A Scalable Approach to User-session-based Testing of Web Applications through Concept Analysis

Sreedevi Sampath, University of Delaware
Valentin Mihaylov, Drexel University
Amie Souter, Drexel University
Lori Pollock, University of Delaware

Automated Software Engineering Conference (ASE), 2004
September 22, 2004

Web-based Applications

- Web pages on server, clients access pages
  - Example: e-commerce applications
- Characteristics
  - Short time to market
  - Extensive use
  - High reliability, continuous availability
  - Changing user profiles
- User-session data
  - Client URL requests and name-value pairs (data)

Challenges of User-session-based Testing

- Leverage user-session data as test cases applicable in beta/maintenance testing phases
  - Elbaum et al. (2003)
- Manage/replay large set of user sessions
- Existing test suite reduction non-scalable
  - Harrold et al. (1993)

Contributions

- View user sessions as use-cases
- Apply concept analysis for test suite reduction
- Perform incremental test suite update
- Automate the testing framework [ICSM 04]
- Evaluate cost effectiveness
  - Test suite size
  - Program coverage
  - Fault detection

⇒ Scalable user-session-based testing

Concept Analysis

Mathematical technique for clustering objects that have common discrete attributes

Input – theory

Set of objects, O

Set of attributes, A

Binary relation, $R \subseteq O \times A$

Example Concept Analysis – Input

Objects: **{airplane, boat, rollerskates, unicycle}**

Attributes: **{wheel(s), over80mph, passengers, wear, engine}**

Binary Relation Table:

attributes

objects

	wheel(s)	over80mph	passengers	wear	engine
airplane	●	●	●		●
boat		●	●		●
rollerskates	●			●	
unicycle	●				

Concept Analysis - Output

- Identifies concepts for a given tuple (O, A, R)
- A *concept* is a tuple

$$t = (O_i, A_j)$$

- Concepts form a partial order defined as

$$(O_1, A_1) \leq (O_2, A_2) \; iff \; O_1 \subseteq O_2$$

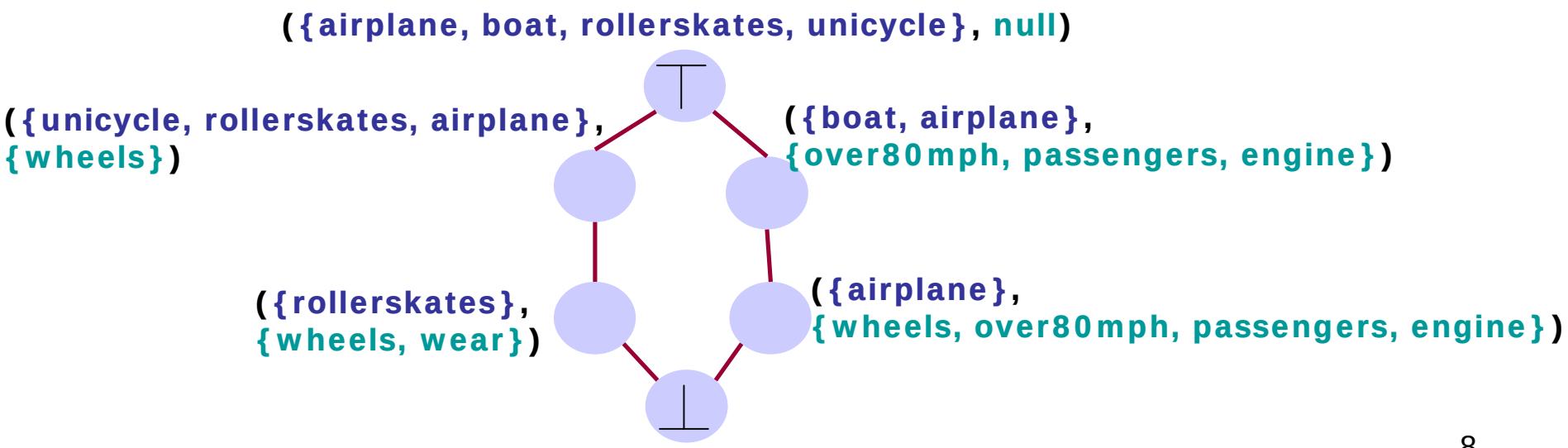
$$(O_1, A_1) \leq (O_2, A_2) \; iff \; A_1 \supseteq A_2$$

Example Concept Analysis Output

attributes

objects

	wheels	over80mph	passengers	wear	engine
airplane	●	●	●		●
boat		●	●		●
rollerskates	●			●	
unicycle	●				



Concept Analysis for Web Testing

attributes (URLs)

objects

	GD	GR	GL	PL	GS	GB	GM
us1	●	●	●				
us2	●	●	●		●		●
us3	●	●	●	●	●		
us4	●	●	●		●	●	
us5	●	●	●				
us6	●	●	●	●	●	●	

- Set of objects
  - User sessions, *s*, denoted by IP addresses
- Set of attributes
  - URLs, *u*
- Binary relation table
  - A pair *(s, u)* is in the relation table if *s* requests *u*

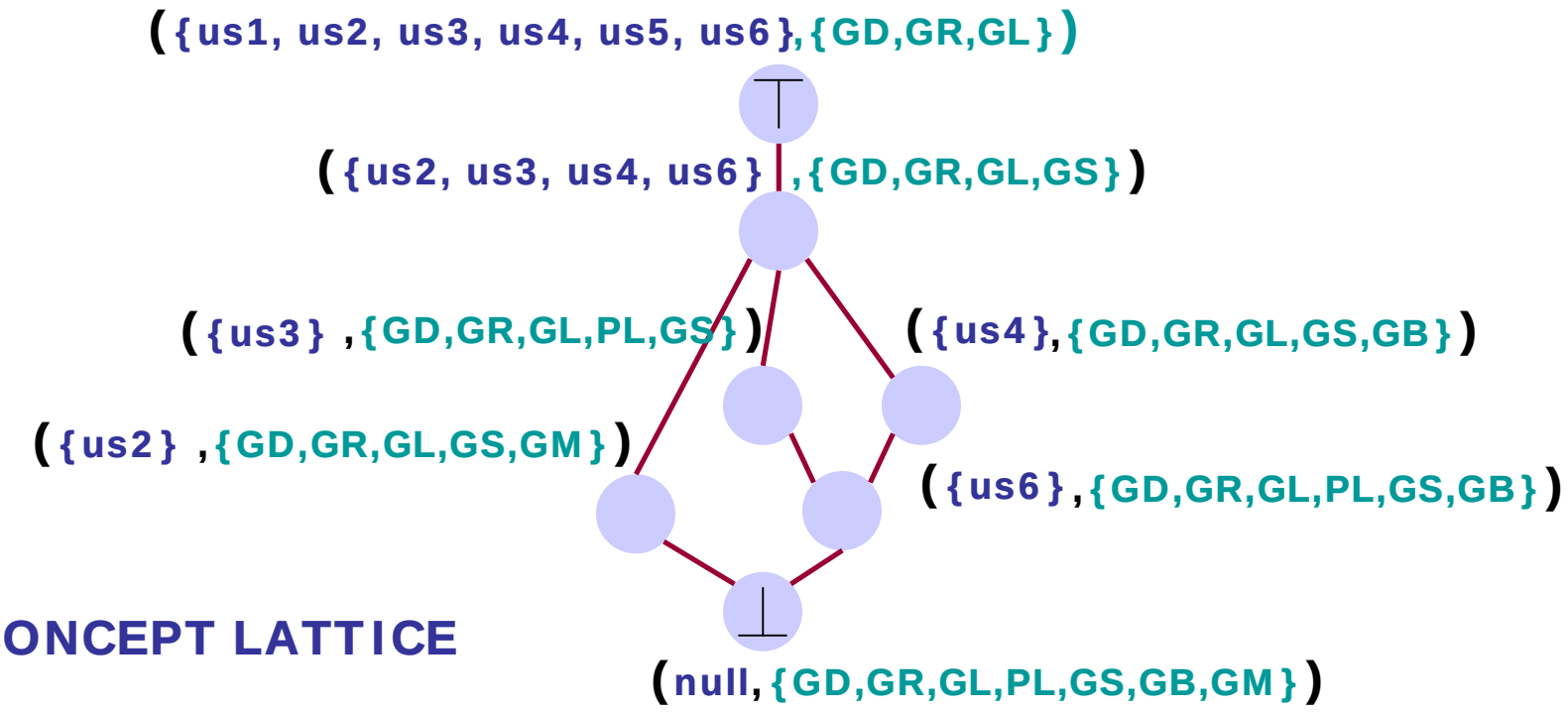
RELATION TABLE

attributes (URLs)

objects

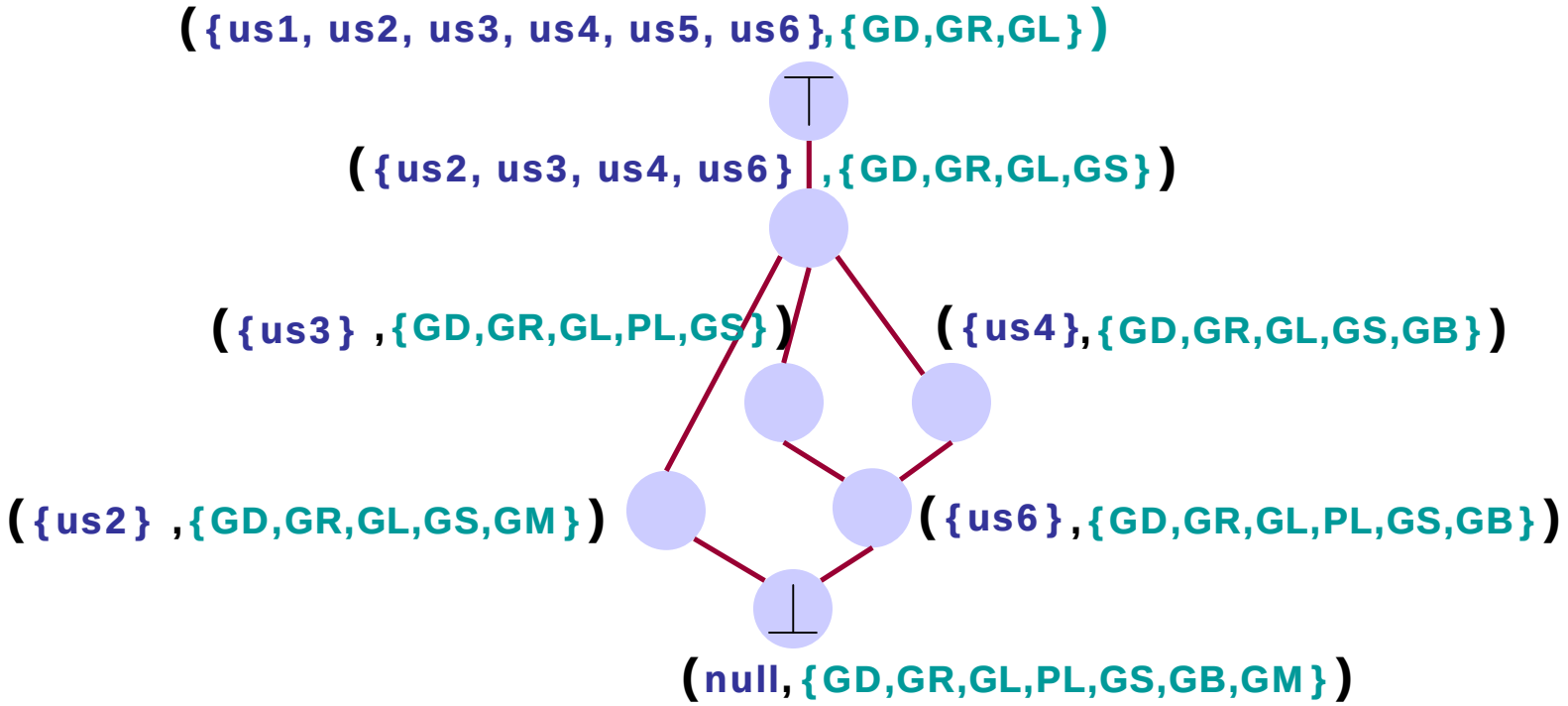
	GD	GR	GL	PL	GS	GB	GM
us1	●	●	●				
us2	●	●	●		●		●
us3	●	●	●	●	●		
us4	●	●	●		●	●	
us5	●	●	●				
us6	●	●	●	●	●	●	

CONCEPT LATTICE

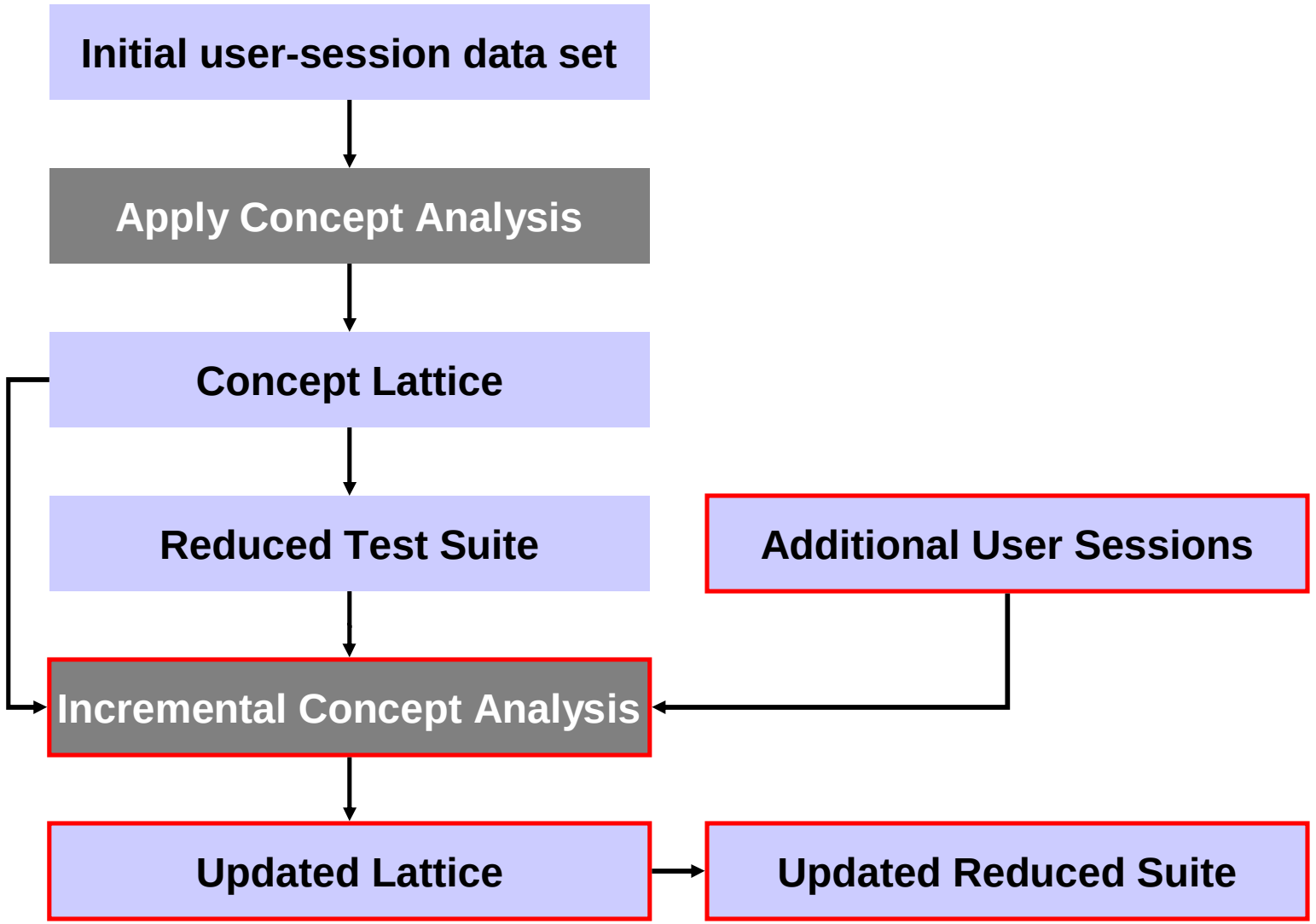


Test Suite Reduction

- Exploit lattice's hierarchical use-case clustering [WODA 04]
- Heuristic for test suite reduction
  - Smallest set of user sessions that will cover all URLs of the application executed by original suite



Incremental Test Suite Update



Incremental Test Suite Update

- Utilize Godin et al.'s incremental algorithm
  - Create new nodes/edges
  - Modify existing nodes/edges
- **Key insight: Scalability**

  Existing internal nodes do not sink to bottom
  - Test cases not maintained for internal nodes

Incremental Test Suite Update

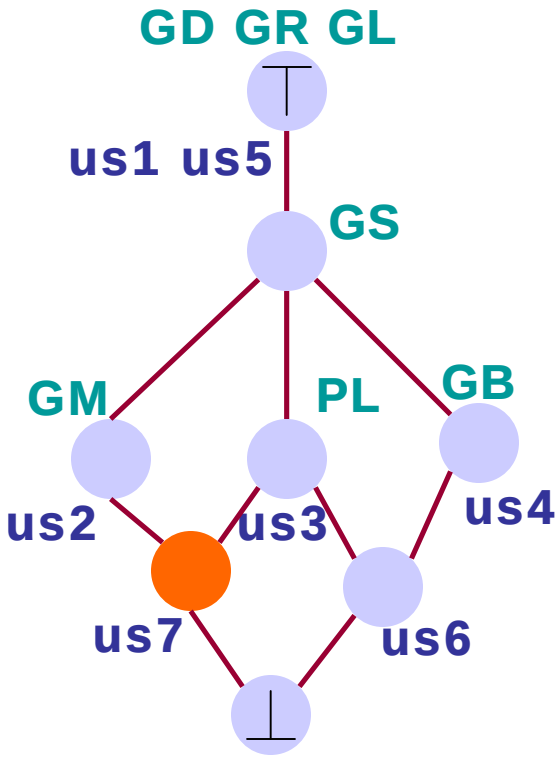
RELATION TABLE

attributes (URLs)

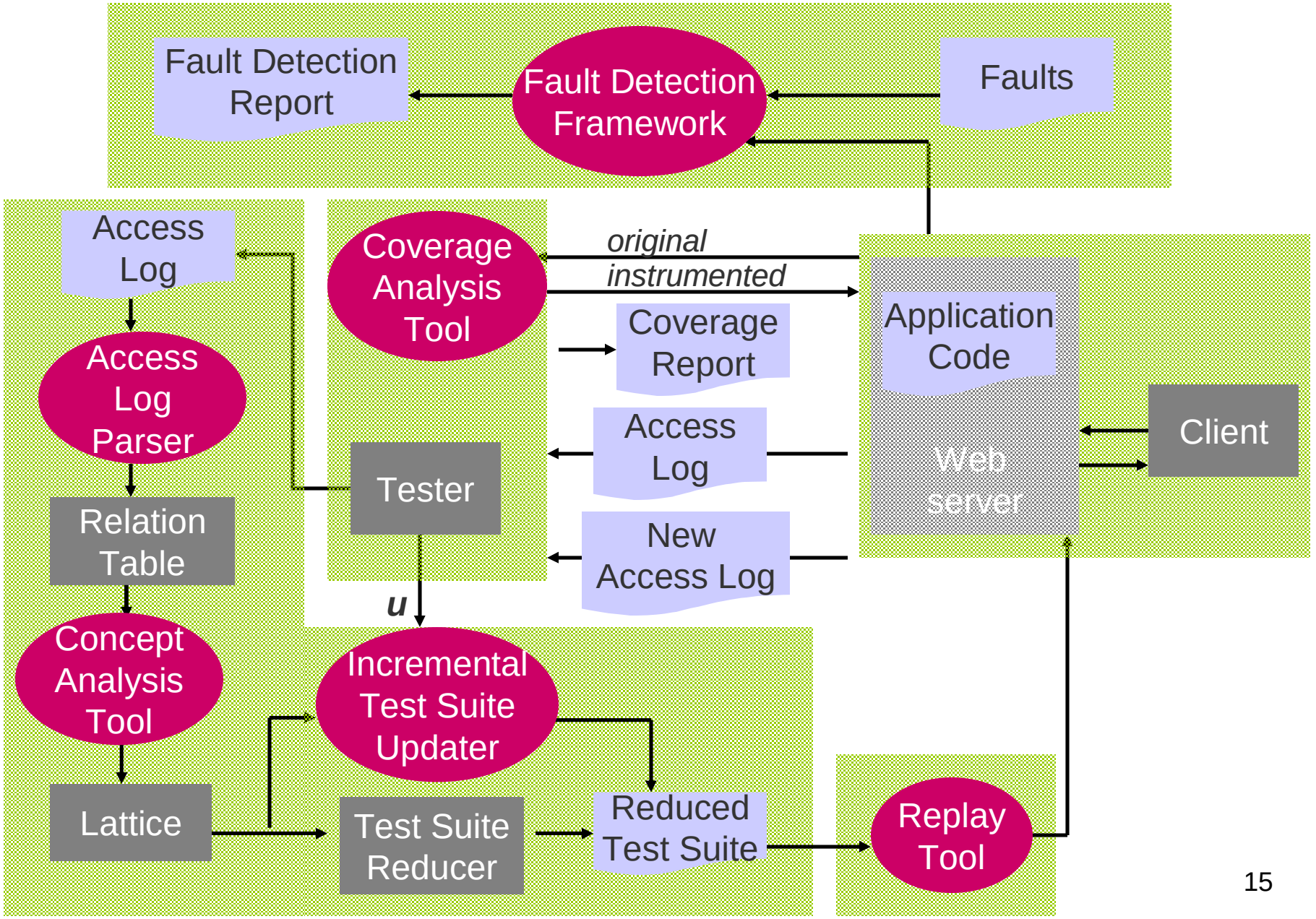
objects

	GD	GR	GL	PL	GS	GB	GM
us1	●	●	●				
us2	●	●	●		●		●
us3	●	●	●	●	●		
us4	●	●	●		●	●	
us5	●	●	●				
us6	●	●	●	●	●	●	
us7	●	●	●	●	●		●

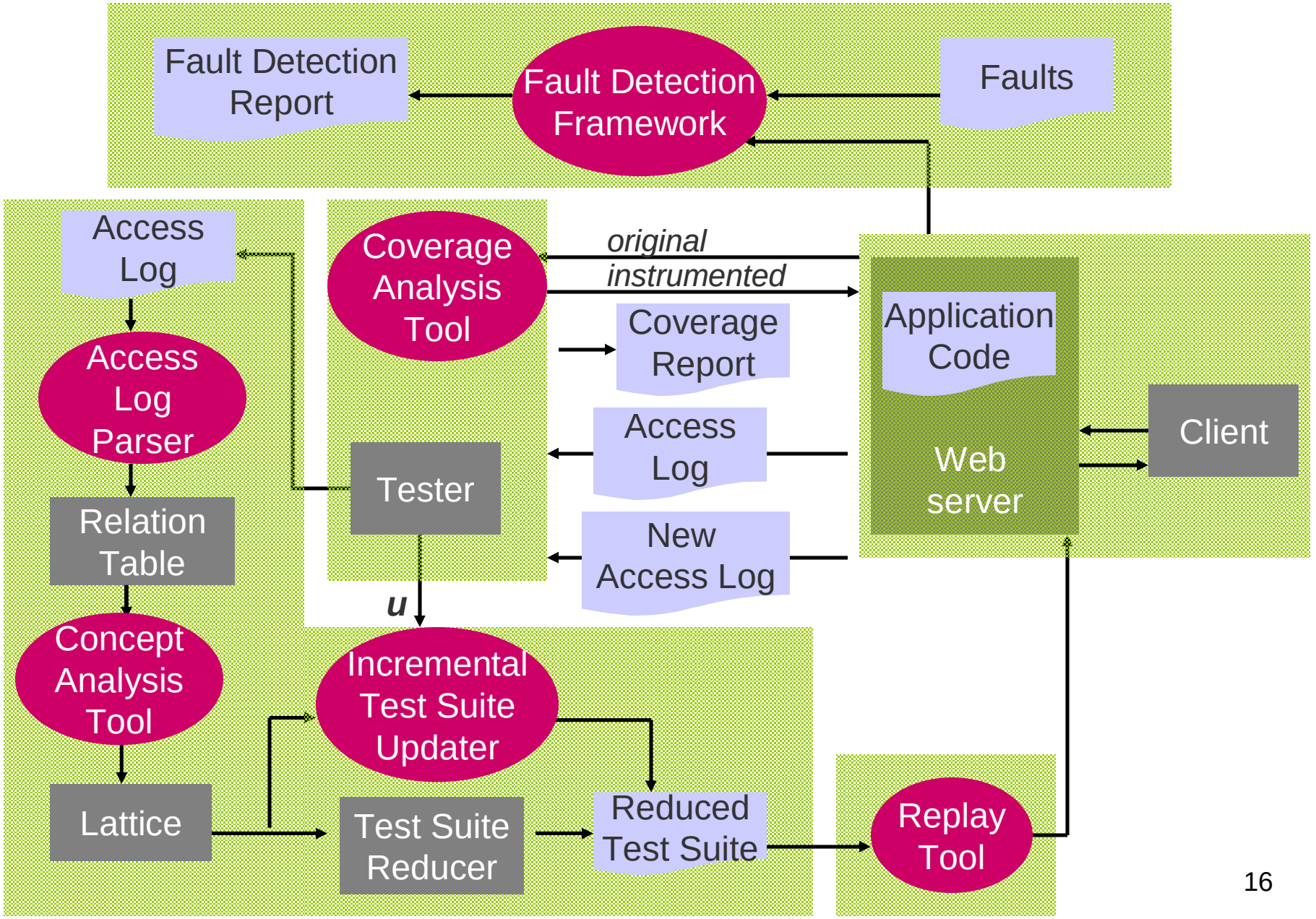
SPARSE UPDATED LATTICE



Web Testing Framework



Web Testing Framework



Experimental Evaluation

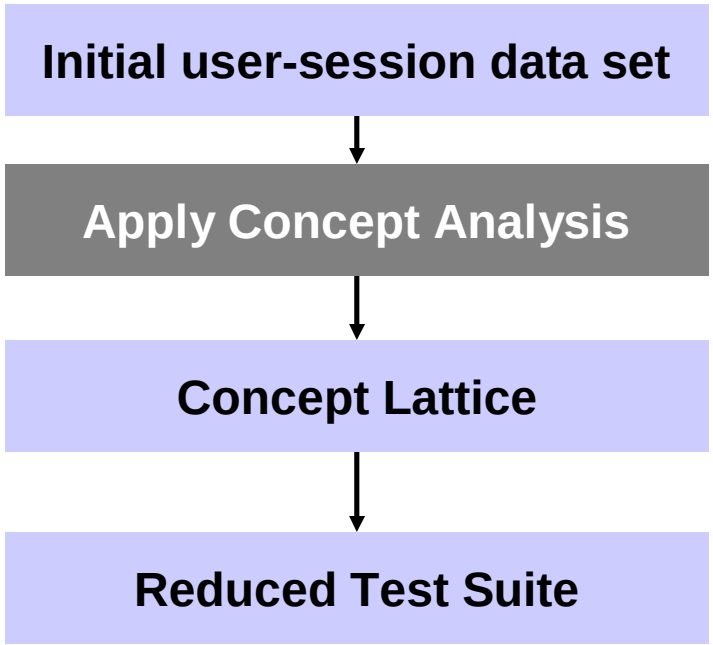
- Evaluate test suite reduction achieved through concept analysis
  - Test suite size
  - Replay and oracle time
- Cost-effectiveness of incremental vs. batch
- Program coverage analysis of reduced suite
- Fault detection capability of reduced suite

Study Setup

- Bookstore application
  - 9748 lines of code
  - 385 methods
  - 11 classes
- JSP front-end, MySQL backend
- Resin web server
- 123 user sessions
- 40 faults

Test Suite Reduction

Methodology



Metrics

- Test suite size
- Replay time
- Oracle time

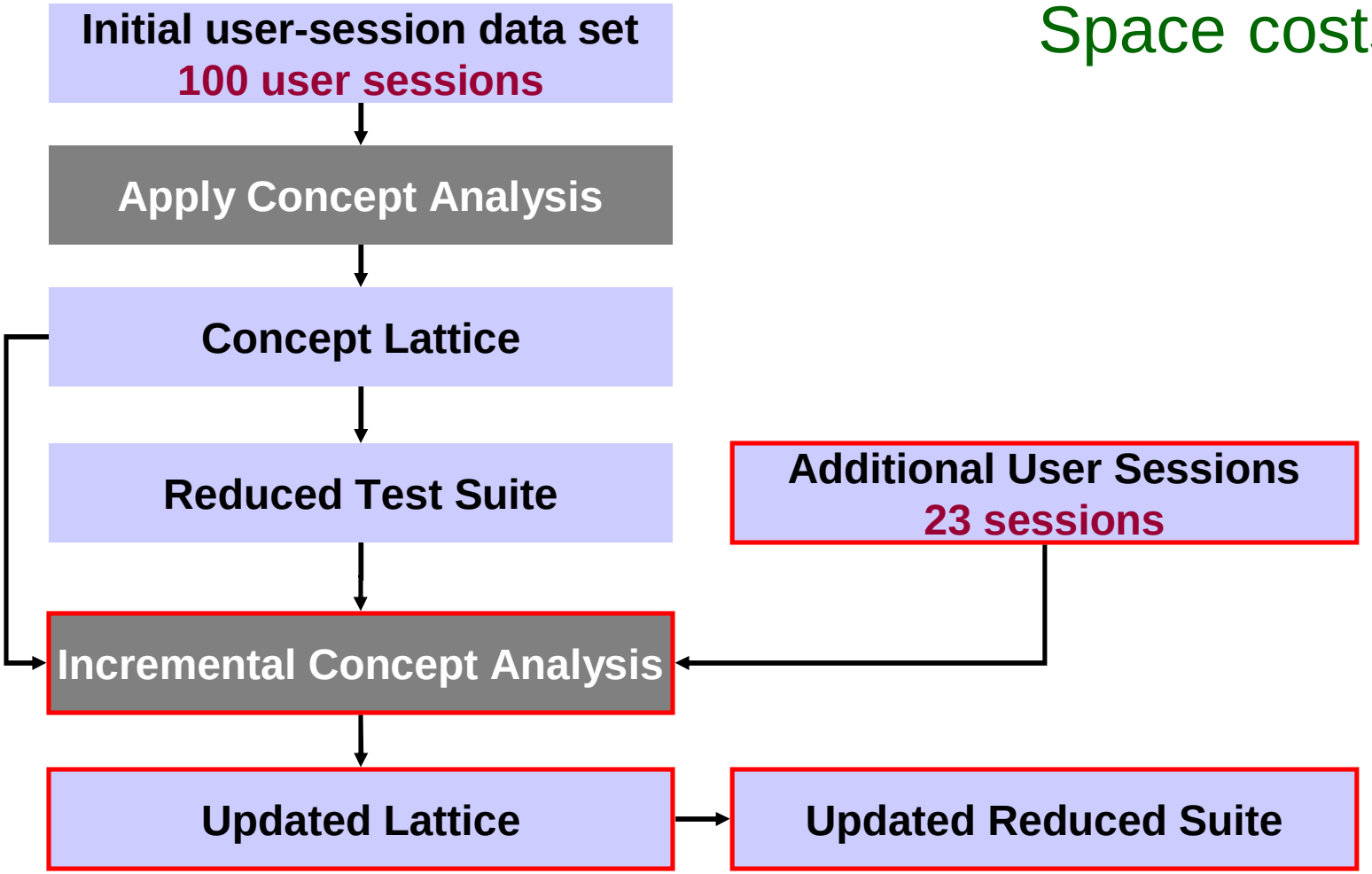
Test Suite Reduction Results

Metrics	Original Suite	Reduced Suite	Reduction
Test suite size	123	15	87.8%
Replay time	16m56s	4m22s	74.2%
Oracle time	25m30s	5m17s	79.3%

- The large percent reduction in test suite size results in reduction in replay and oracle time
- Considerable savings in replay and oracle when compared to time for suite reduction, 19s [ICSM 04]

Incremental versus Batch

Methodology



Metric

Space costs

Incremental versus Batch Results

Metric	Original Suite	Reduced Suite	Reduction
Space	4.3MB	1MB	76.7%

- **Scalability:** Incremental test suite update saves space by not maintaining original suite of sessions
- Perform incremental update overnight with day's collection of user sessions to produce updated suite

Program Coverage

Methodology

- Use *Clover* for coverage analysis
- Restore database state before replay
- Use *wget* for replaying user sessions
- Pass cookies and post-data information

Metrics

- Statement and method coverage

Program Coverage Results

Metrics	Original Suite	Reduced Suite	Preserved Coverage
Statement Coverage	60.3%	58%	96.2%
Method Coverage	53.2%	53.2%	100%

- Reduced suite preserves program coverage obtained from original suite
- Low loss of program coverage due to
  - Heuristic that covers all URLs of the application
  - Reduction with low loss of use case representation [WODA 04]

Fault Detection Study

Methodology

- Manually seed 40 faults in application
- Replay user sessions through
  - Correct version of application to generate expected result
  - Faulty versions of application to generate actual result
- *Diff* the expected and actual results

Metric

- Number of faults detected

Fault Detection Study Results

Metrics	Original Suite	Reduced Suite	Preserved
Faults Detected	87.5%	70%	80%

- Reduced suite maintains fault detection capability
- Investigating additional test suite reduction heuristics

Related Work

- Concept Analysis in Software Engineering
  - Recovering components, Eisenbarth et al. (2003)
  - Debugging temporal specifications, Ammons et al. (2003)
- Reducing Test Suites
  - Harrold et al. (1993)
  - Offutt et al. (1995)
- Web Testing
  - Link and form testers
  - Liu et al. (2000)
  - Ricca and Tonella (2001)
  - Di Lucca et al. (2002)
  - Elbaum et al. (2003)

Conclusions

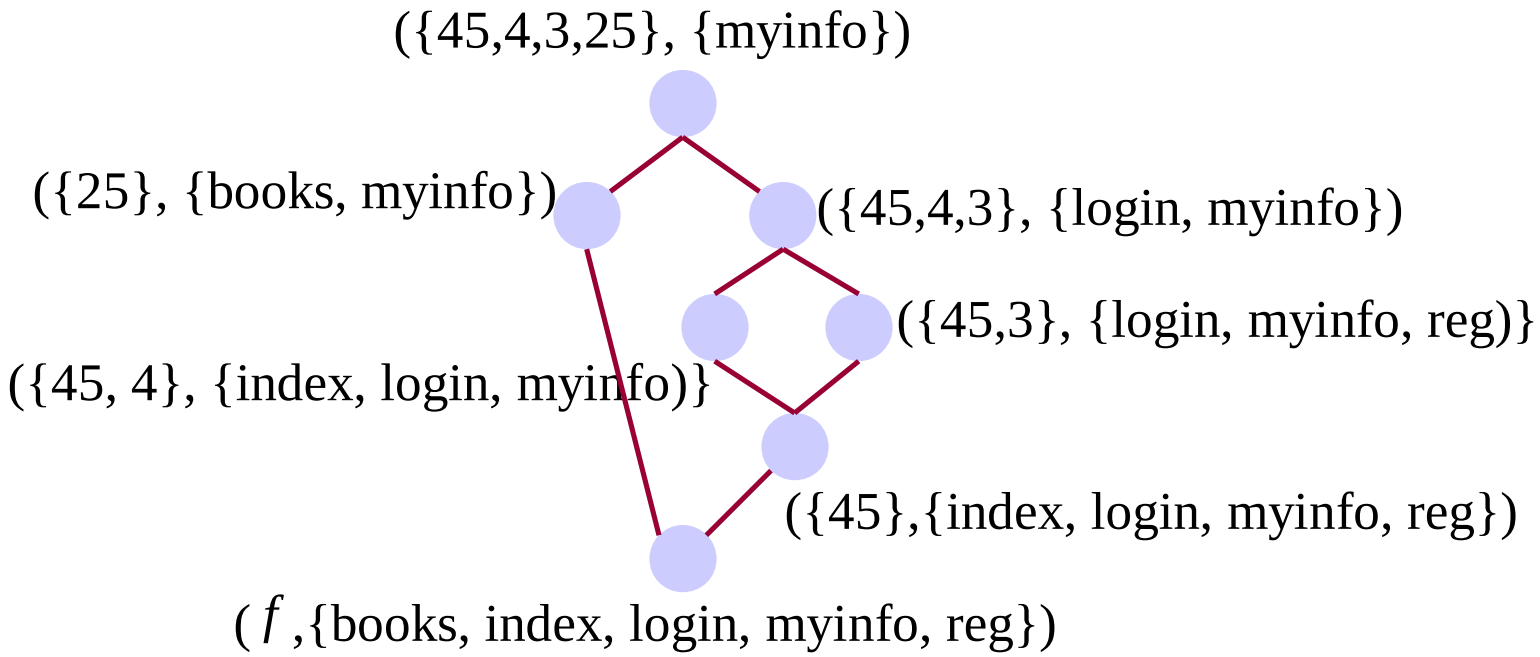
- Test suite reduction by concept analysis
  - Achieves large reduction in test suite size
  - Saves oracle and replay time
- Incremental test suite update presents scalable approach to user-session-based testing
- Reduced test suite preserves program coverage and fault detection capability

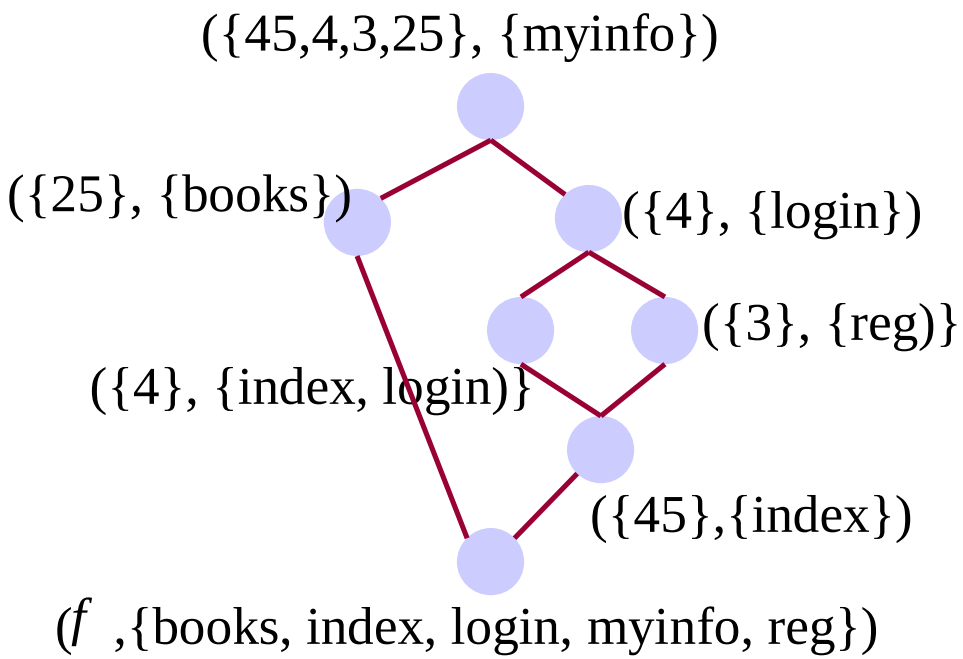
Future Work

- More significant empirical evaluation
- Extend incremental concept analysis to handle program evolution
- Alternate heuristics for test suite reduction

Relation Table and Concept Lattice

	index	login	reg	books	myinfo
45	●	●	●		●
3		●	●		●
4	●	●			●
25				●	●




({45,4,3,25}, {myinfo})
({25}, {books})
({4}, {login})
({3}, {reg)}
({4}, {index, login)}
({45},{index})
(f ,{books, index, login, myinfo, reg})