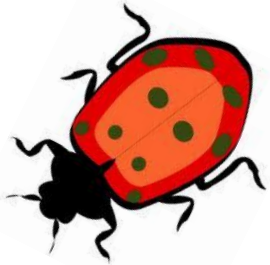


@tComment: Testing Javadoc Comments to Detect Comment-Code Inconsistencies

Shin Hwei Tan
University of Illinois

Darko Marinov
University of Illinois

Lin Tan
University of Waterloo

Gary T. Leavens
University of Central Florida

What are *Javadoc* Comments?

```
/*
* Returns a synchronized map backed by the given map.
…
* @param map the map to synchronize, must not be null
* @return a synchronized map backed by the given map
* @throws IllegalArgumentException if the map is null
*/
static <K,V> Map<K,V> synchronizedMap(Map<K,V> map)
```

Overall Description

Block Tags

- @param - Parameter name, Description
- @return - Description
- @throws - Exception name, Condition under which the exception is thrown

Popularity of *Javadoc* Comments

- Writing ***Javadoc*** comments is a common practice
- Lots of ***Javadoc*** comments exist in Java libraries.

Project	Number of methods	Number of *Javadoc* comments for methods	Ratio (%)
Collections	3,874	2,434	63
GlazedLists	2,753	1,741	63
JFreeChart	6,205	6,186	100
JodaTime	3,887	2,917	75
Log4j	2,115	958	45
Lucene	5,222	2,205	42
Xalan	5,404	3,229	60

Javadoc Comments can be Inconsistent with Code

```
/*…
* @param map the map to synchronize, must not be null
* @return a synchronized map backed by the given map
* @throws IllegalArgumentException if the map is null
*/
static <K,V> Map<K,V> synchronizedMap(Map<K,V> map)
```

Expected behavior for synchronizedMap (null):

- Throws *IllegalArgumentException*

Actual behavior:

- Throws *NullPointerException*

Checking Comments

Important to find bugs in comments

- Comments are not executed
- Comments are read by developers to understand code
- Incorrect comments could cause developers to write wrong code **[SOSP'07]**

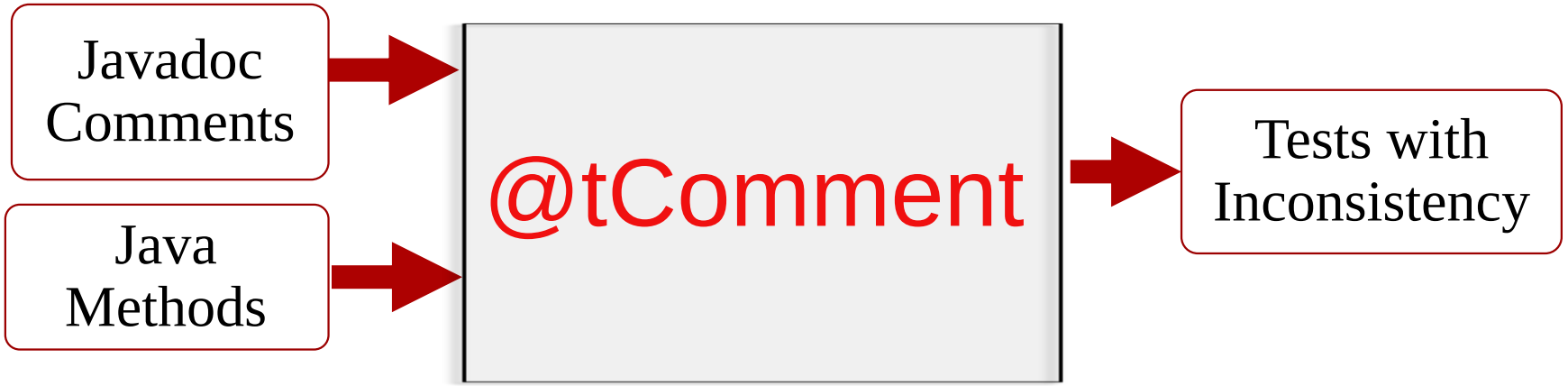
Challenging to automatically analyze comments

- Ambiguities in understanding general text
- NLP made progress but struggles with general text

Prior Work on Checking Code-Comment Consistency

Build domain-specific analyses

- *iComment* (locking protocols & function calls) **[SOSP'07]**
- *aComment* (interrupts) **[ICSE'11]**
  - System code (C/C++)
  - Extract machine-checkable rules from comments
  - Use *static analysis* to check consistency between code and comments/rules



New Domain

- Method properties for null values and related exceptions

Dynamic Analysis (Random Testing)

- Fewer false alarms compared to static analysis

Improved Testing

- Reduce false alarms in test generation tool

Evaluation on 7 Libraries

- Found 28 inconsistencies, 12 were fixed

Example Inconsistency of Type 1: Correct Code, Incorrect Comment

```
/*…
* @param map the map to synchronize, must not be null
* @return a synchronized map backed by the given map
* @throws IllegalArgumentException if the map is null
*/
static <K,V> Map<K,V> synchronizedMap(Map<K,V> map)
```

Example Inconsistency of Type 1: Correct Code, Incorrect Comment

```
/*…
* @param map the map to synchronize, must not be null
* @return a synchronized map backed by the given map
* @throws IllegalArgumentException if the map is null
*/
static <K,V> Map<K,V> synchronizedMap(Map<K,V> map)
```

Example Inconsistency of Type 1: Correct Code, Incorrect Comment

```
/*…
* @param map the map to synchronize, must not be null
* @return a synchronized map backed by the given map
* @throws IllegalArgumentException if the map is null
*/
static <K,V> Map<K,V> synchronizedMap(Map<K,V> map)
```


```
public void test1() throws Throwable {
    java.util.Map var0 = null;
    try {
        java.util.Map var1= ...synchronizedMap(var0);
    } catch (IllegalArgumentException expected) {return;}
    fail("Expected exception of type IllegalArgumentException
                but got NullPointerException“);
}
```

Example Inconsistency of Type 1: Correct Code, Incorrect Comment

```
/*…
* @param map the map to synchronize, must not be null
* @return a synchronized map backed by the given map
* @throws IllegalArgumentException if the map is null
*/
static <K,V> Map<K,V> synchronizedMap(Map<K,V> map)
```

- ✓ Confirmed & fixed by Collections developers

@tComment

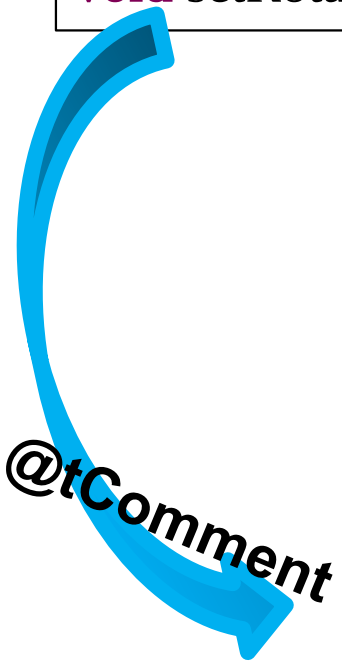
```
public void test1() throws Throwable {
    java.util.Map var0 = null;
    try {
        java.util.Map var1= ...synchronizedMap(var0);
    } catch (IllegalArgumentException expected) {return;}
    fail("Expected exception of type IllegalArgumentException
                but got NullPointerException“);
}
```

Example Inconsistency of Type 2: Fault in Code, Correct Comment

```
/*…
* @param anchor the anchor (<code>null</code> not permitted).
*/
void setRotationAnchor(TextAnchor anchor)
```

Example Inconsistency of Type 2: Fault in Code, Correct Comment

```
/*…
* @param anchor the anchor (<code>null</code> not permitted).
*/
void setRotationAnchor(TextAnchor anchor)
```



Example Inconsistency of Type 2: Fault in Code, Correct Comment

```
/*…
* @param anchor the anchor (<code>null</code> not permitted).
*/
void setRotationAnchor(TextAnchor anchor)
```

@tComment

```
public void test2() throws Throwable {
    ...CategoryPointerAnnotation var0 = new …CategoryPointerAnnotation(
                        “$0.00”, (java.lang.Comparable)'#', 10.0d, 10.0d);
    org.jfree.ui.TextAnchor var1 = null;
    try {
                var0.setRotationAnchor(var1);
                 fail("Expected exception but got Normal Execution");
            } catch (Exception expected) {}
  }
```

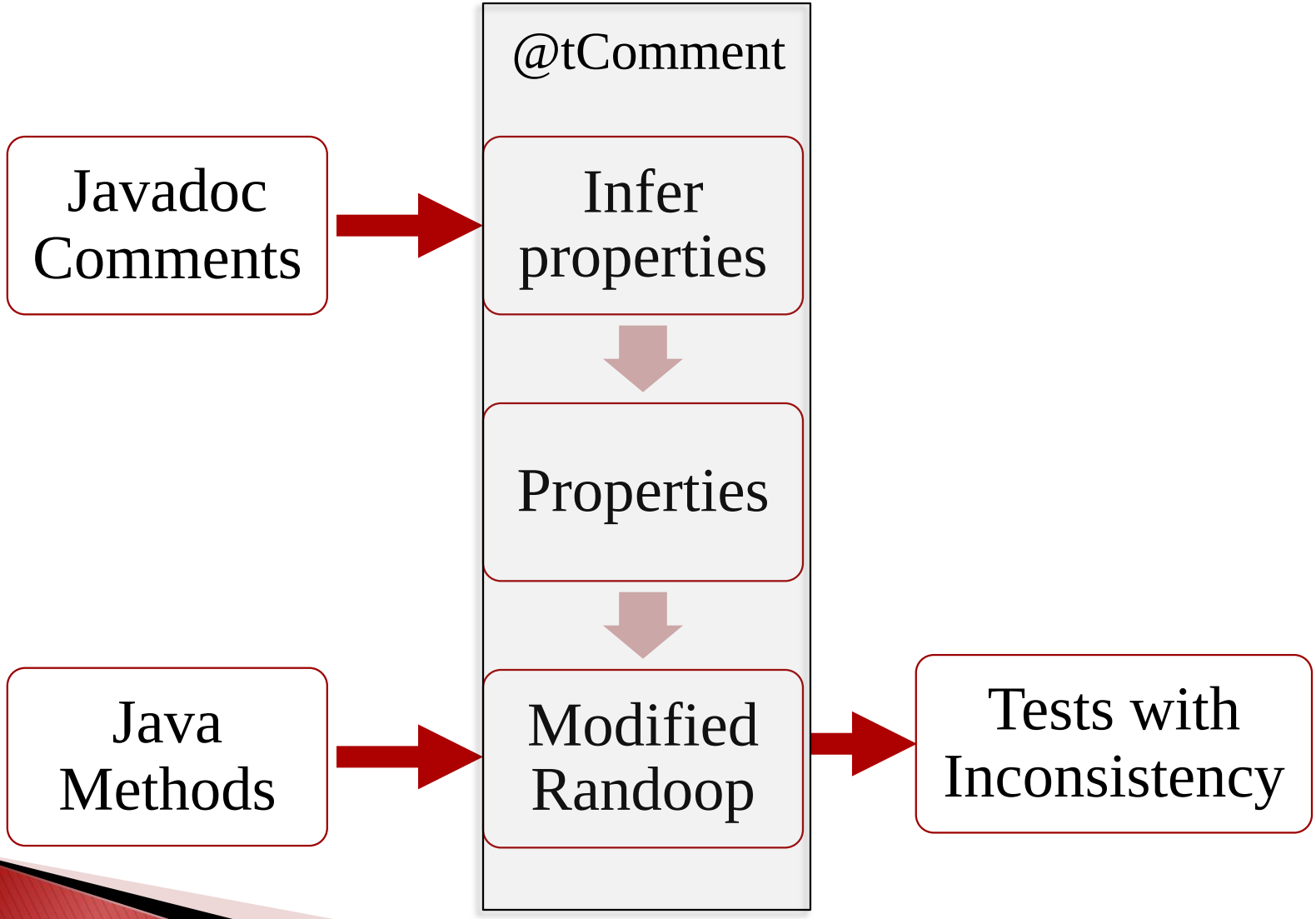
Example Inconsistency of Type 2: Fault in Code, Correct Comment

```
/*…
* @param anchor the anchor (<code>null</code> not permitted).
*/
void setRotationAnchor(TextAnchor anchor)
```

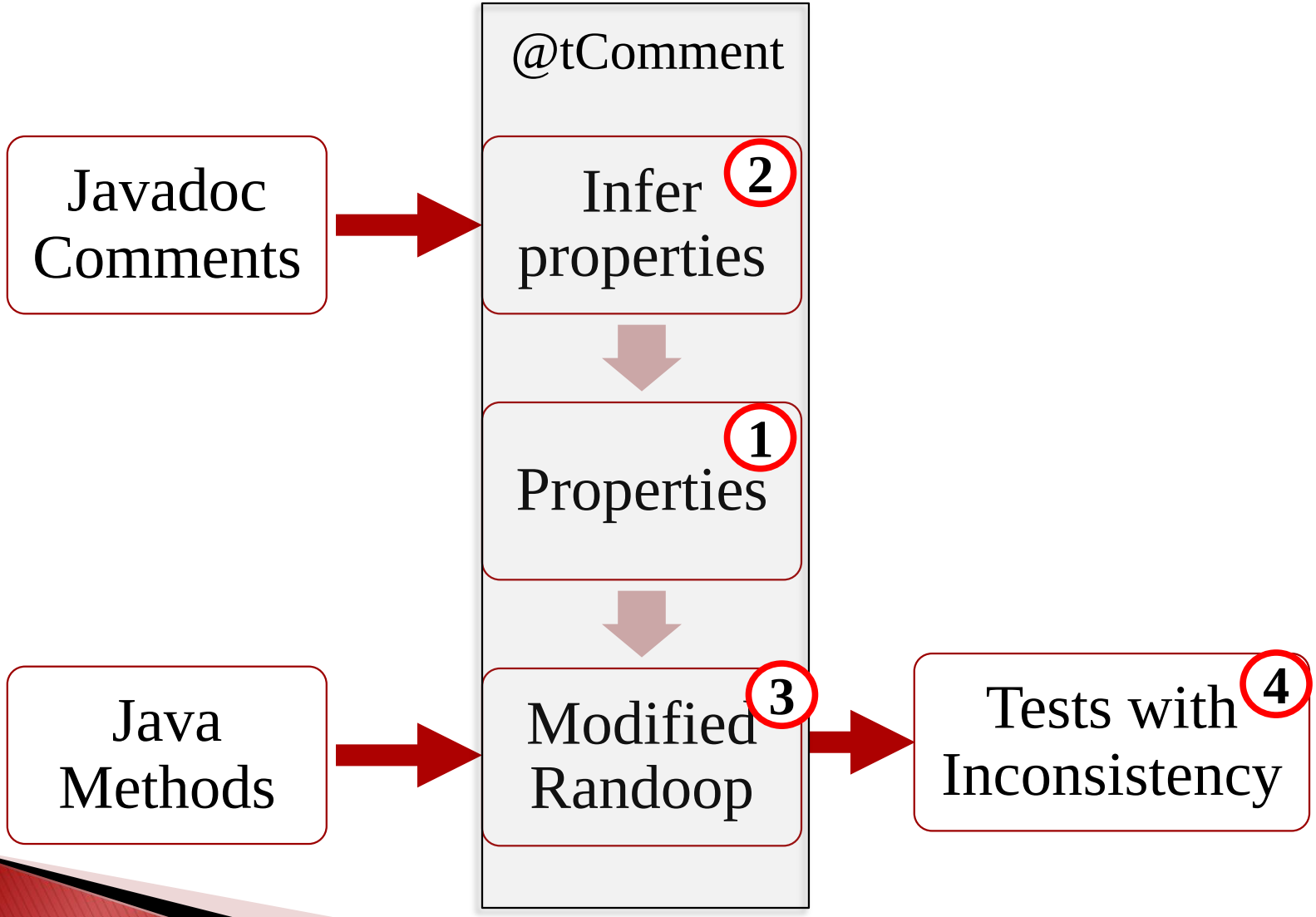
✓ Confirmed & fixed by JFreeChart developers


```
public void test2() throws Throwable {
    ...CategoryPointerAnnotation var0 = new …CategoryPointerAnnotation(
                        “$0.00”, (java.lang.Comparable)'#', 10.0d, 10.0d);
    org.jfree.ui.TextAnchor var1 = null;
    try {
                var0.setRotationAnchor(var1);
                 fail("Expected exception but got Normal Execution");
              } catch (Exception expected) {}
  }
```

@tComment Design



@tComment Design



Properties for Method Parameters: If the Parameter is Null…

1

- *Null Normal*
  - …, the method should execute normally (no exception)
  - Eg: *@param* predicate the predicate to use, may be null

Properties for Method Parameters: If the Parameter is Null…

1

- *Null Normal*
  - …, the method should execute normally (no exception)
  - Eg: *@param* predicate the predicate to use, may be null
- *Null Any Exception*
  - …, the method should throw some exception
  - Eg: @param collection the collection to add to, must not be null

Properties for Method Parameters: If the Parameter is Null…

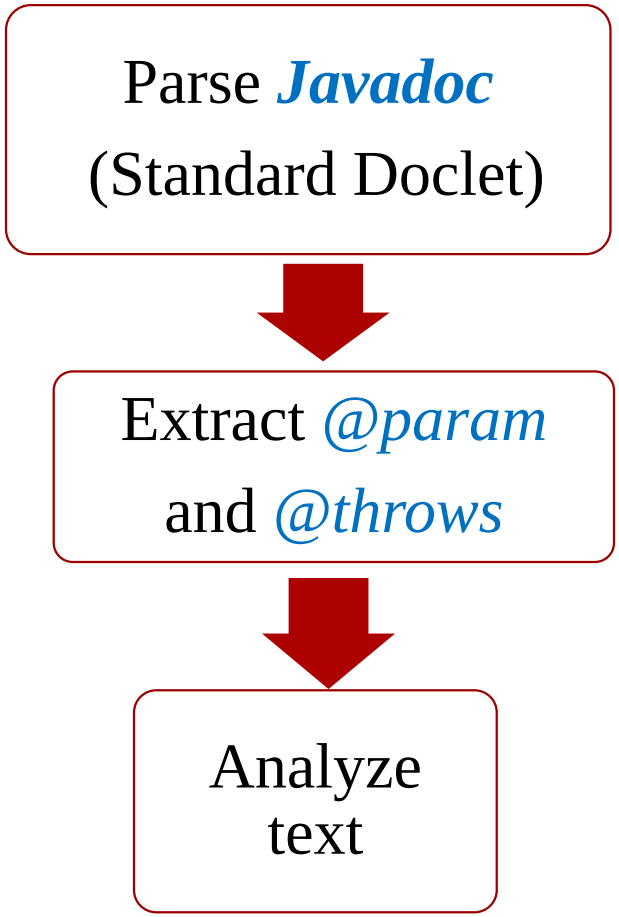
1

- *Null Normal*
  - …, the method should execute normally (no exception)
  - Eg: *@param* predicate the predicate to use, may be null
- *Null Any Exception*
  - …, the method should throw some exception
  - Eg: @param collection the collection to add to, must not be null
- *Null Specific Exception* (*id==null => IllegalArgumentExc*)
  - …, the method should throw a specific type of exception
  - Eg: *@throws* IllegalArgumentException if the id is null

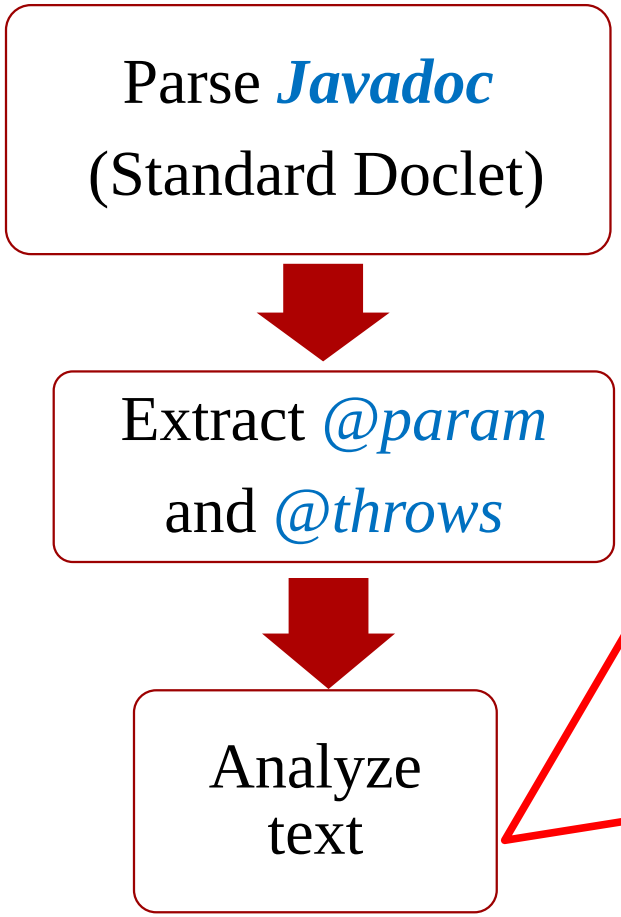
Properties for Method Parameters: If the Parameter is Null…

- *Null Normal*
  - …, the method should execute normally (no exception)
  - Eg: *@param* predicate the predicate to use, may be null
- *Null Any Exception*
  - …, the method should throw some exception
  - Eg: @param collection the collection to add to, must not be null
- *Null Specific Exception* (*id==null => IllegalArgumentExc*)
  - …, the method should throw a specific type of exception
  - Eg: *@throws* IllegalArgumentException if the id is null
- *Null Unknown*
  - …, the method behavior is unknown
  - Eg: *@param* array the array over which to iterate

Inferring Properties from Comments ②



Inferring Properties from Comments 2



- Negation words (±3 distance within ***null***)
  →*Null Any Exception*
- No negation words
  →*Null Normal*
- ***null*** in *@throws* tag, then searches the list of parameter names. If found
  →*Null Specific Exception*
  param == null =>SpecificException
- Generates multiple properties for *"or"* and *"either"* in the @throws tag

Randoop Test Generation [ICSE'07]

while ***timeLimit*** not reached:

a. Create a new sequence

1. Randomly pick a method call $m(T_1...T_k)$
2. For each parameter, randomly pick a sequence S_i that can construct the object for that parameter
   - Select null with some probability (***nullRatio***)
3. Create new sequence $S_{new} \leftarrow S_{1;}...S_k$; $m(var_1...var_k)$

b. Classify the new sequence S_{new}

- Execute & check contract violations?
  - Yes - Output as failure-revealing test case
  - No - Add to sequences if not redundant, else discard

@tComment Modified Randoop

3

while ***timeLimit*** not reached:

a. **Create a new sequence**

1. Randomly pick a method call $m(T_1...T_k)$
2. For each parameter, randomly pick a sequence S_i that can construct the object for that parameter
   - Select null with some probability (***nullRatio***)
3. Create new sequence $S_{new} \leftarrow S_{1;}...S_k; m(var_1...var_k)$

b. **Classify the new sequence S_{new}**

- Execute & check contract violations & @tComment properties?
  - Violate contract - Output as failure-revealing tests
  - Violate properties - Output as comment-code inconsistency
  - No - Add to sequences if not redundant, else discard

From Properties to Inconsistency

4

```
/**
* @param logger logger, may not be null.
* @param sourceClass source class, may be null.
* @param sourceMethod method, may be null.
*/
void exiting(Logger logger, String sourceClass, String sourceMethod)
```

From Properties to Inconsistency

4

```
/**
* @param logger logger, may not be null.
* @param sourceClass source class, may be null.
* @param sourceMethod method, may be null.
*/
void exiting(Logger logger, String sourceClass, String sourceMethod)
```

From Properties to Inconsistency

4

```
/**
* @param logger logger, may not be null.
* @param sourceClass source class, may be null.
* @param sourceMethod method, may be null.
*/
void exiting(Logger logger, String sourceClass, String sourceMethod)
```

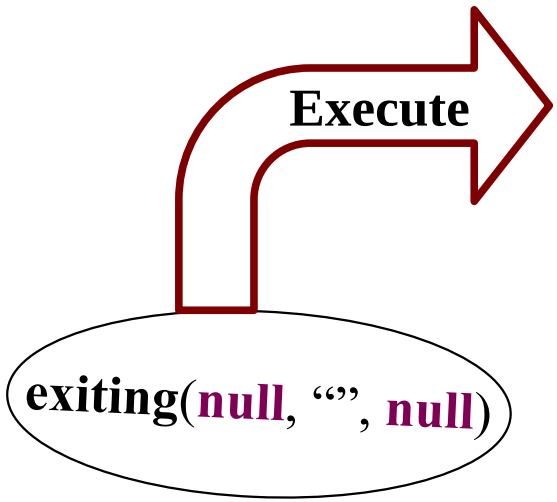
From Properties to Inconsistency

4

```
/**
* @param logger logger, may not be null.
* @param sourceClass source class, may be null.
* @param sourceMethod method, may be null.
*/
void exiting(Logger logger, String sourceClass, String sourceMethod)
```

Null Any Exception

Null Normal



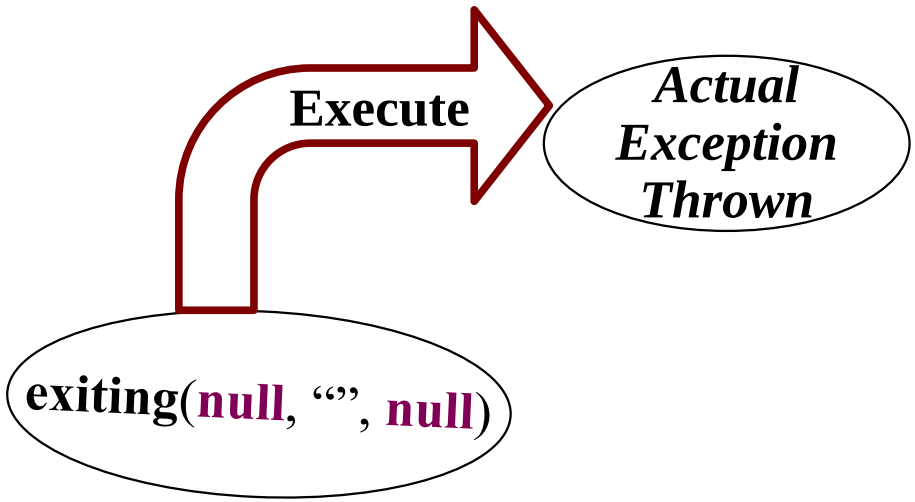
From Properties to Inconsistency

4

```
/**
* @param logger logger, may not be null.
* @param sourceClass source class, may be null.
* @param sourceMethod method, may be null.
*/
void exiting(Logger logger, String sourceClass, String sourceMethod)
```

Null Any Exception

Null Normal



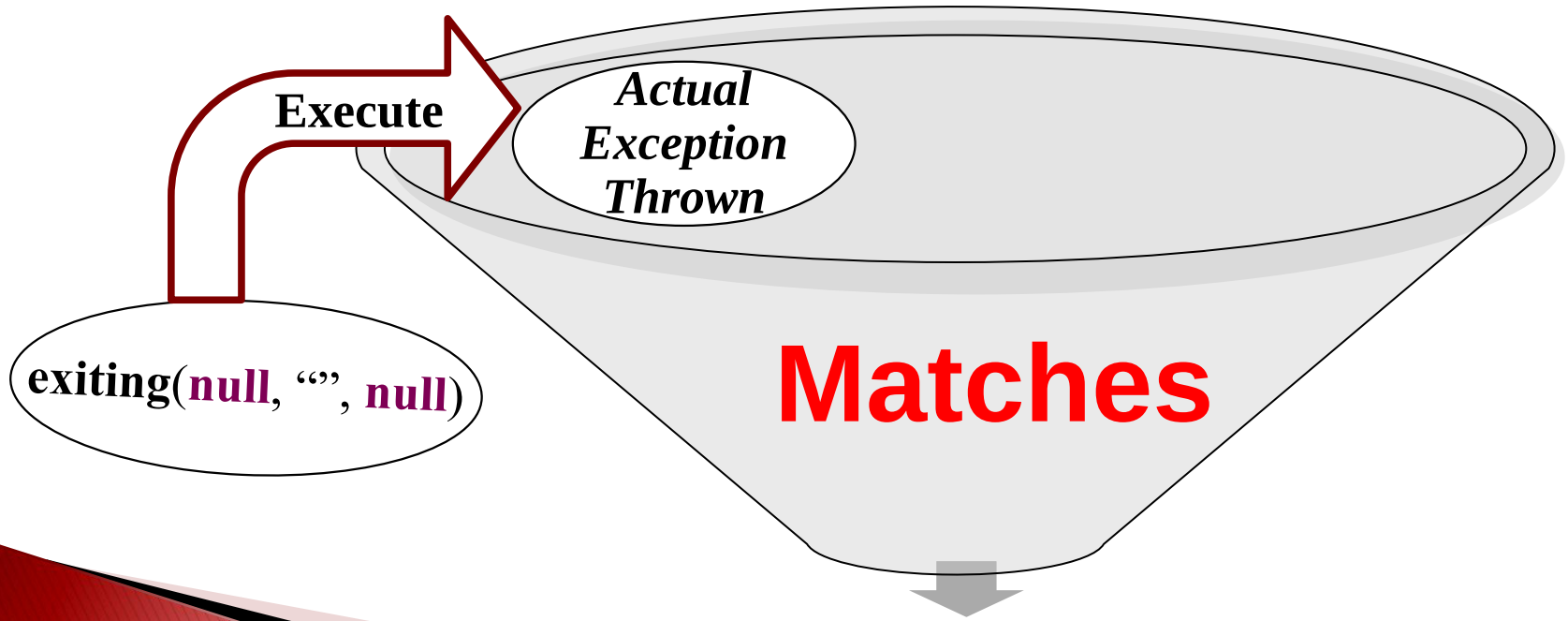
From Properties to Inconsistency

4

```
/**
* @param logger logger, may not be null.
* @param sourceClass source class, may be null.
* @param sourceMethod method, may be null.
*/
void exiting(Logger logger, String sourceClass, String sourceMethod)
```

Null Any Exception

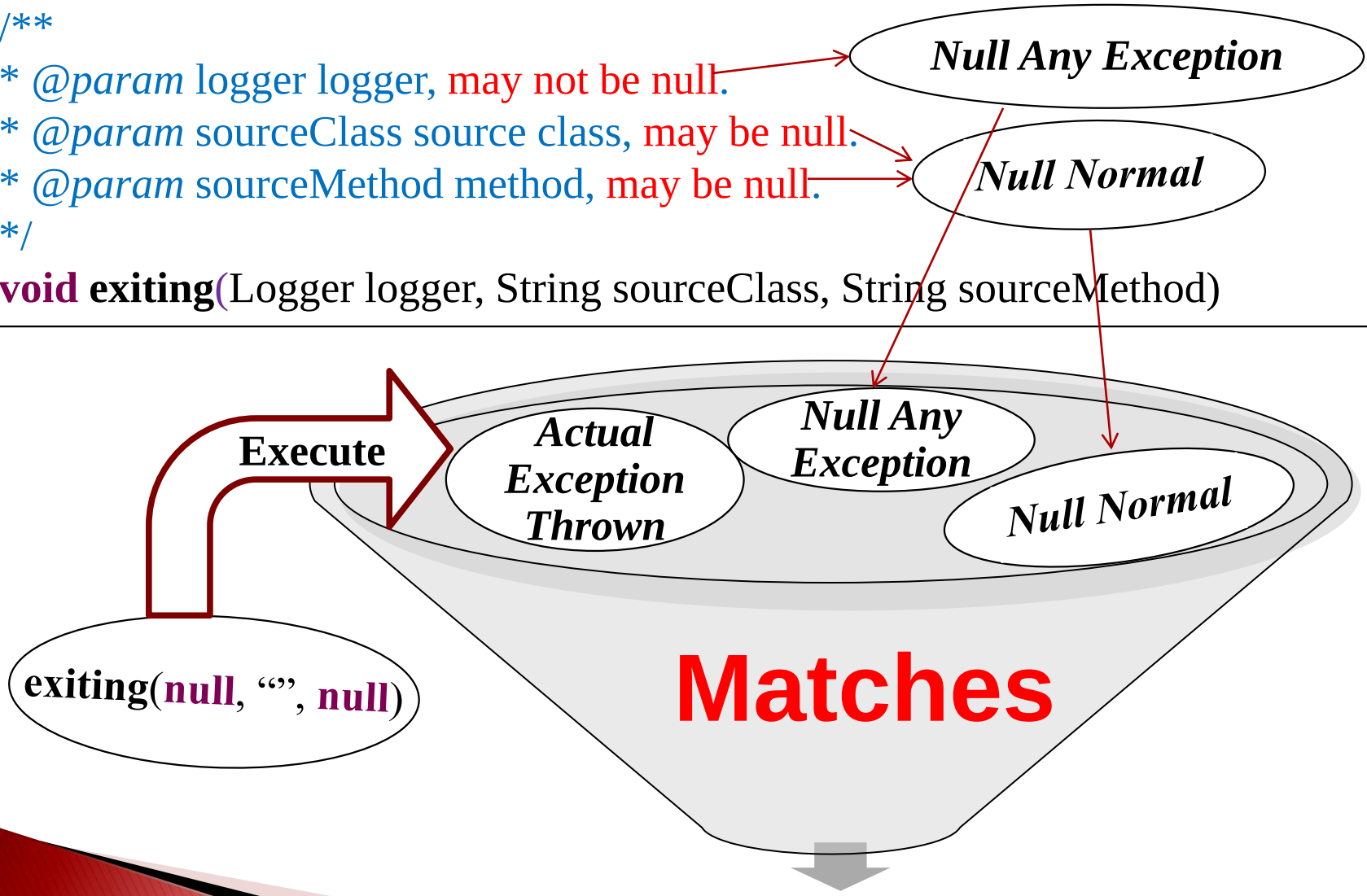
Null Normal



From Properties to Inconsistency

4

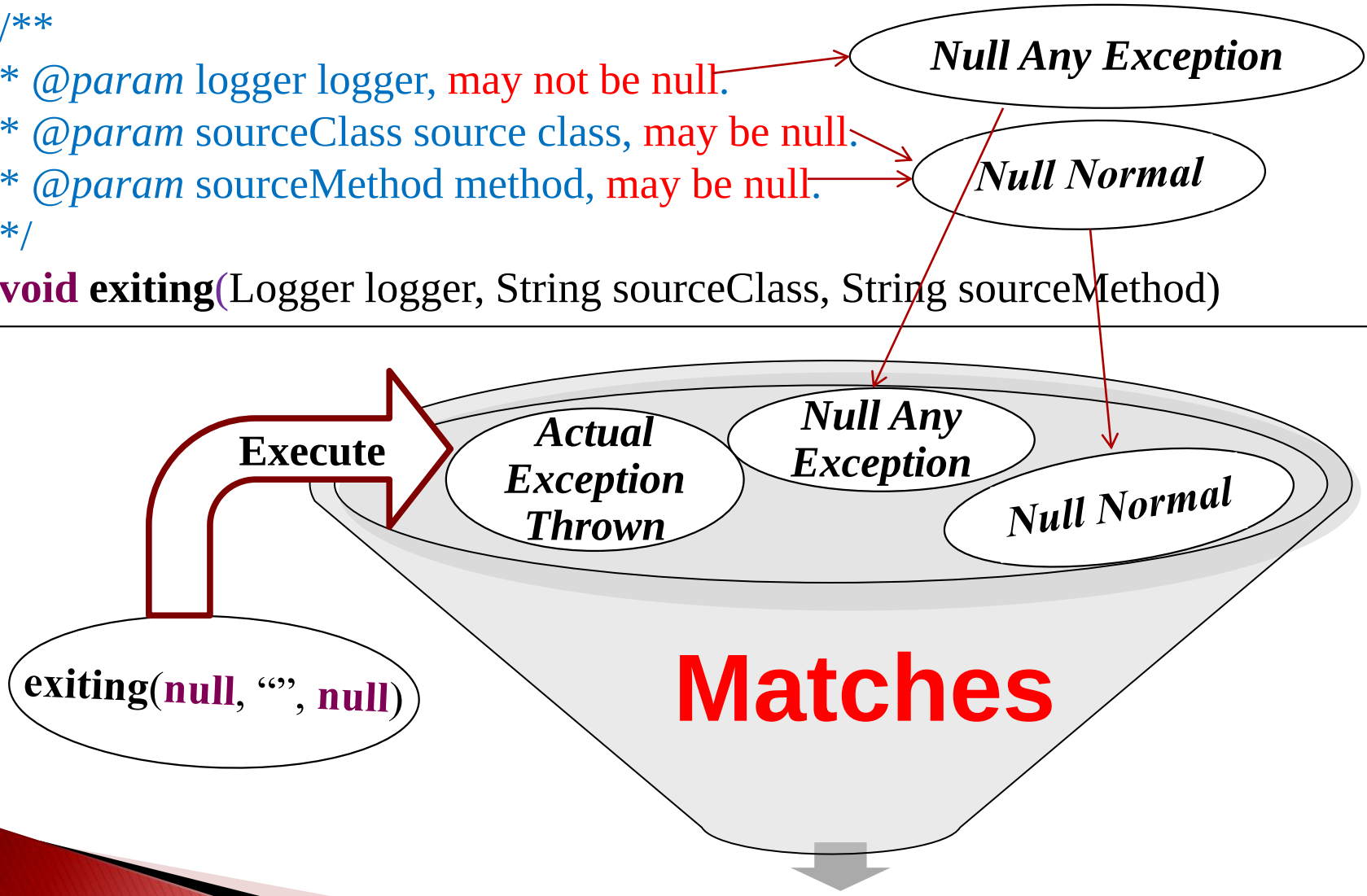
```
/**
* @param logger logger, may not be null.
* @param sourceClass source class, may be null.
* @param sourceMethod method, may be null.
*/
void exiting(Logger logger, String sourceClass, String sourceMethod)
```



From Properties to Inconsistency

4

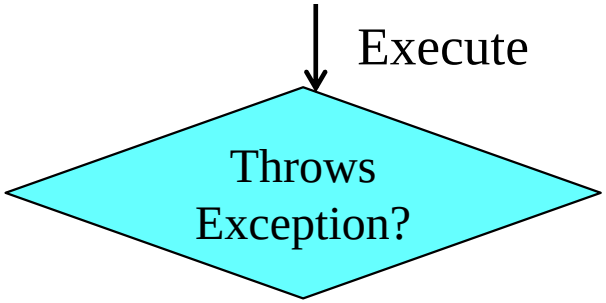
```
/**
* @param logger logger, may not be null.
* @param sourceClass source class, may be null.
* @param sourceMethod method, may be null.
*/
void exiting(Logger logger, String sourceClass, String sourceMethod)
```



isCommentCodeInconsistency?

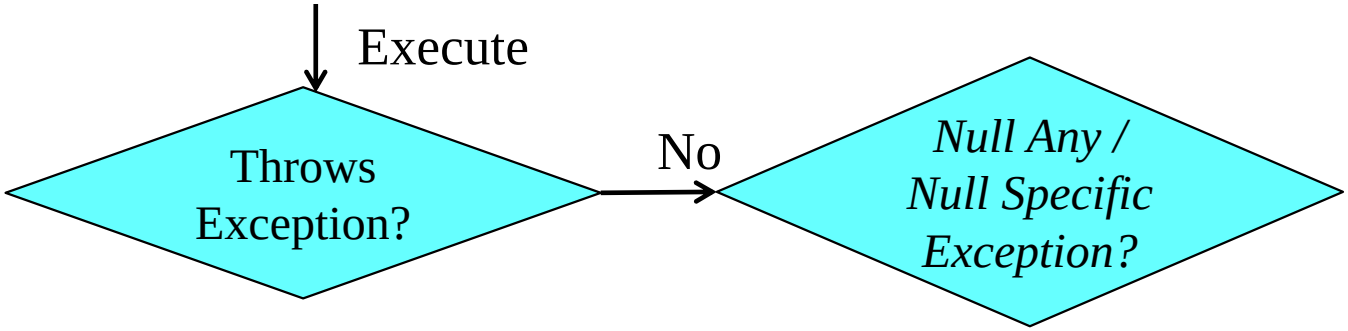
Different kinds of Matches

4



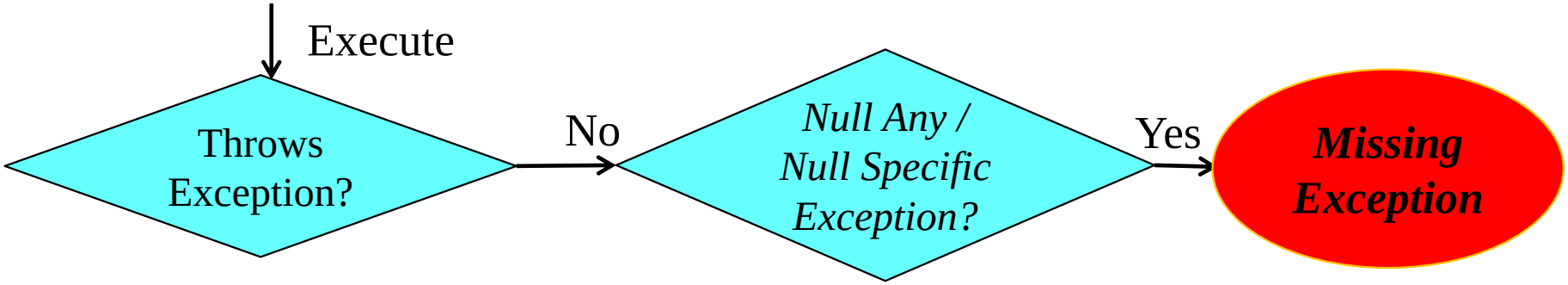
Different kinds of Matches

4



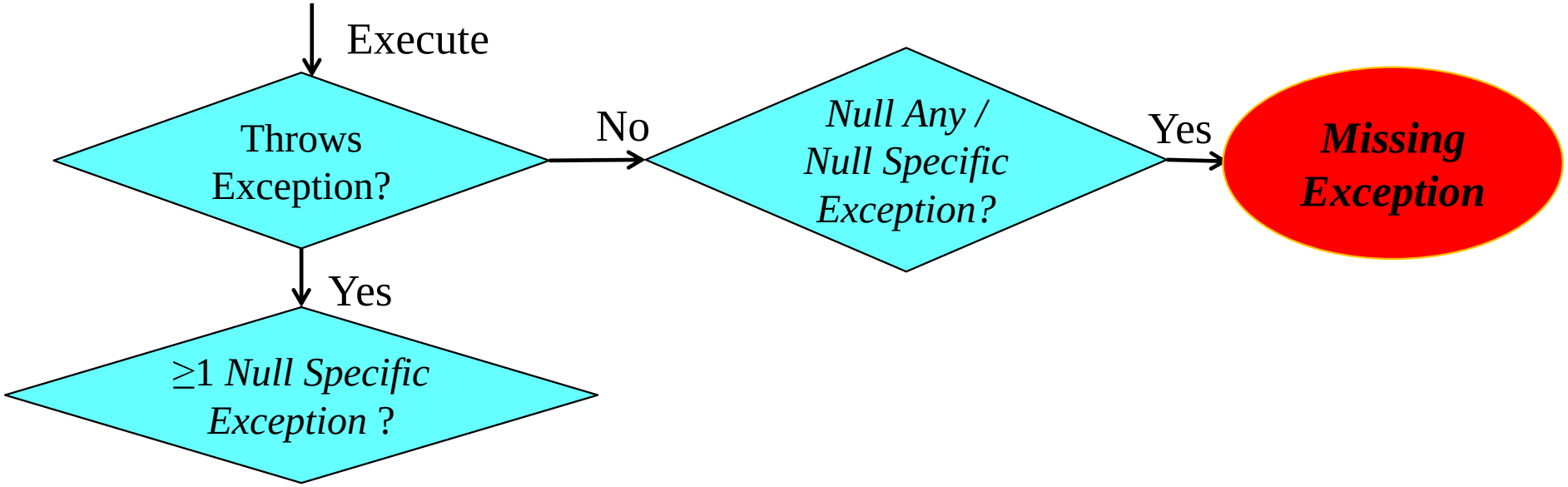
Different kinds of Matches

4



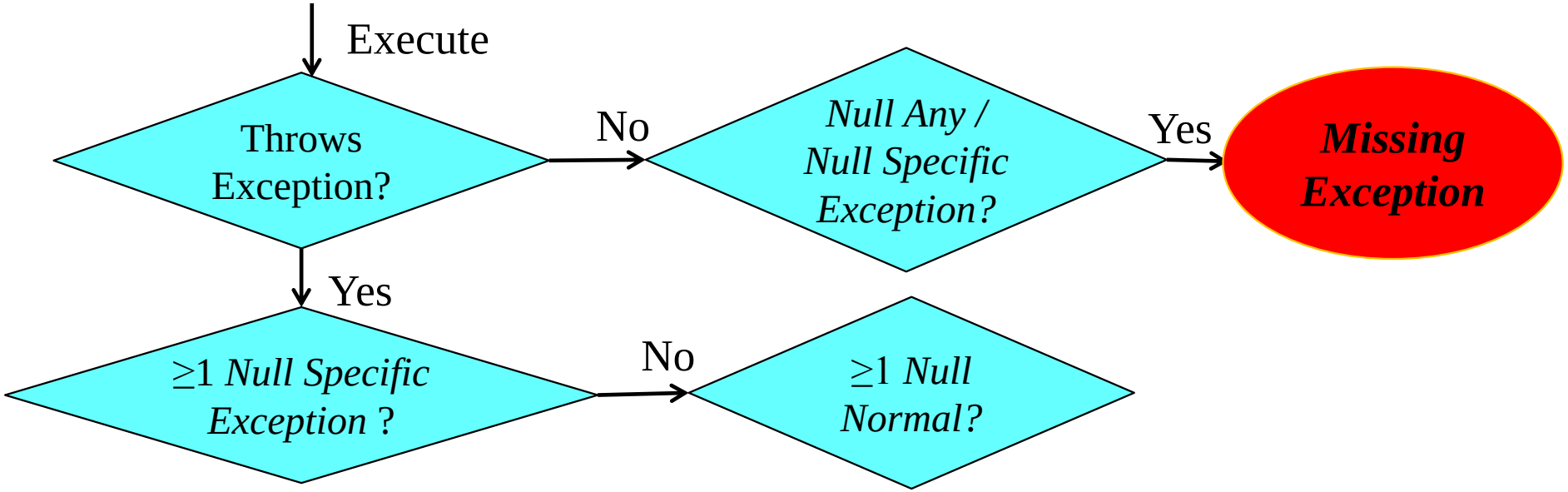
Different kinds of Matches

4



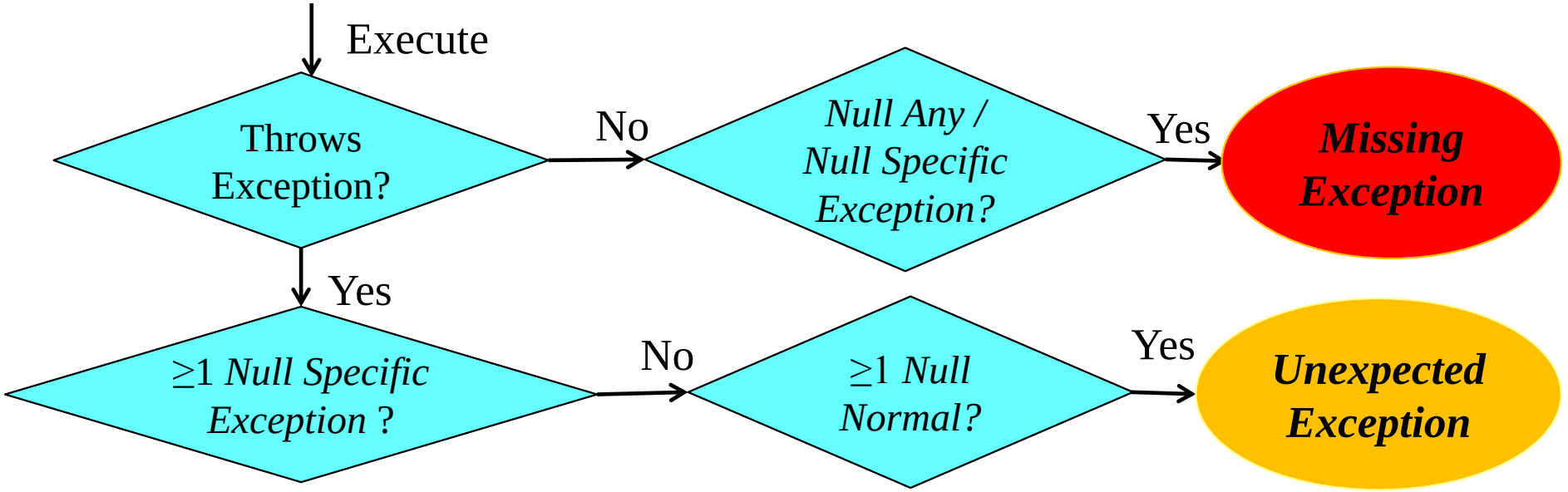
Different kinds of Matches

4



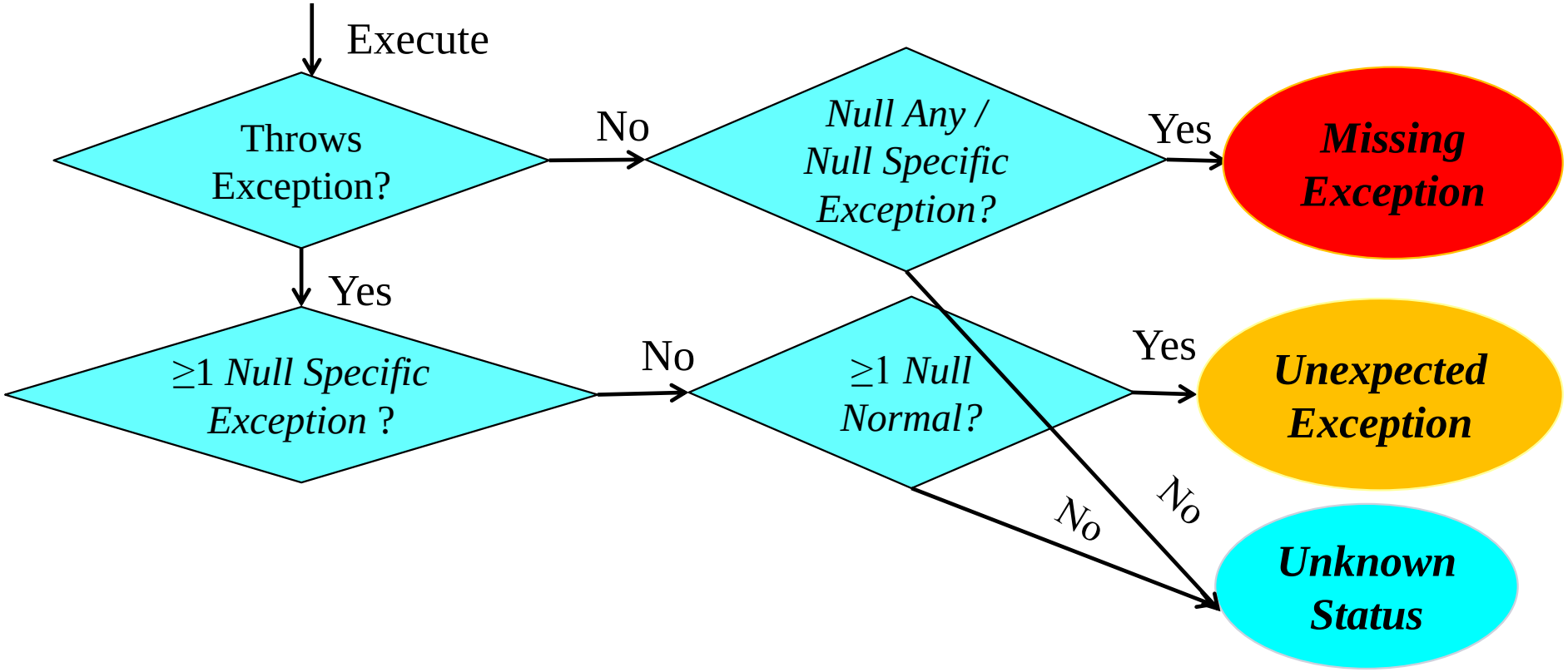
Different kinds of Matches

4



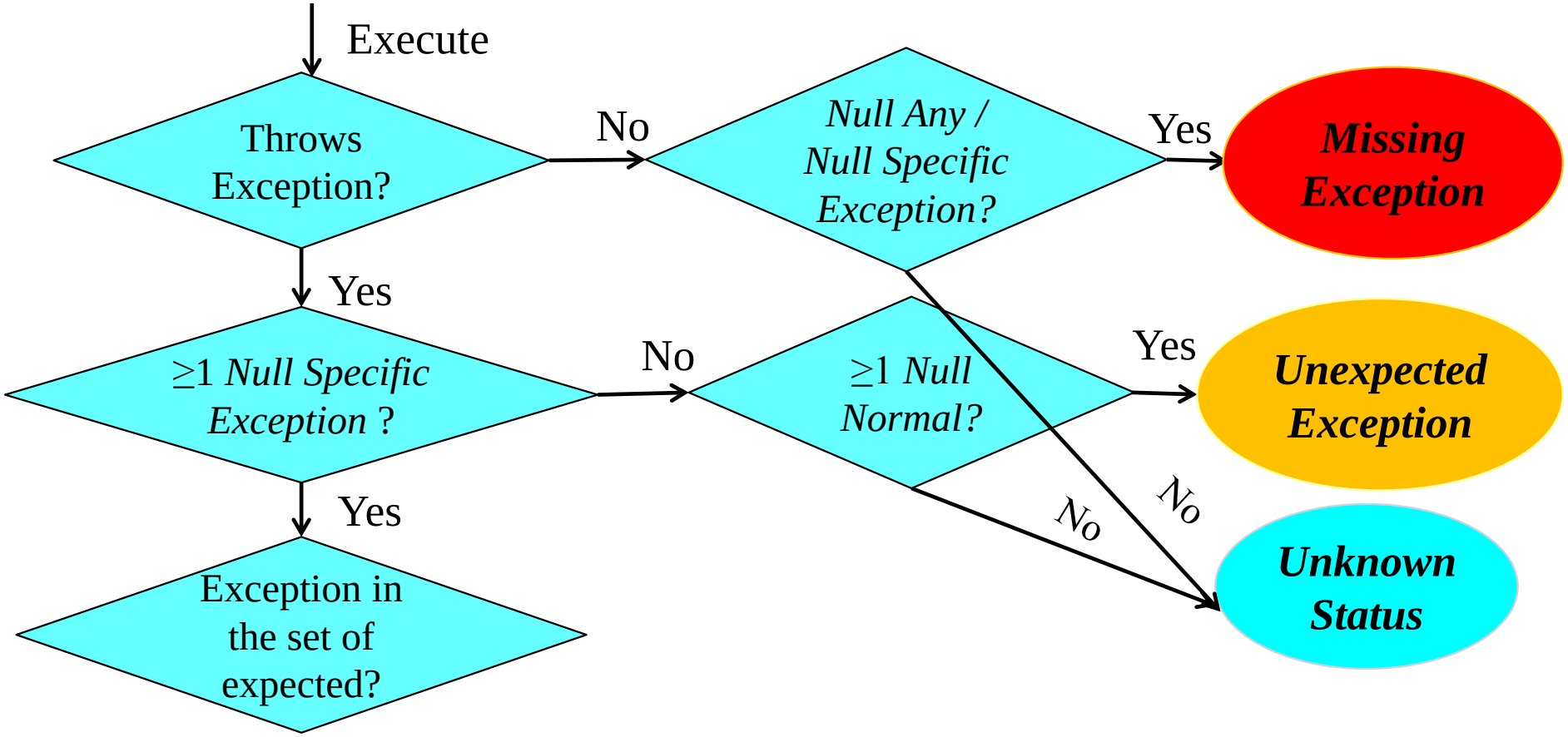
Different kinds of Matches

4



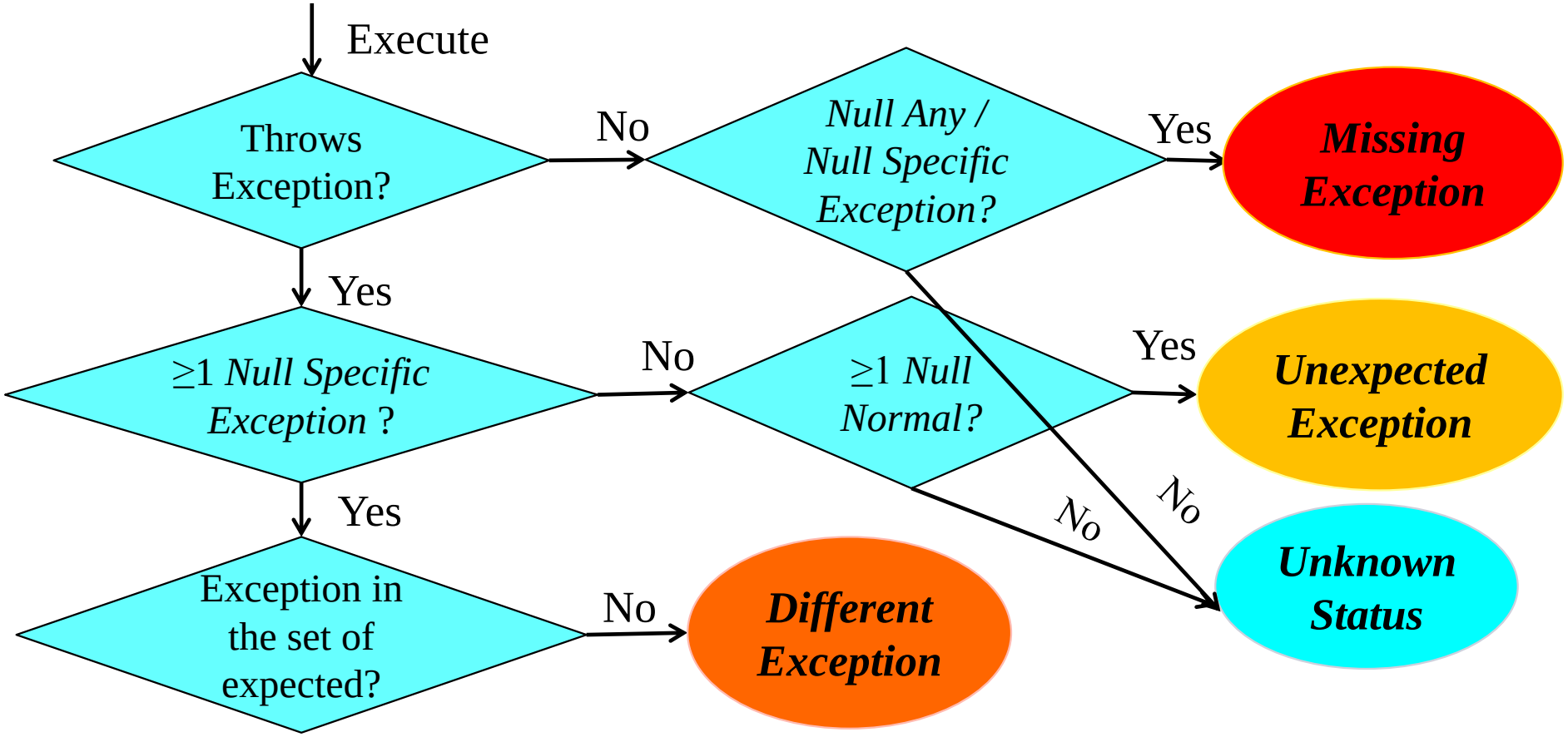
Different kinds of Matches

4

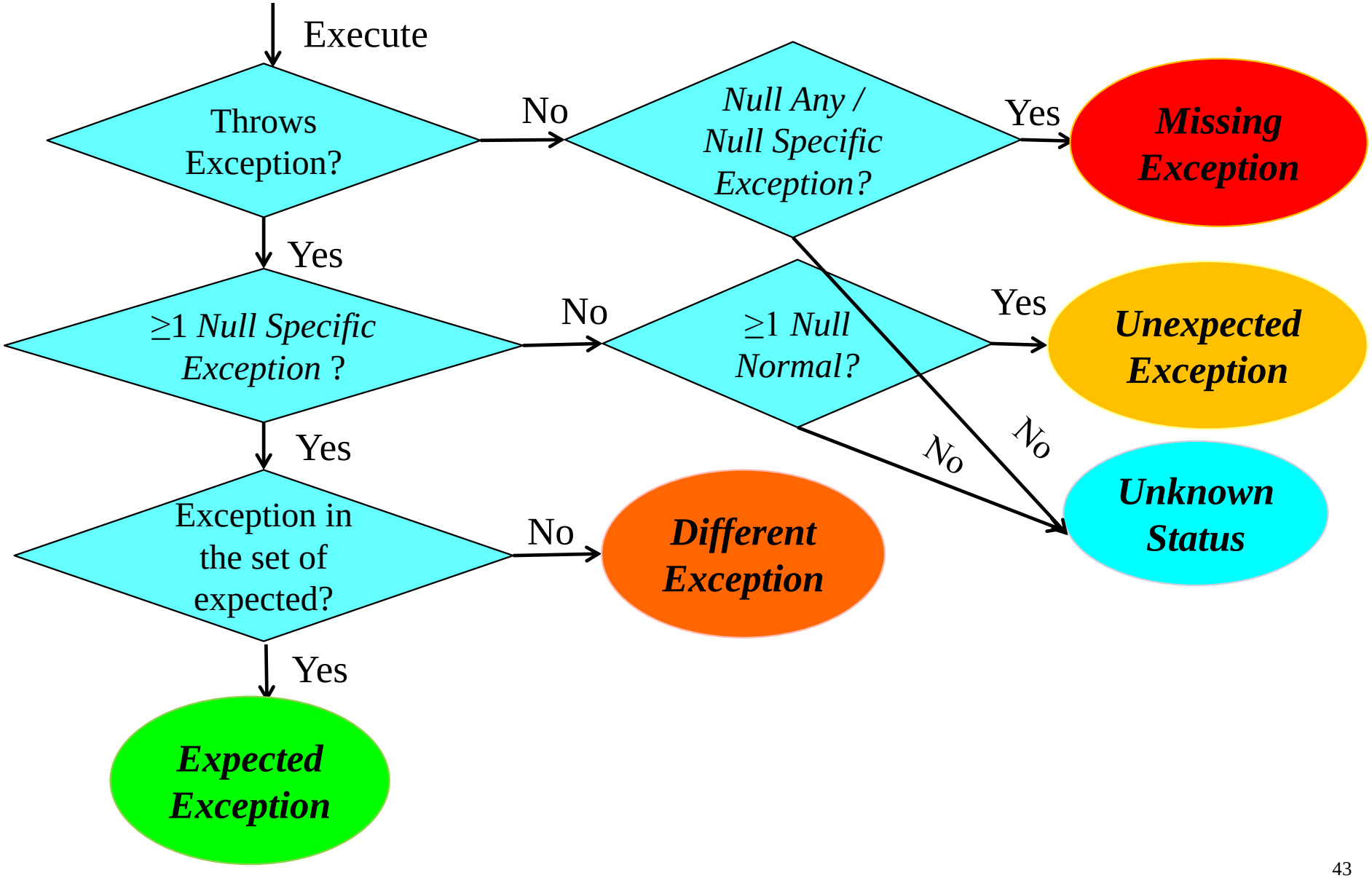


Different kinds of Matches

4

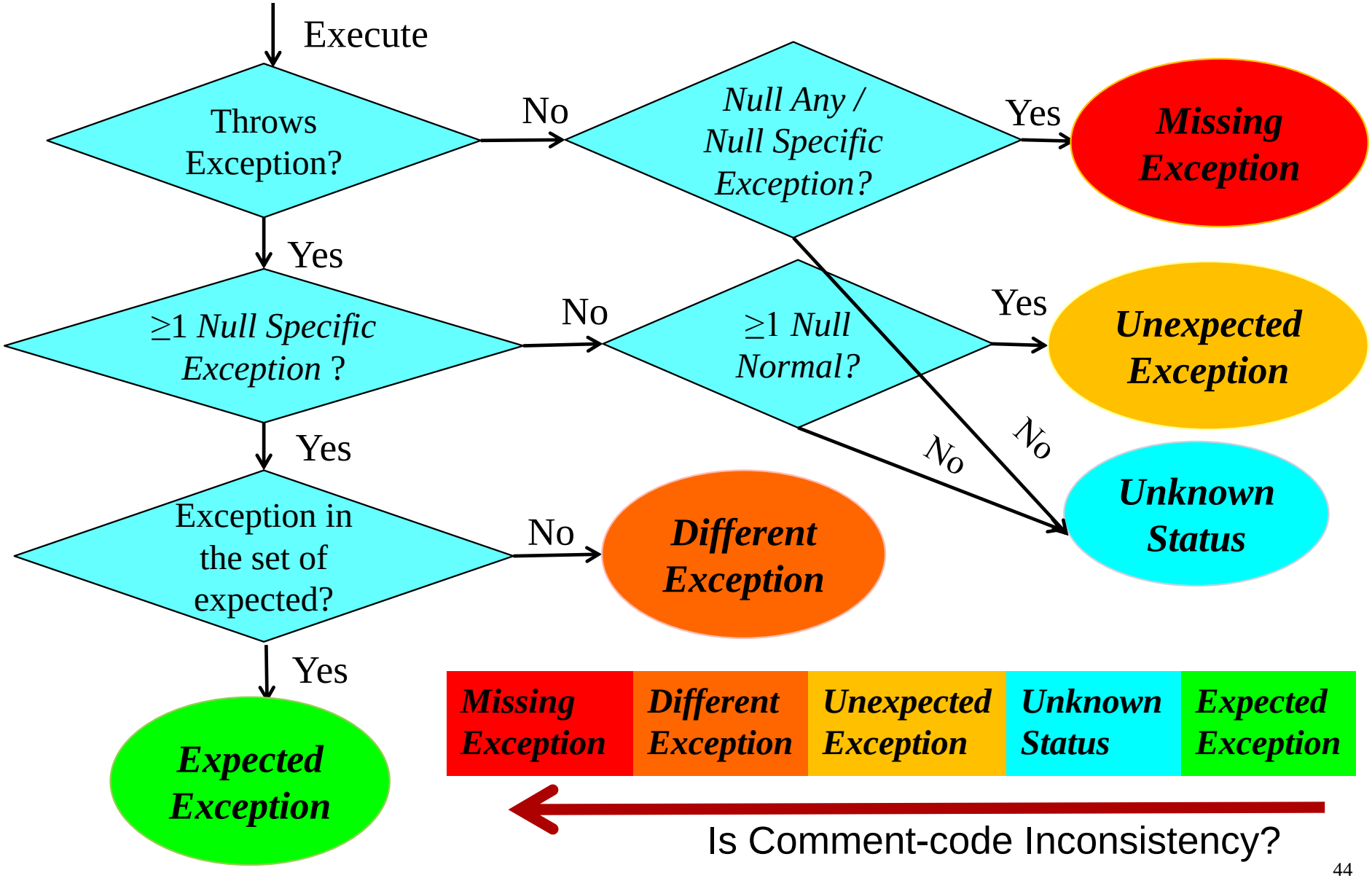


Different kinds of Matches



Different kinds of Matches

4



Evaluation: Experimental Setup

- 7 open source projects
- Ran @tComment to infer properties
- Ran Modified Randoop to check properties
  - Varying 2 options on Modified Randoop
    - ***nullRatio*** *(* How often *null* is chosen as input for a method?)
    - ***timeLimit*** (How long should Modified Randoop take for generating tests before stopping?)
  - Best configuration for all projects
    - ***nullRatio*** = **0.6**
    - ***timeLimit*** **= *3600s*** *(reached plateau effect after that)*

Subject Projects

Project	Description	# LOC	# Classes	# Methods
Collections	Collection library and utilities	19,417	274	3,874
GlazedLists	List transformations in Java	19,203	239	2,753
JFreeChart	Chart creator	51,376	396	6,205
JodaTime	Date and time library	18,428	154	3,887
Log4j	Logging service	14,452	221	2,115
Lucene	Text search engine	38,051	422	5,222
Xalan	XML transformations	53,642	510	5,404

Number of Matches

Project	Missing Exception	Different Exception	Unexpected Exception	Unknown Status	Expected Exception	Tested Properties
Collections	12	4	6	94	36	115
GlazedLists	0	0	6	151	1	11
JFreeChart	1	0	2	127	6	42
JodaTime	3	0	13	37	3	31
Log4j	1	0	3	186	152	179
Lucene	4	0	2	368	2	12
Xalan	9	0	2	544	32	43
Total	30	4	34	1507	232	433

Number of Matches

Project	Missing Exception	Different Exception	Unexpected Exception	Unknown Status	Expected Exception	Tested Properties
Collections	12	4	6	94	36	115
GlazedLists	0	0	6	151	1	11
JFreeChart	1	0	2	127	6	42
JodaTime	3	0	13	37	3	31
Log4j	1	0	3	186	152	179
Lucene	4	0	2	368	2	12
Xalan	9	0	2	544	32	43
Total	30	4	34	1507	232	433

Tested only a fraction of properties inferred

True Inconsistencies & False Alarms

Project	Missing Exception = TI + FA	Different Exception = TI + FA	Unexpected Exception = TI + FA
Collections	12 + 0	3 + 1	0 + 6
GlazedLists	0 + 0	0 + 0	1 + 5
JFreeChart	1 + 0	0 + 0	2 + 0
JodaTime	3 + 0	0 + 0	0 + 13
Log4j	1 + 0	0 + 0	0 + 3
Lucene	0 + 4	0 + 0	1 + 1
Xalan	4 + 5	0 + 0	0 + 2
Total	21+ 9	3 + 1	4 + 30

				Total
True Inconsistencies (TI)	21	3	4	28
False Alarms (FA)	9	1	30	40

Sources of False Alarm

Incorrect inference

- Wrong type of inferred properties

Missing properties

- Missing property (*Null Unknown*) causes unexpected exception

Incorrect/missing properties for another method

- Method under test depend on other method, other method causes unexpected exception

Comment Analysis Results

Project	Precision [%]			Recall [%]			Accuracy [%]
	Norm	Any	Spec	Norm	Any	Spec	
Collections	75	92	100	100	100	97	97
GlazedLists	100	100	100	100	100	100	100
JFreeChart	100	100	100	100	100	100	100
JodaTime	100	75	100	100	100	78	98
Log4j	100	100	100	100	100	100	100
Lucene	100	67	100	80	100	100	99
Xalan	50	100	100	100	100	100	99
Total/Overall	98	98	100	99	100	93	99

2 properties are inferred, 1 incorrect

High accuracy (97-100%)

Summary of Results

Comment-Code Inconsistency Detection

- Detected 28 comment-code inconsistencies
- 40 false alarms

Comment Analysis Result

- High accuracy of 97–100% without using NLP techniques
  - Javadoc comments are well-structured
  - Not much variance in paraphrases

Conclusion

- An inconsistency between comment and code is highly indicative of program faults
- @tComment checks consistency of Java method bodies and Javadoc comments properties related to null values and exceptions
- Evaluated on 7 open-source projects
- Discovered 28 inconsistencies and 12 were already fixed

Acknowledgement: Travel expenses sponsored by NSF CCF 07-46856 grants and ACM-W Scholarships for Attendance at Research Conferences.