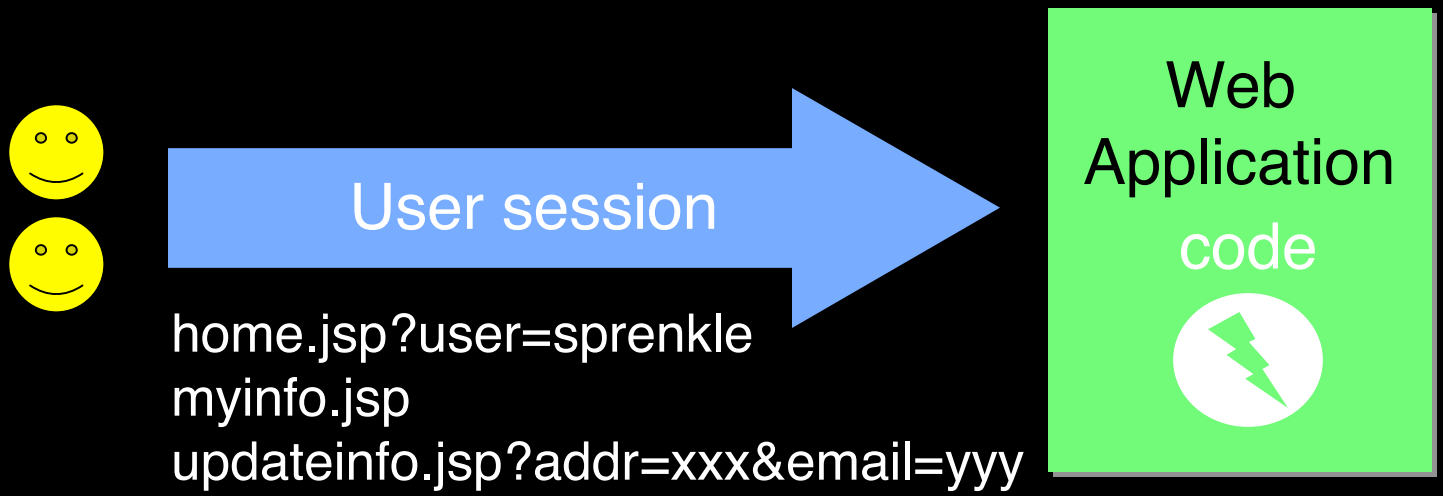
An Empirical Comparison of Test Suite Reduction Techniques for User-session-based Testing of Web Applications

Sara Sprenkle, Sreedevi Sampath,
Emily Gibson, and Lori Pollock
University of Delaware

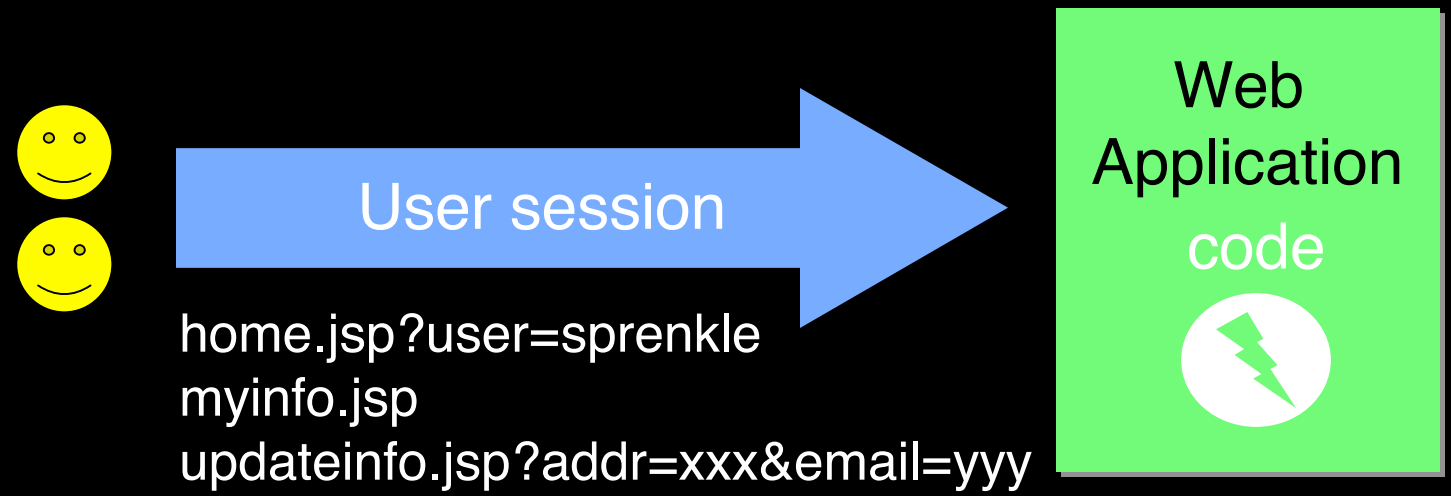
Amie Souter
Sarnoff Corporation

User-session-based Testing



- User sessions as test cases
  - Real usage of Web applications
  - Beta/maintenance testing phases

User-session-based Testing

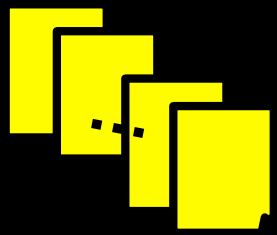


- User sessions as test cases
  - Real usage of Web applications
  - Beta/maintenance testing phases
- Problem: potentially collect too many user sessions to test practically

Suite Reduction: State of the Art

- Goal: maintain the original suite's effectiveness
  - e.g., code coverage, fault detection
- Approaches to Reduction
  - Requirements-based
  - Concept-based
- Paper's Focus: Empirical Study
  - What are the techniques' tradeoffs?
  - Which is the most practical technique?

Requirements-based Reduction



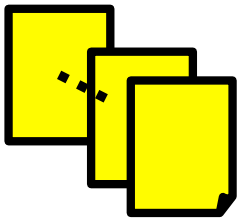
Original test suite T

choose criterion, collect data, create requirements mapping for each test case

$r_1 \rightarrow t_0, t_2, \ldots, t_j$
$r_2 \rightarrow t_7$
…
$r_n \rightarrow t_1, t_9, \ldots, t_n$

r → set of test cases that meet requirement

reduce



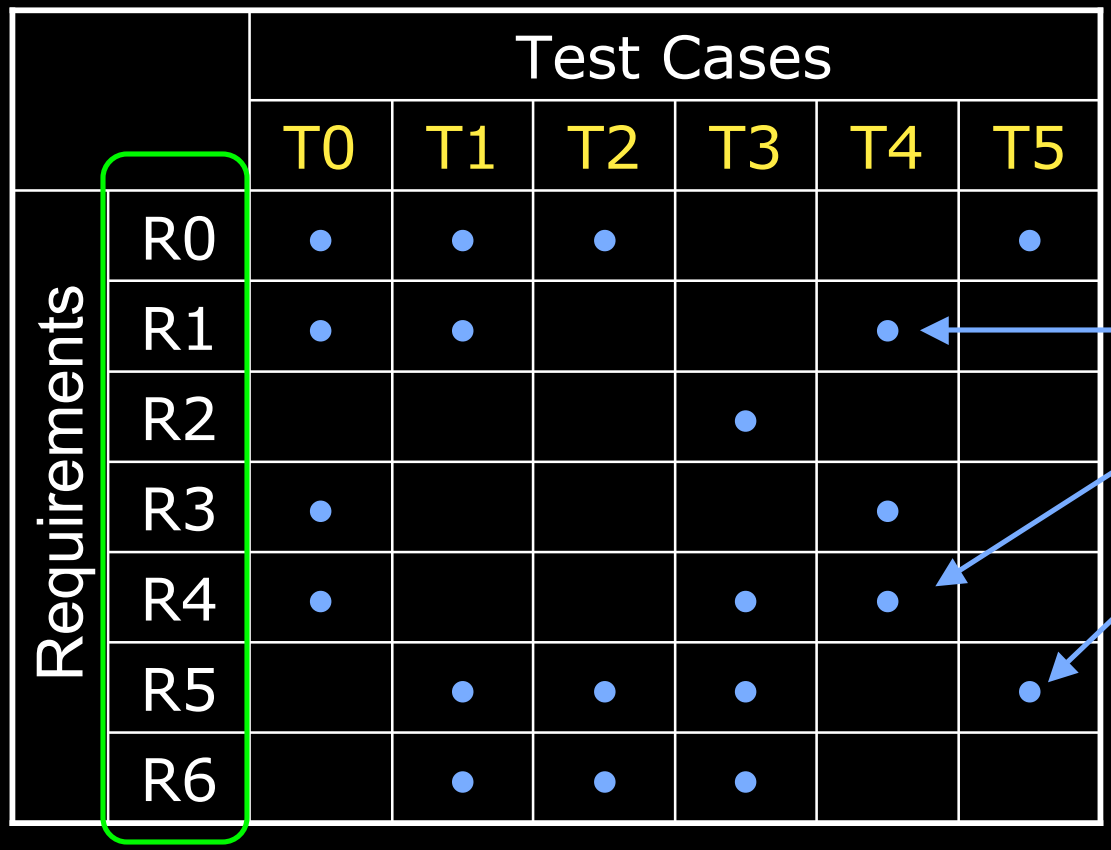
Reduced test suite T'

Test Case Selection Alternatives:

- Random
- Greedy
- Harrold, Gupta, Soffa [HGS93]

Requirements-based Mapping

- Selection: ensures that the reduced test suite covers all requirements



Requirements that the test case covers

Test Case Selection: Random

- Randomly choose next test case

Choosing T2
covers R0, R5, R6

		Test Cases					
		T0	T1	**T2**	T3	T4	T5
Requirements	R0	•	•	◆			•
	R1	•	•			•	
	R2				•		
	R3	•				•	
	R4	•			•	•	
	R5		•	◆	•		•
	R6		•	◆	•		

Test Case Selection: Random

- Randomly choose next test case

Choosing T2 covers R0, R5, R6

Choosing T1 also covers R1

		Test Cases					
		T0	**T1**	**T2**	T3	T4	T5
Requirements	R0	●	◆	●			●
	R1	●	◆			●	
	R2				●		
	R3	●				●	
	R4	●			●	●	
	R5		◆	●	●		●
	R6		◆	●	●		

Test Case Selection: Random

- Randomly choose next test case

Choosing T2 covers R0, R5, R6
Choosing T1 also covers R1
Choosing T0 covers R3, R4

		Test Cases					
		T0	**T1**	**T2**	T3	T4	T5
Requirements	R0	◆	•	•			•
	R1	◆	•			•	
	R2				•		
	R3	◆				•	
	R4	◆			•	•	
	R5		•	•	•		•
	R6		•	•	•		

Test Case Selection: Random

- Randomly choose next test case

Choosing T2 covers R0, R5, R6
Choosing T1 also covers R1
Choosing T0 covers R3, R4
Choosing T5 adds nothing

		Test Cases					
		T0	**T1**	**T2**	T3	T4	**T5**
Requirements	R0	●	●	●			◆
	R1	●	●			●	
	R2				●		
	R3	●				●	
	R4	●			●	●	
	R5		●	●	●		◆
	R6		●	●	●		

Test Case Selection: Random

- Randomly choose next test case

Choosing T2 covers R0, R5, R6

Choosing T1 also covers R1

Choosing T0 covers R3, R4

Choosing T5 adds nothing

Choosing T3 gets R2

		Test Cases					
		T0	**T1**	**T2**	**T3**	T4	**T5**
Requirements	R0	●	●	●			●
	R1	●	●			●	
	R2				◆		
	R3	●				●	
	R4	●			◆	●	
	R5		●	●	◆		●
	R6		●	●	◆		

Test Case Selection: Random

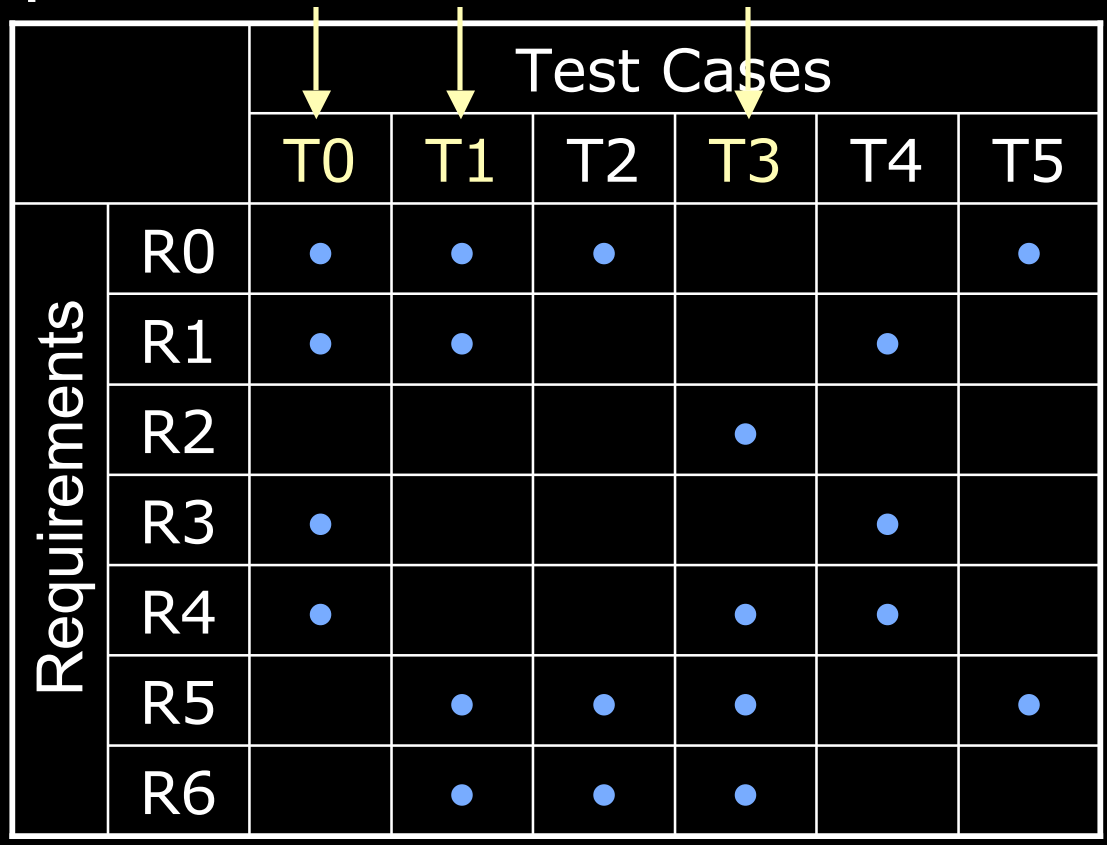
- Randomly choose next test case

Reduced test suite:
{T0, T1, T2, T3, T5}

		Test Cases					
		T0	**T1**	**T2**	**T3**	T4	**T5**
Requirements	R0	•	•	•			•
	R1	•	•			•	
	R2				•		
	R3	•				•	
	R4	•			•	•	
	R5		•	•	•		•
	R6		•	•	•		

Test Case Selection: Greedy

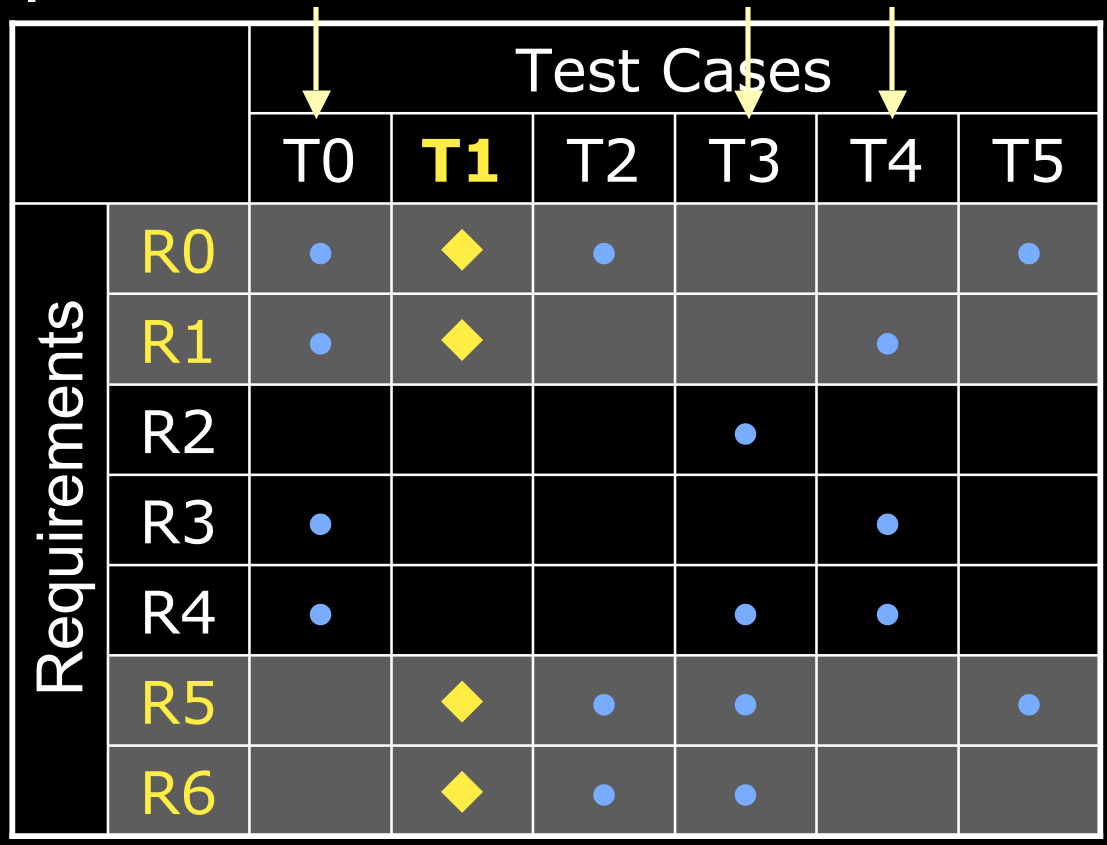
- Select next test case to cover most uncovered requirements



Test Case Selection: Greedy

- Select next test case to cover most uncovered requirements

Choosing T1 covers R0, R1, R5, R6



Test Case Selection: Greedy

- Select next test case to cover most uncovered requirements

Choosing T1 covers R0, R1, R5, R6

Choosing T3 covers R2, R4

		Test Cases					
		T0	**T1**	T2	**T3**	T4	T5
Requirements	R0	●	●	●			●
	R1	●	●			●	
	R2				◆		
	R3	●				●	
	R4	●			◆	●	
	R5		●	●	◆		●
	R6		●	●	◆		

Test Case Selection: Greedy

- Select next test case to cover most uncovered requirements

Choosing T1 covers R0, R1, R5, R6

Choosing T3 covers R2, R4

Choosing T4 covers R3

		Test Cases					
		T0	**T1**	T2	**T3**	**T4**	T5
Requirements	R0	●	●	●			●
	R1	●	●			◆	
	R2				●		
	R3	●				◆	
	R4	●			●	◆	
	R5		●	●	●		●
	R6		●	●	●		

Test Case Selection: Greedy

- Select next test case to cover most uncovered requirements

Reduce Suite:
{T1, T3, T4}

		Test Cases					
		T0	**T1**	T2	**T3**	**T4**	T5
Requirements	R0	•	•	•			•
	R1	•	•			•	
	R2				•		
	R3	•				•	
	R4	•			•	•	
	R5		•	•	•		•
	R6		•	•	•		

Test Case Selection: HGS

- Select next test case to cover the least-covered requirement

Cardinality: # of test cases that cover each requirement

		card	Test Cases					
			T0	T1	T2	T3	T4	T5
Requirements	R0	4	⊙	⊙	⊙			⊙
	R1	3	•	•			•	
	R2	1				•		
	R3	2	•				•	
	R4	3	•			•	•	
	R5	4		•	•	•		•
	R6	3		•	•	•		

Test Case Selection: HGS

- Select next test case to cover the least-covered requirement

Only T3 covers R2

		card	Test Cases					
			T0	T1	T2	T3	T4	T5
Requirements	R0	4	•	•	•			•
	R1	3	•	•			•	
	R2	1				•		
	R3	2	•				•	
	R4	3	•			•	•	
	R5	4		•	•	•		•
	R6	3		•	•	•		

Test Case Selection: HGS

- Select next test case to cover the least-covered requirement

Only T3 covers R2

Choosing T3 covers R2, R4, R5, R6

		card	Test Cases					
			T0	T1	T2	**T3**	T4	T5
Requirements	R0	4	●	●	●			●
	R1	3	●	●			●	
	R2	1				◆		
	R3	2	●				●	
	R4	3	●			◆	●	
	R5	4		●	●	◆		●
	R6	3		●	●	◆		

Test Case Selection: HGS

- Select next test case to cover the least-covered requirement

Only T3 covers R2

Choosing T3 covers R2, R4, R5, R6

R3 is covered by two test cases.

Choose the test case that covers most uncovered requirements, T0

Requirements		card	Test Cases T0	T1	T2	**T3**	T4	T5
	R0	4	•	•	•			•
	R1	3	•	•			•	
	R2	1				◆		
	R3	2	•				•	
	R4	3	•			◆	•	
	R5	4		•	•	◆		•
	R6	3		•	•	◆		

Test Case Selection: HGS

- Select next test case to cover the least-covered requirement

Only T3 covers R2

Choosing T3 covers R2, R4, R5, R6

R3 is covered by two test cases.

Choose the test case that covers most uncovered requirements, T0

			Test Cases					
		card	**T0**	T1	T2	**T3**	T4	T5
Requirements	R0	4	◆	•	•			•
	R1	3	◆	•			•	
	R2	1				•		
	R3	2	◆				•	
	R4	3	◆			•	•	
	R5	4		•	•	•		•
	R6	3		•	•	•		

Test Case Selection: HGS

- Select next test case to cover the least-covered requirement

Reduced Suite:
{T0, T3}

		card	Test Cases					
			T0	T1	T2	**T3**	T4	T5
Requirements	R0	4	•	•	•			•
	R1	3	•	•			•	
	R2	1				•		
	R3	2	•				•	
	R4	3	•			•	•	
	R5	4		•	•	•		•
	R6	3		•	•	•		

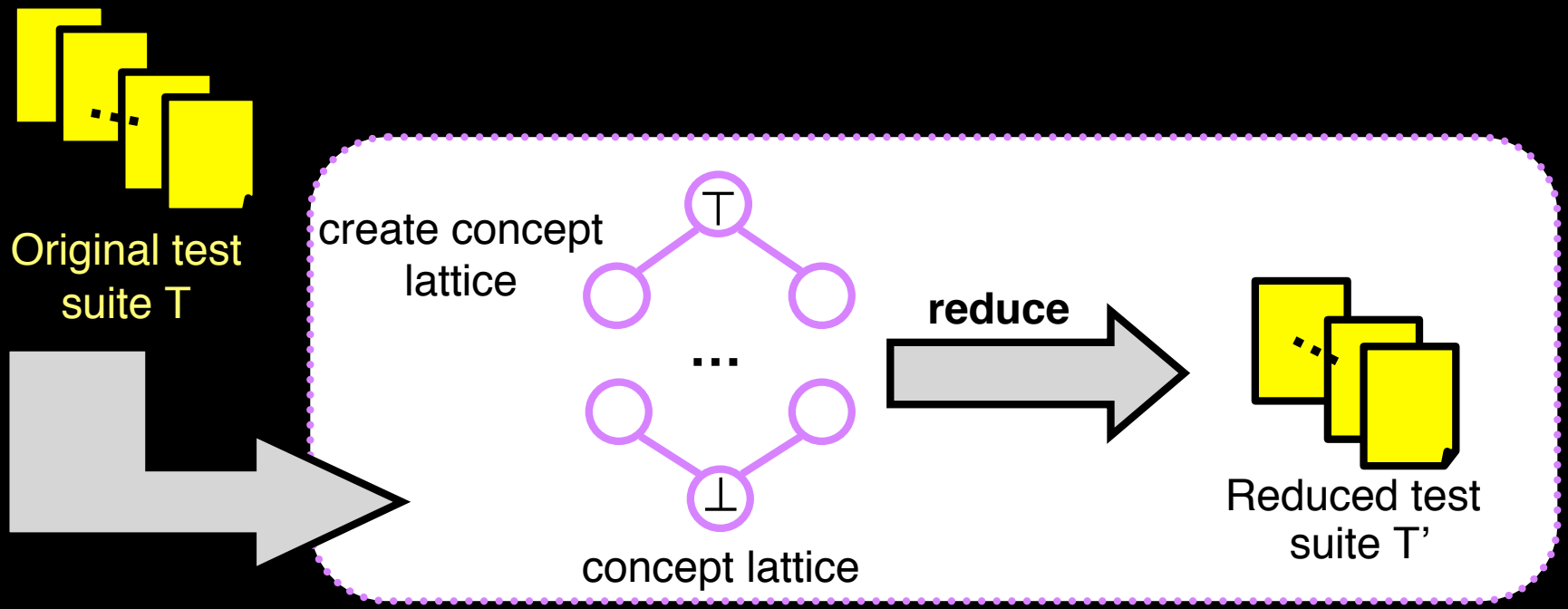
Test Case Selection Alternatives

- Select test case until suite satisfies criterion
  - Random: randomly
  - Greedy: test case that covers most requirements
  - HGS: test case that covers the least-covered requirement

Test Case Selection: Expected Tradeoffs

- Random
  - + Simple algorithm
  - – Larger reduced suites
- Greedy
  - + Closer to minimal reduced suite
  - – But not minimal
- HGS
  - + Approximates minimal covering reduced suite
  - - More complicated algorithm (tie-breaking strategies)

Concept-based Reduction

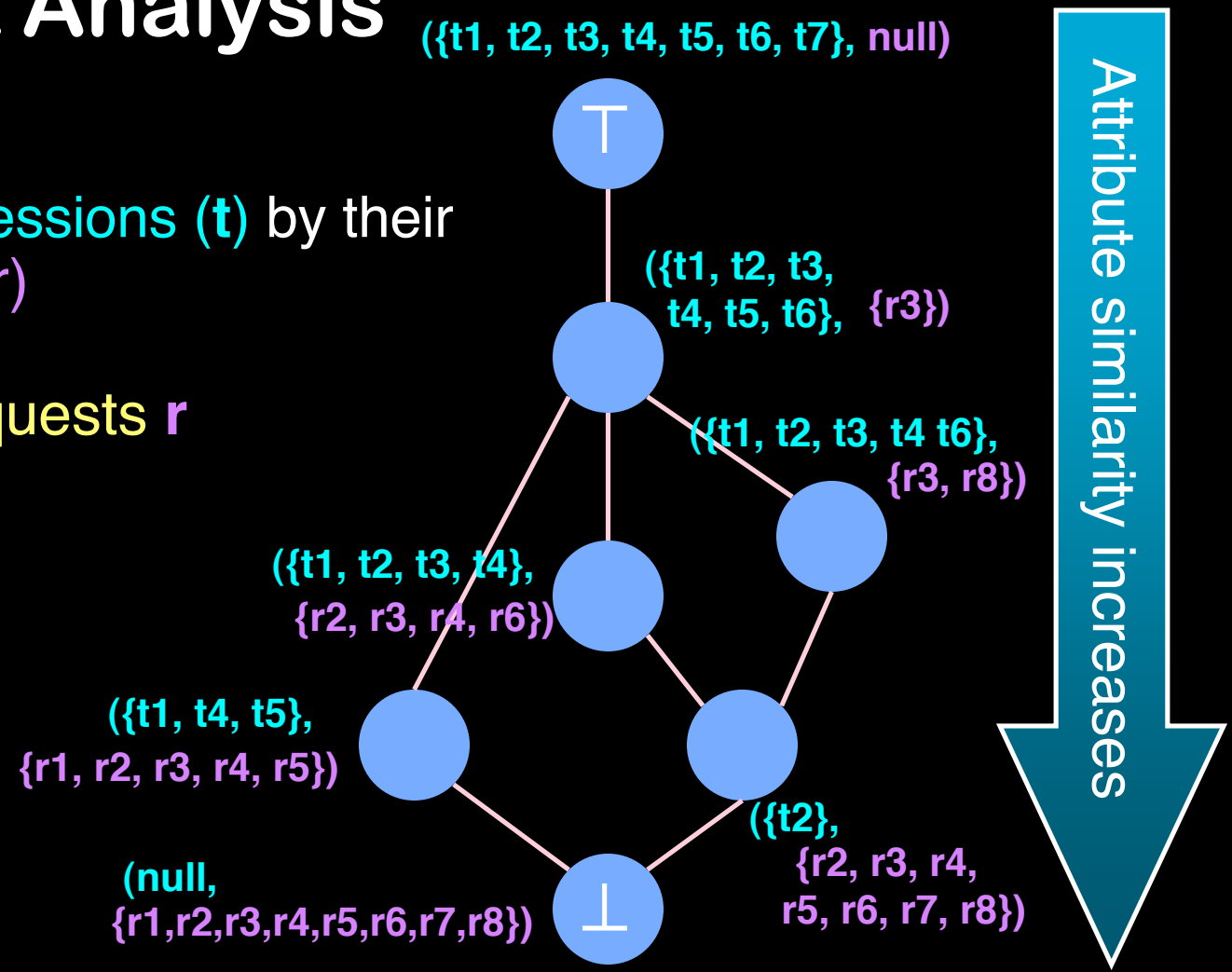


- Designed for Web applications [ASE04]
- Cluster user sessions by common URLs
  - Represent similar use cases
- Apply heuristic to select from clusters

Clustering User Sessions via Concept Analysis

Cluster user sessions (**t**) by their single URLs (r)

Relation: **t** requests **r**

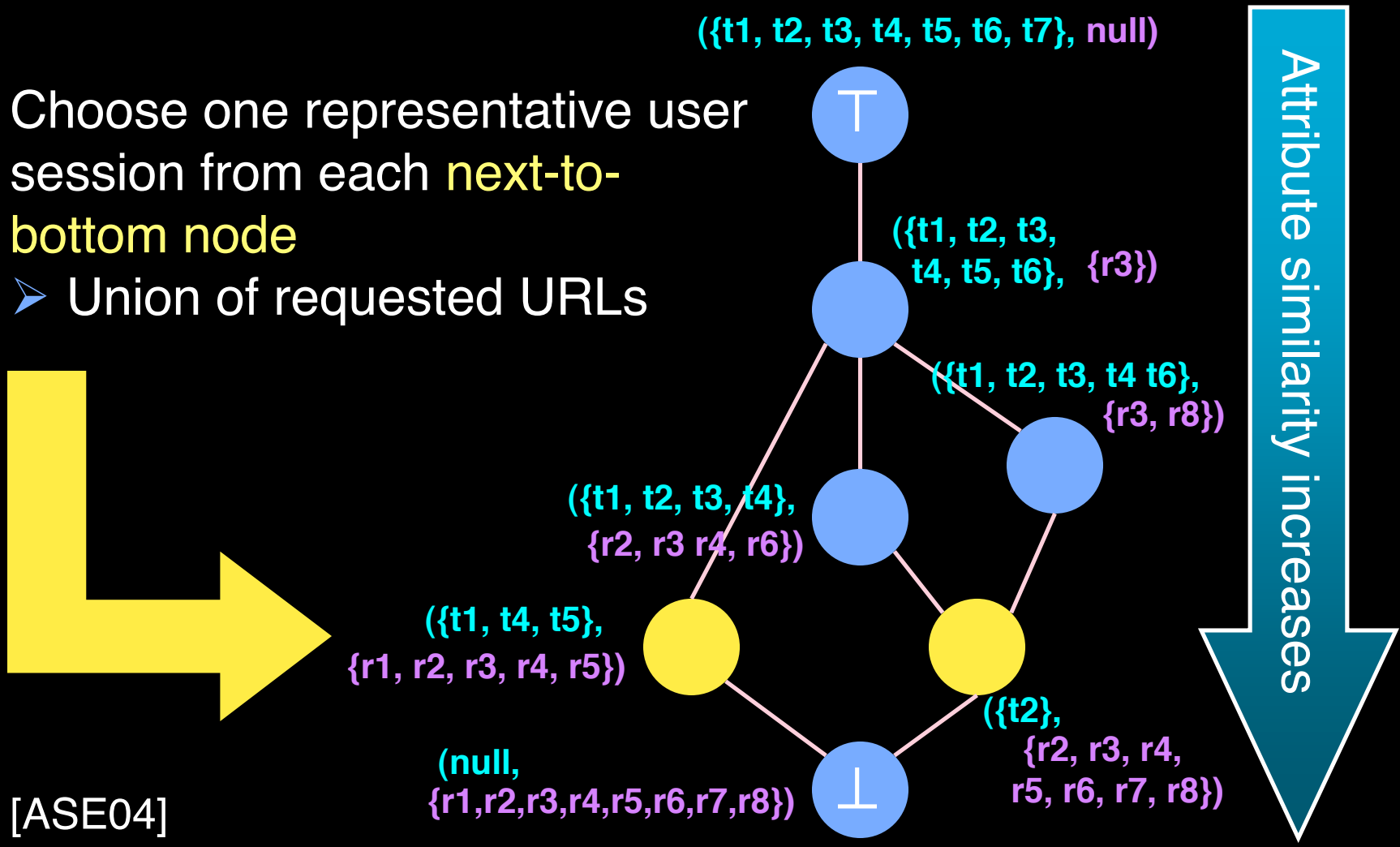


[ASE04]

Test Suite Reduction Heuristic

Choose one representative user session from each next-to-bottom node

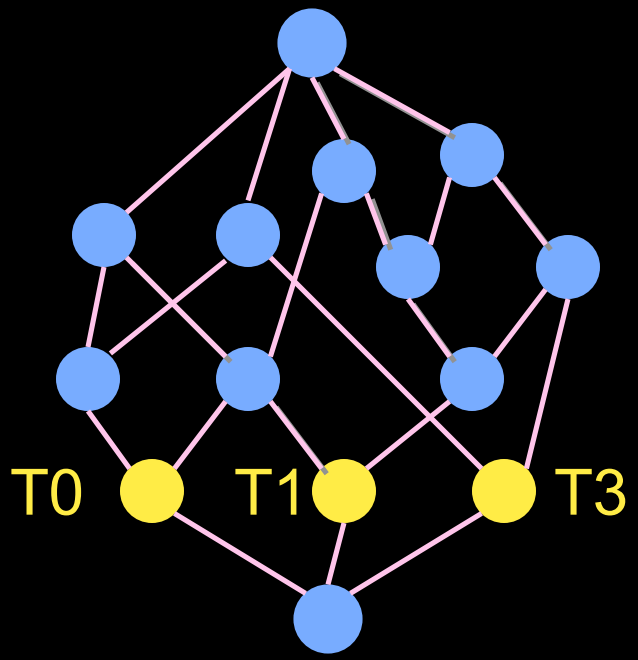
- Union of requested URLs



[ASE04]

Concept-based Reduction: Example

- Choose from next-to-bottom nodes
  - Side-effect: covers URLs



Only 1 session in each next-to-bottom node

		Test Cases					
		T0	T1	T2	T3	T4	T5
Requirements	R0	•	•	•			•
	R1	•	•			•	
	R2				•		
	R3	•				•	
	R4	•			•	•	
	R5		•	•	•		•
	R6		•	•	•		

Empirical Study: Questions

- How well does each technique reduce the test suite?
- How effective is the program coverage of each reduced suite?
- How effective is the fault detection of each reduced suite?
- What are the time and space costs of each technique?

Empirical Study: Expected Results

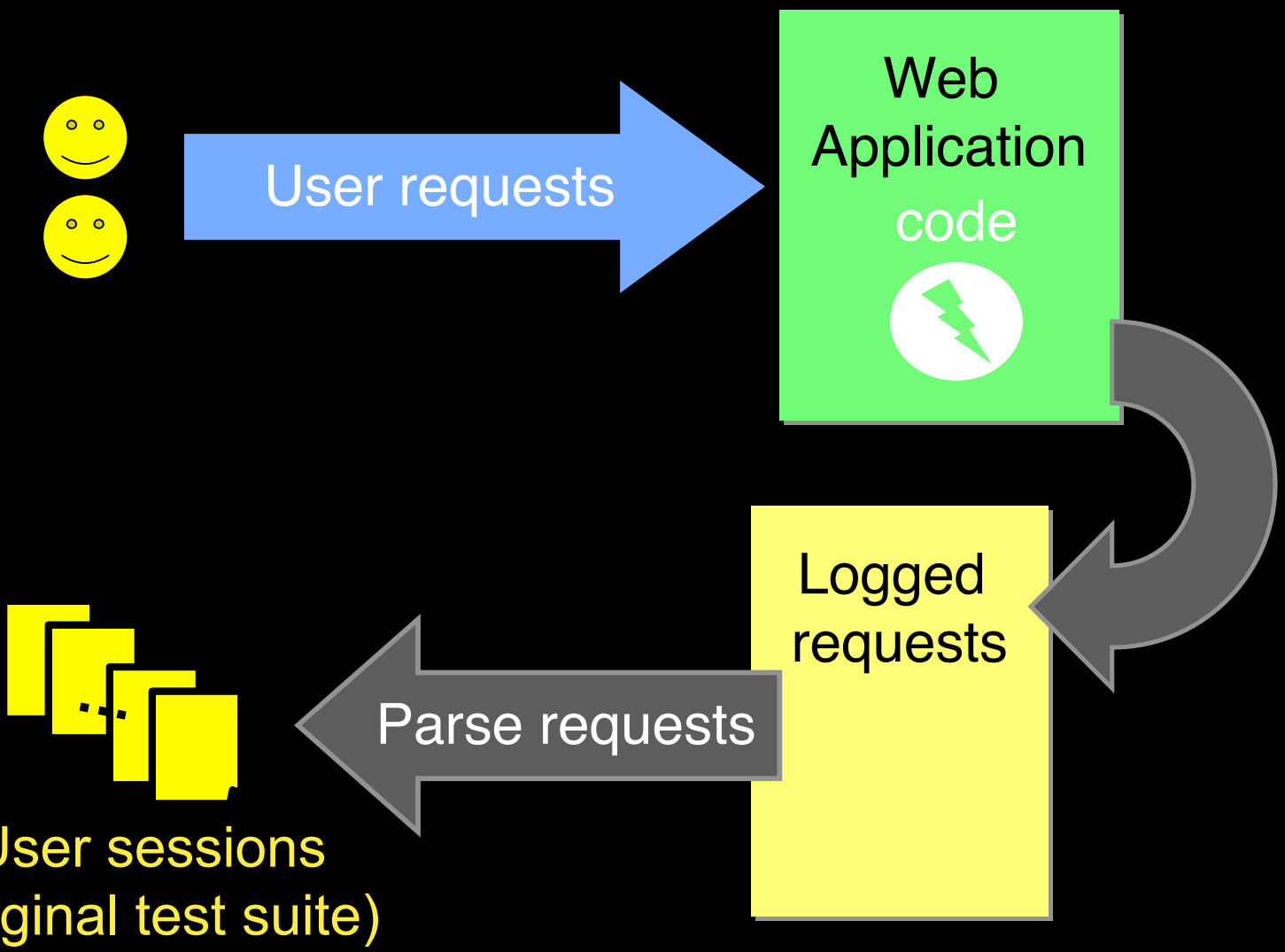
- How well does each technique reduce the test suite?
  - HGS approximates minimal covering set
  - Reduced Suite Size:

    Random > Greedy, Concept > HGS
- How effective is the program coverage of each reduced suite?
  - Program-based *must* match original suite's
  - Concept represents use cases not included by simple URL coverage
  - Program-requirements-based ≥ Concept
  - Concept > URL-requirements-based

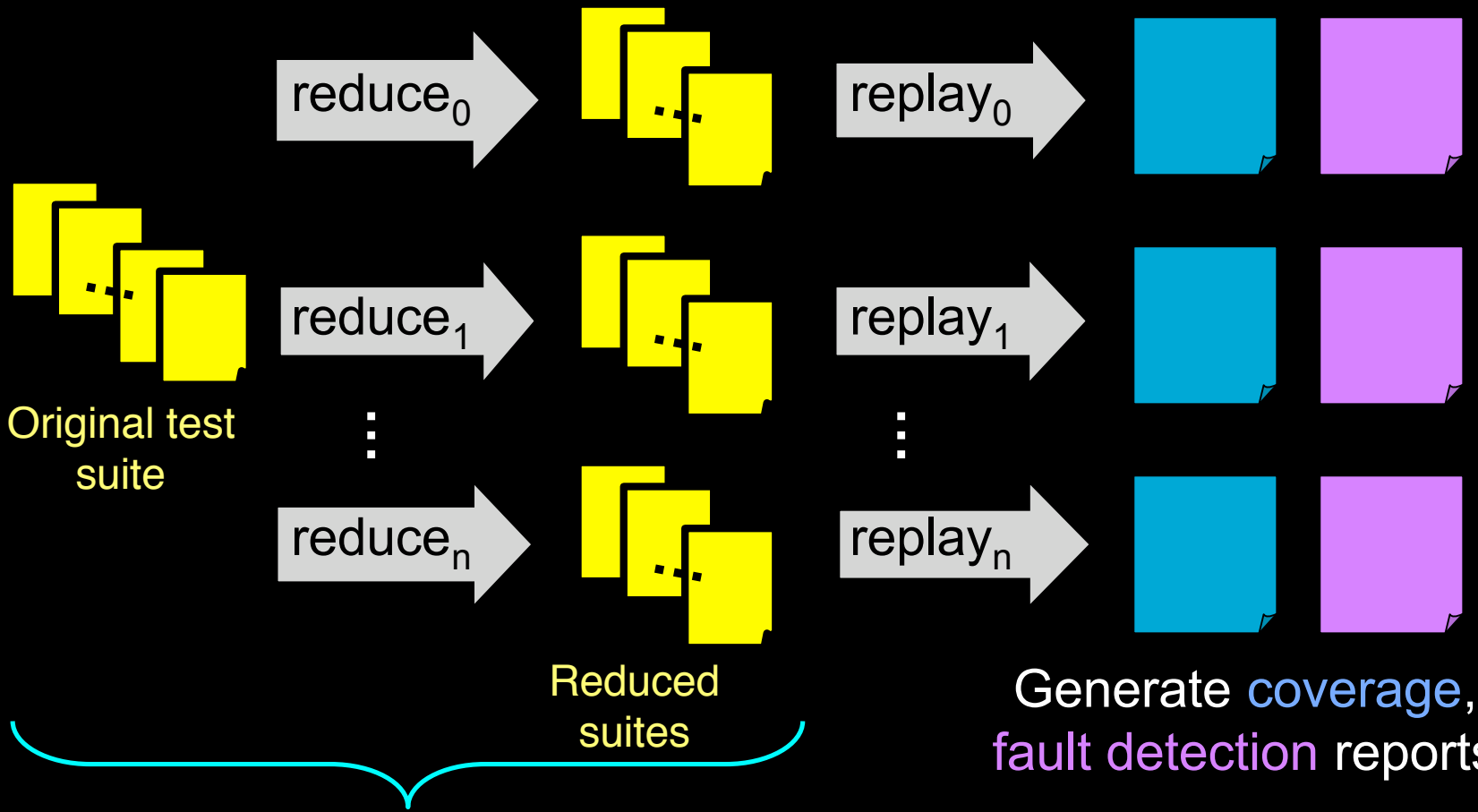
Empirical Study: Expected Results

- How effective is the fault detection of each reduced suite?
  - Concept represents use cases not captured by URL coverage
  - Program-requirements-based ≥ Concept
  - Concept > URL-requirements-based
- What are the time and space costs of each technique?
  - Requirements-based: creating req. mappings
  - Concept: creating lattice
  - Requirements-based > Concept

Empirical Study: Methodology



Empirical Study: Methodology



Measure time, space to generate reduced suites

- includes generating mappings, concept lattice

Compared techniques

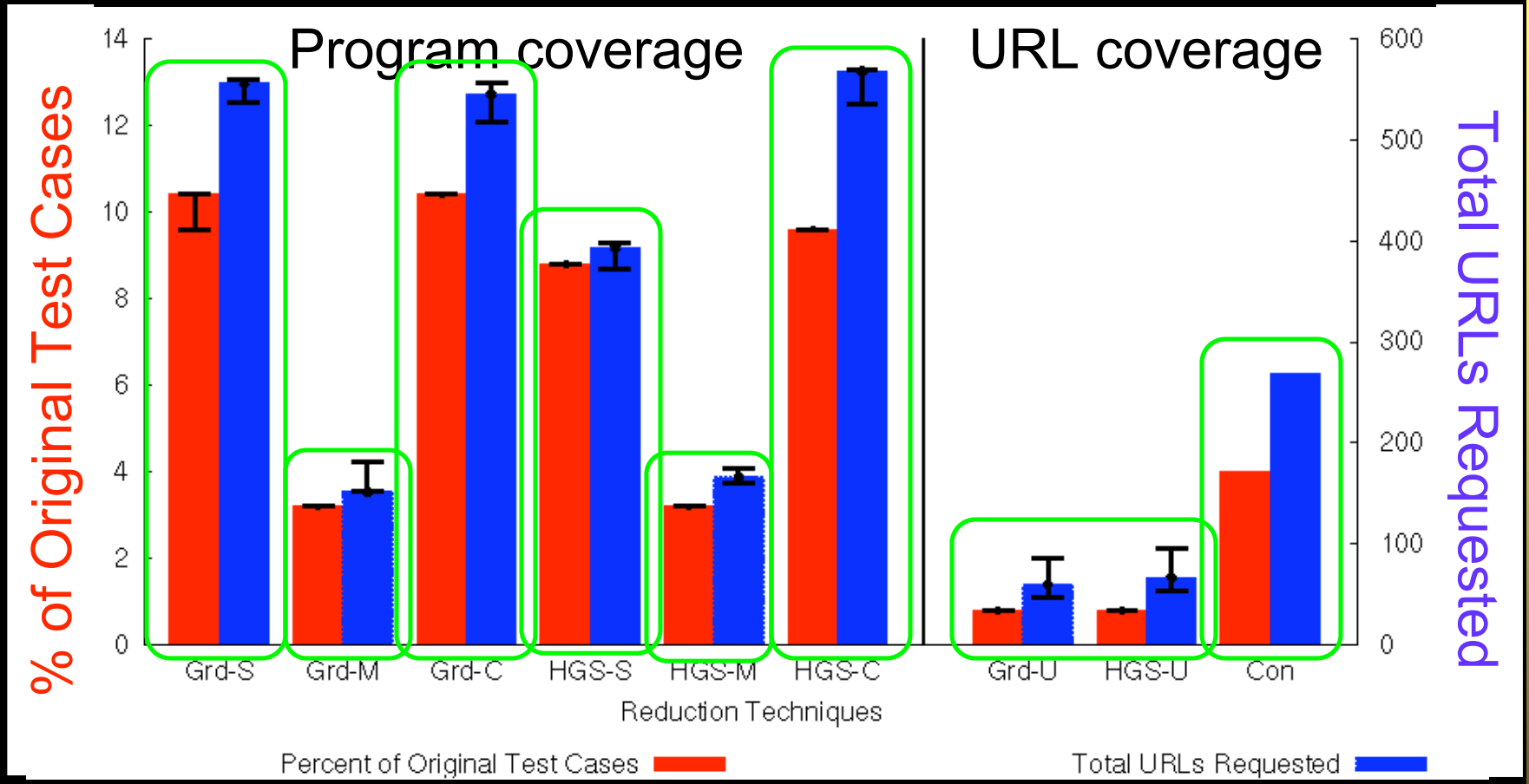
- Requirements-based
  - Selection: Random, Greedy, HGS
  - Requirements: all-statements, all-methods, all-conditionals, all-URLs
  - HGS, Greedy, Random are nondeterministic
    - Run 100 times for each requirement
    - Generated 1200 reduced suites for each app
- Concept-based
  - Selection: from next-to-bottom nodes
  - If >1 session in node,
    - choose randomly, most URLs, least URLs

Subject Applications

- Java-based Web applications

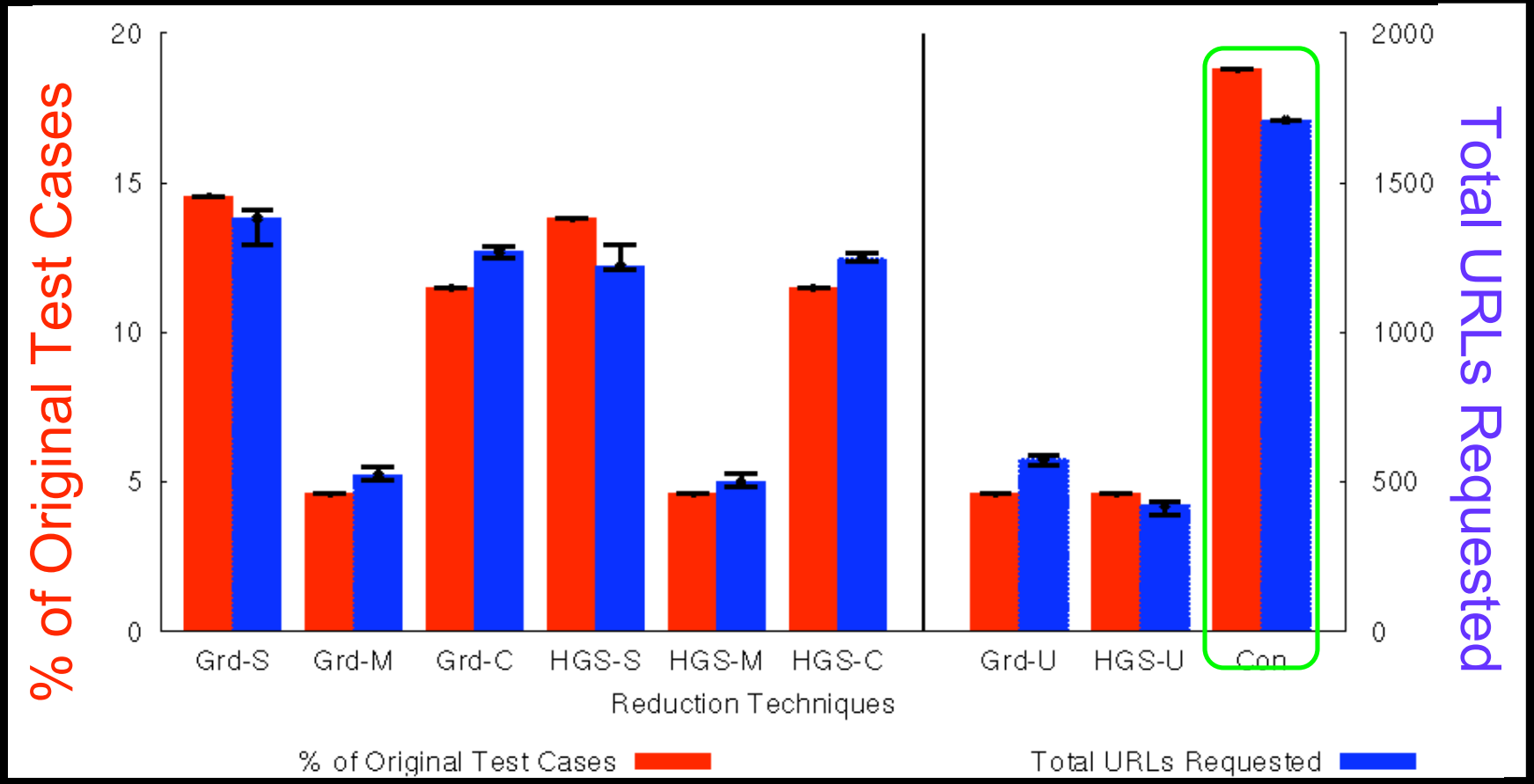
Metrics	Bookstore	CPM
Classes	11	75
Methods	385	172
Conditionals	1808	1274
NC LOC	7791	9300
Seeded Faults	40	86
# User Sessions	125	261
Total URLs Req	3640	3881

Bookstore: Size of Reduced Suites



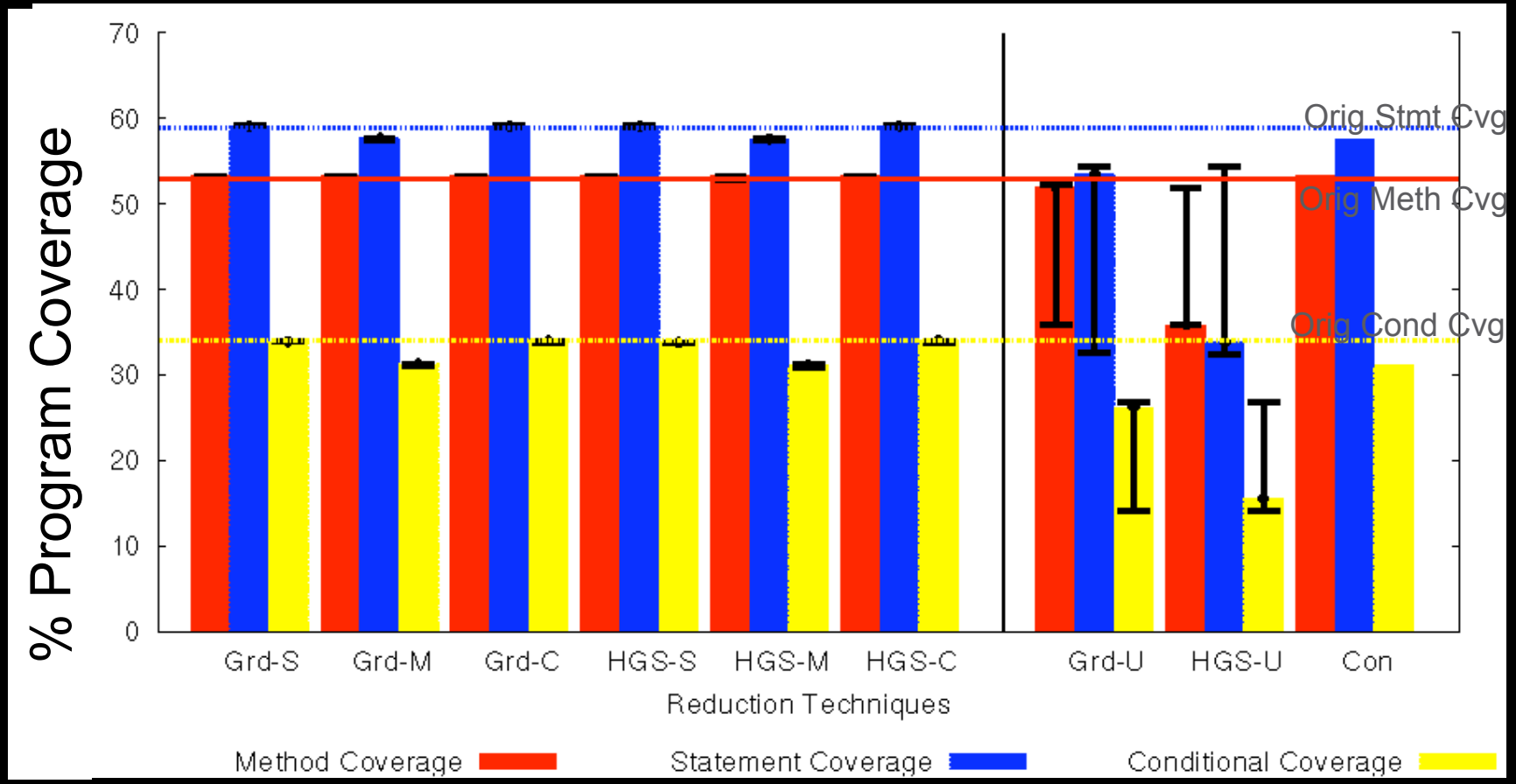
- Greedy/HGS Stmt, Cond > Concept
- Concept > Greedy/HGS Method, URL

CPM: Size of Reduced Suites



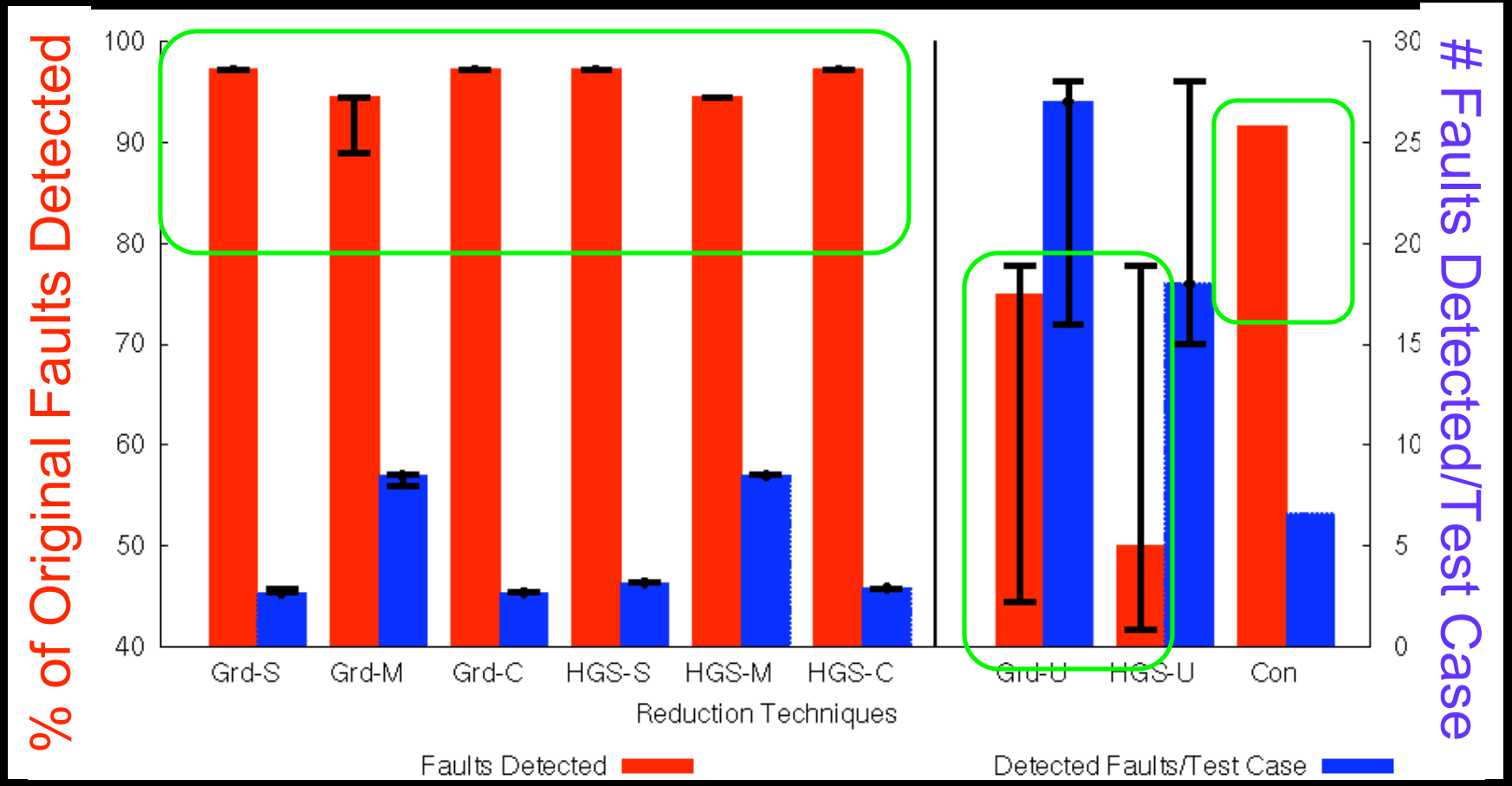
- Concept > all others
- Greedy/HGS: Stmt, Cond > URL, Method

Bookstore: Program Coverage



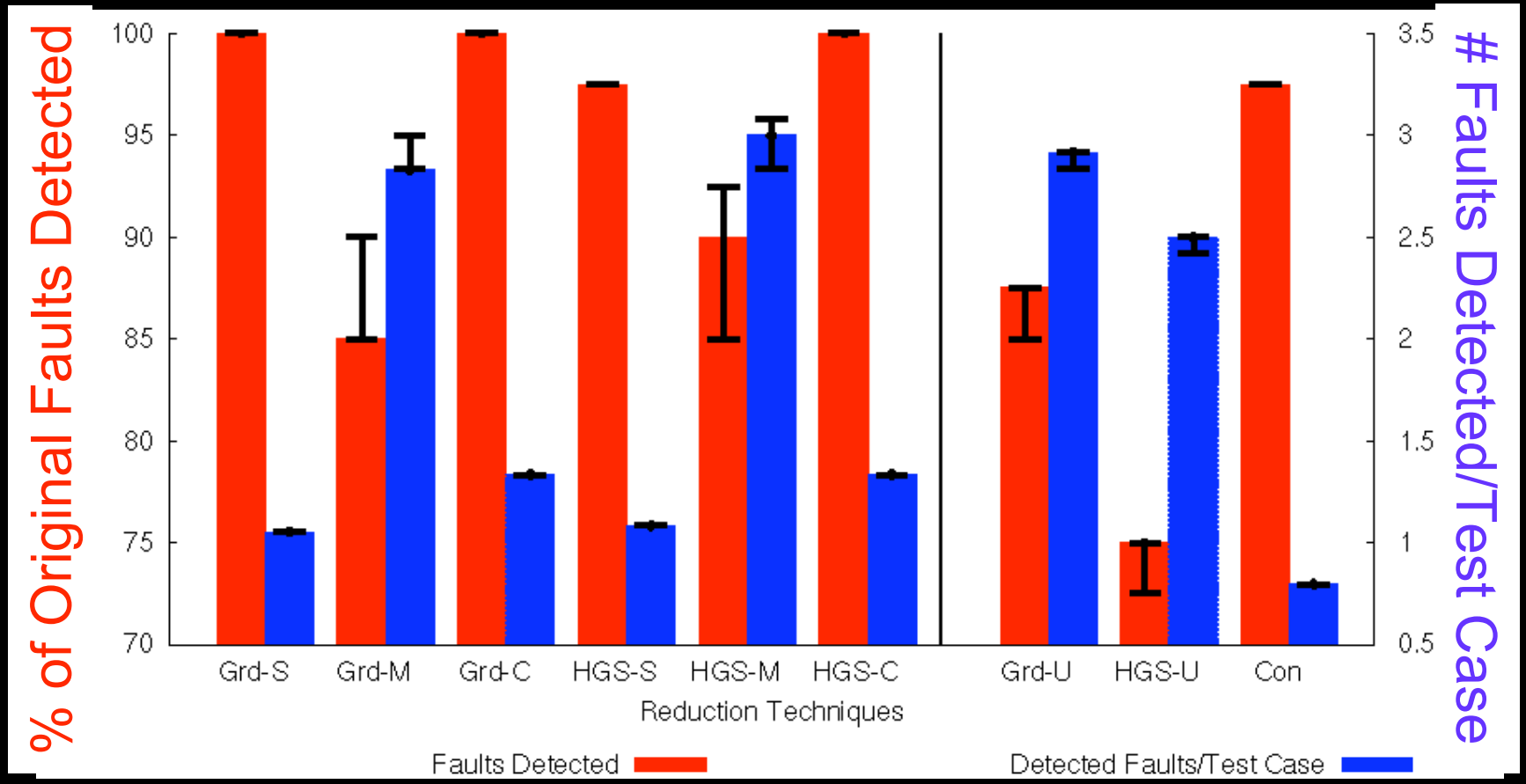
- Concept nearly matches original suite's cvg
- Simple URL coverage: not sufficient

Bookstore: Detected Faults



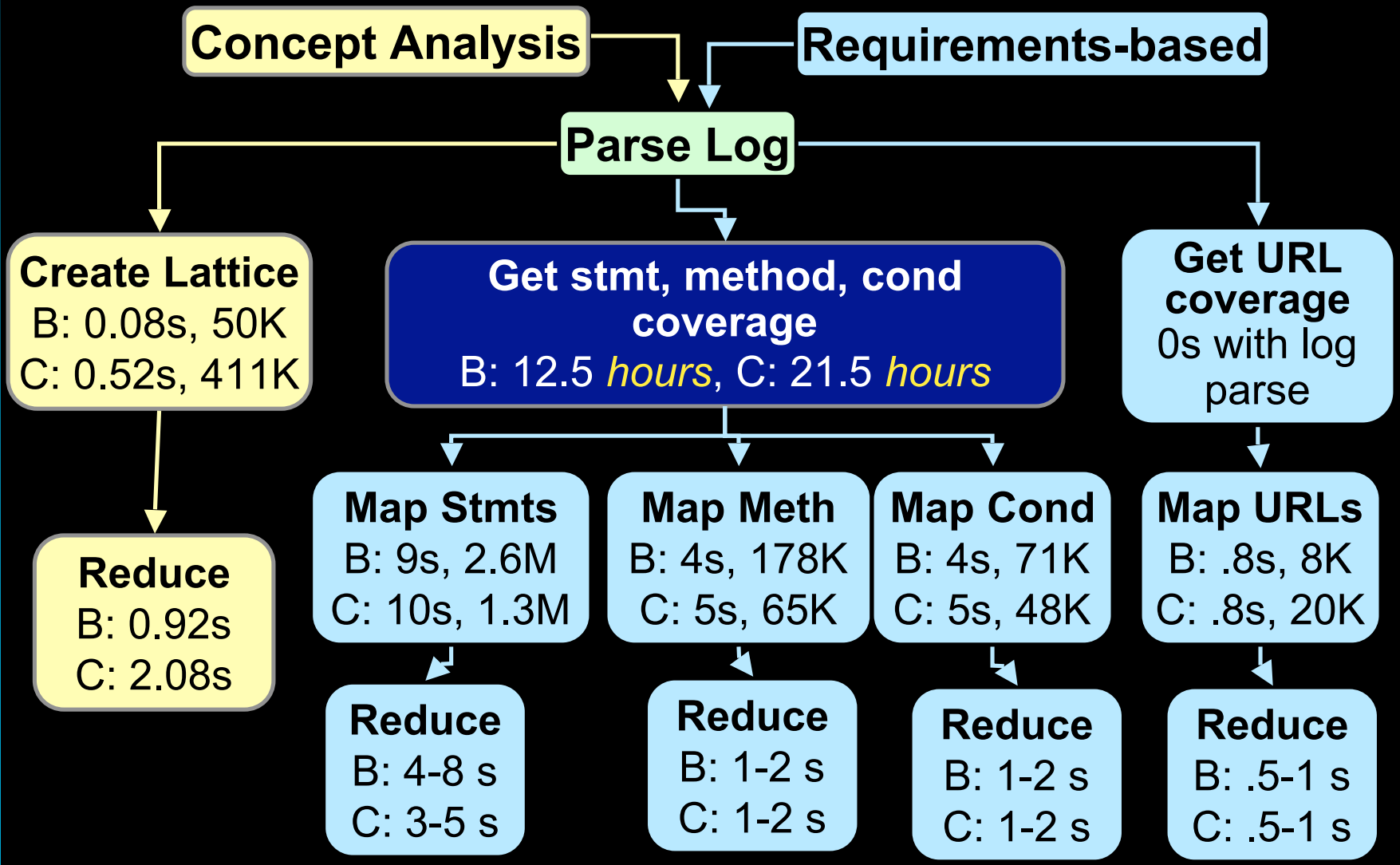
- Greedy/HGS Prog > Concept
- Concept > Greedy/HGS URL

CPM: Detected Faults



- Better fault detection with larger suites
- Greedy/HGS stmt,cond > Concept > Greedy/HGS url, meth

Time and Space Costs



Analysis Summary

- Coverage, Fault Detection
  - ✓Concept ≈ Program-reqs-based
- Size
  - Concept <> HGS
    - Depends on req, original suite
    - Nondeterministic algorithms
  - ✓Random > Greedy ≥ HGS
- Time Cost
  - ✓Program-reqs-based >> Concept

Study Discussion

- Generalize findings as original test suite increases
  - Requirements-mapping cost increases
  - More variation in reduced suite for reqs-based techniques
- Recommended Reduction: Concept
  - Much cheaper reduction cost
  - Consistent, comparable coverage, fault detection

Future Work

- Extend experimental study
  - Different size of original test suites
  - Larger original test suites
  - More applications
  - Additional variations of Concept-based
- Characterize undetected faults
  - Guide improvements to Concept-based

http://www.cis.udel.edu/~hiper