Automatic Model Generation from Documentation for Java API Functions

Juan Zhai, Jianjun Huang, Shiqing Ma, Xiangyu Zhang,
Lin Tan, Jianhua Zhao, Feng Qin

Motivation

- Libraries
  - Part of software behaviors
  - As important as the software itself
- Challenges
  - Binary only, no source code
  - Implemented in different languages
  - Complex optimizations
  - Many more……
- Previous works: generate models manually

Automatically Generate Models for Libraries

How to model library behaviors?

-Q: How do we know how to use libraries?

-A: By reading API documents.

-Q: Can we model libraries from API documents?

-A: Yes. This is what we do.

public void add(int index, E element)

Inserts the specified element at the specified position in this list.

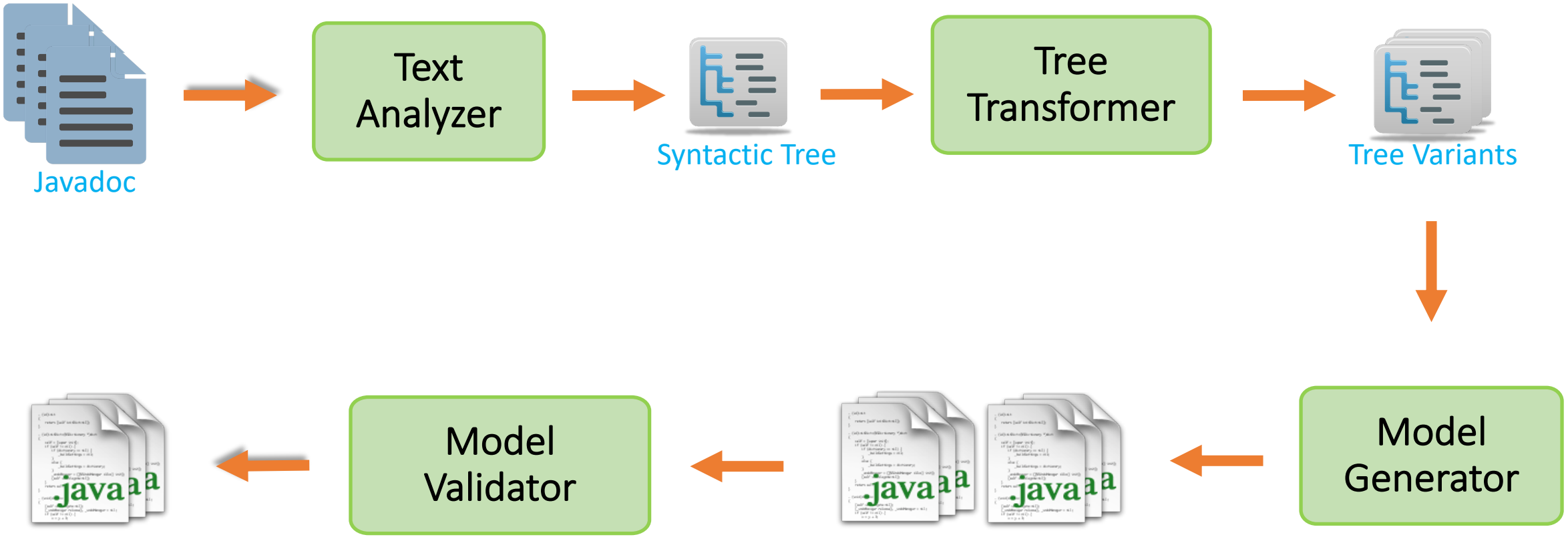
Shifts the element currently at that position (if any),
and any subsequent elements to the right.

Throws:

IndexOutOfBoundsException: if the index is out of range
(index<0||index>size())

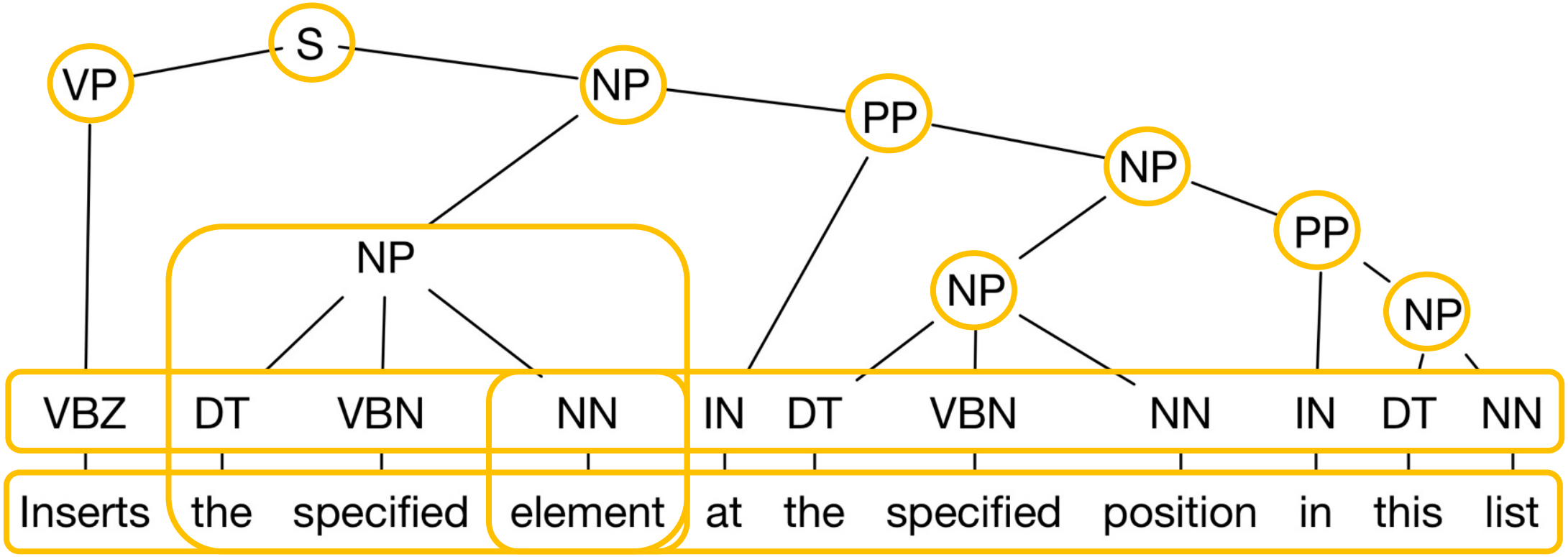
```
public void add(int index, E elememt){
    if(index<0 || index>size())
        throw new IndexOutOfBoundsException();
    size = size + 1;
    for(int i=size-1; i>index; i--)
        elements[i] = elements[i-1];
    elements[index] = element;
}
```

Design: Overview



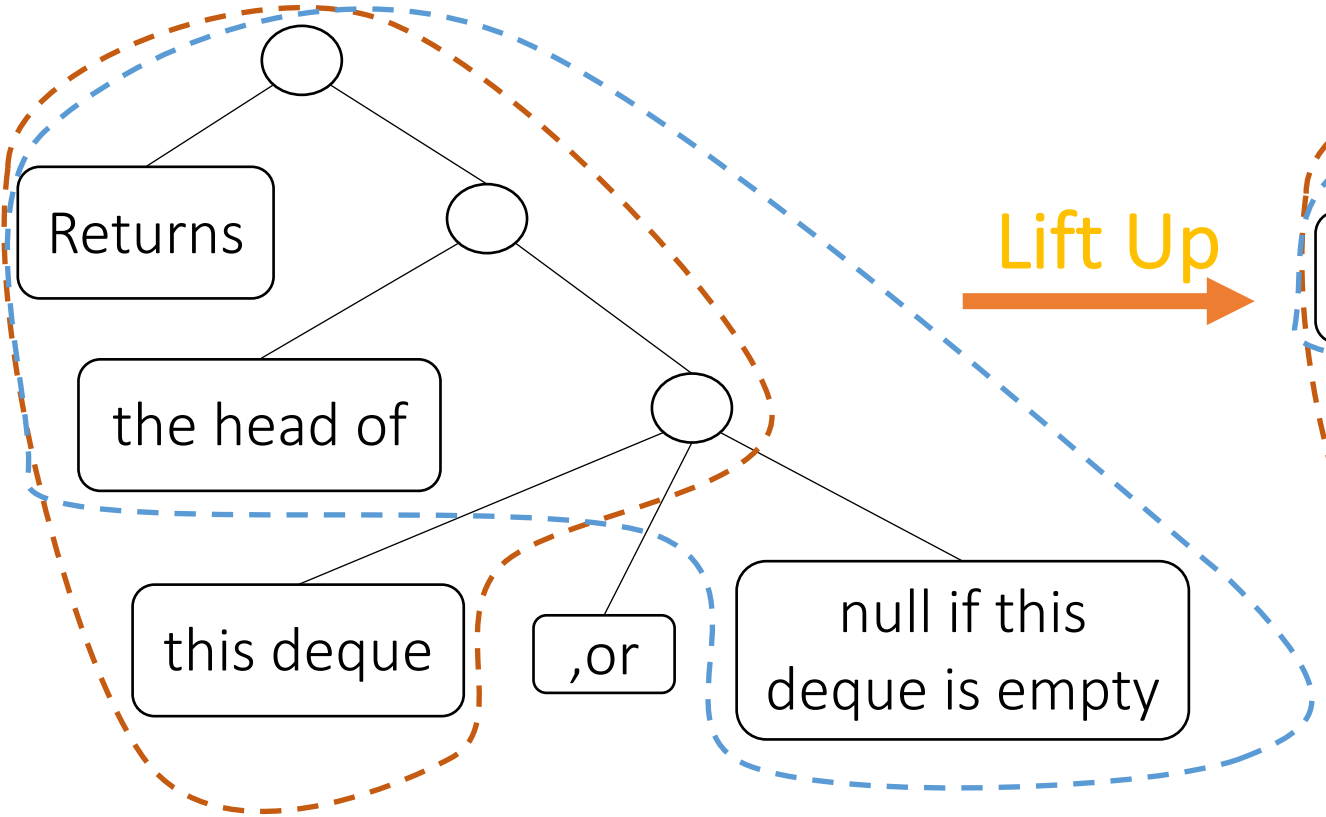
Design: Text Analyzer

- State-of-the-art work: Stanford Parser

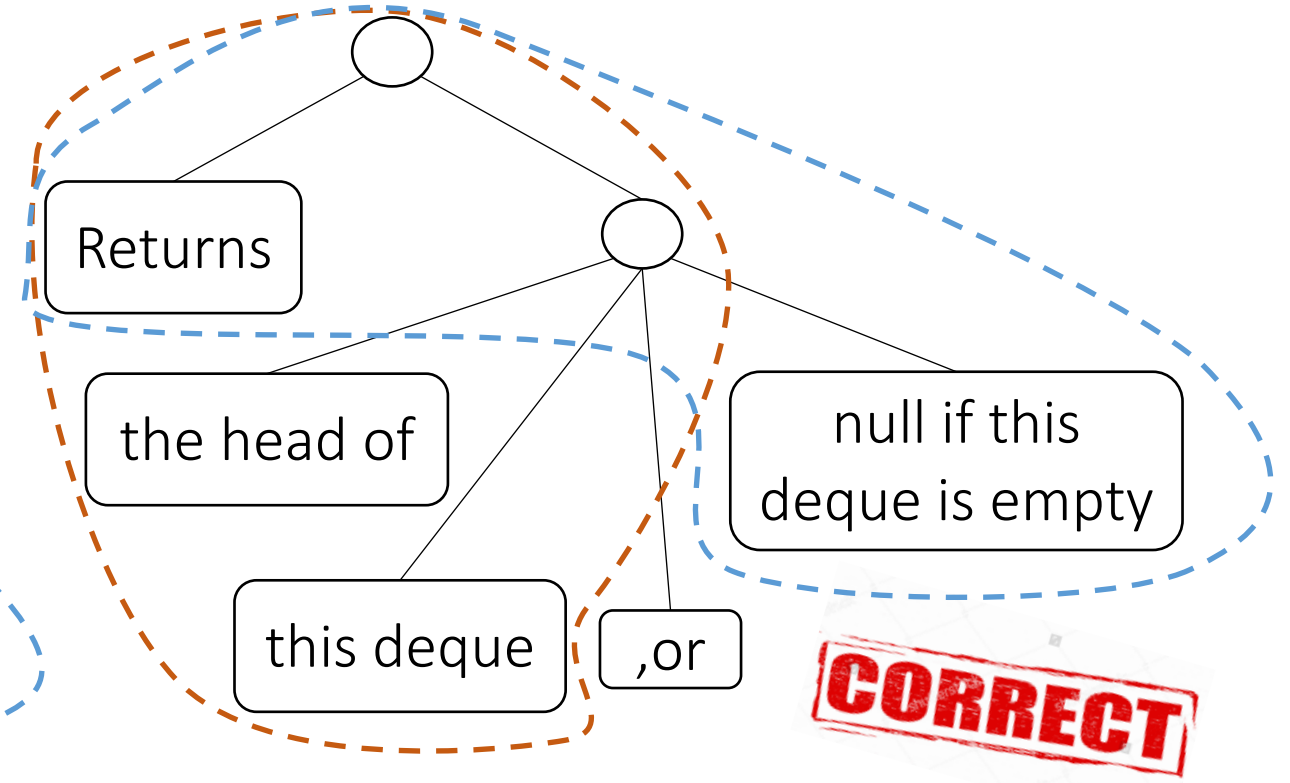


Not perfect because of the nature (e.g. ambiguity) of natural languages

Returns the head of this deque, or null if this deque is empty.



Lift Up



Returns the head of *this deque*,
OR returns the head of *null if this deque is empty*.

Returns the head of *this deque*,
OR returns *null if this deque is empty*.

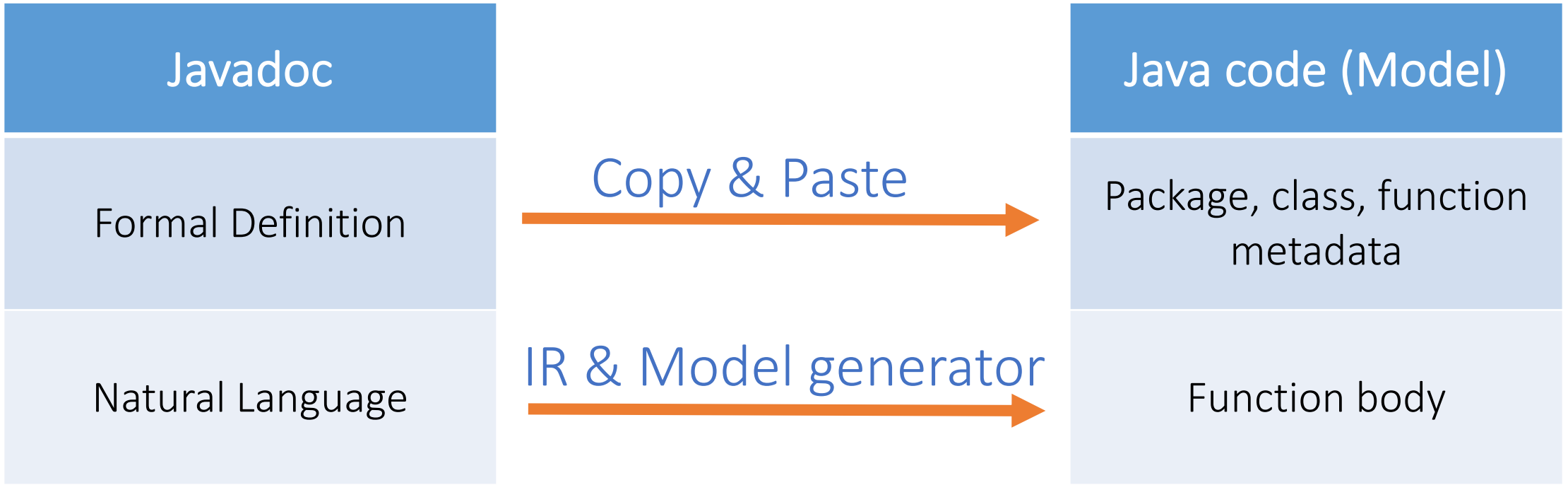
Design: Tree Node Transformer

- Ambiguities of natural languages
  - *K* tree candidates
  - Highest-scoring tree is not always the correct one
  - Too much time to try all candidates

- **Observation:**
  - Caused by phrases starting with *“, or”* and *“, and”*

- Solution:
  - **Lift up** & **Push down** *“, or”*, *“, and”* and *all their right siblings*

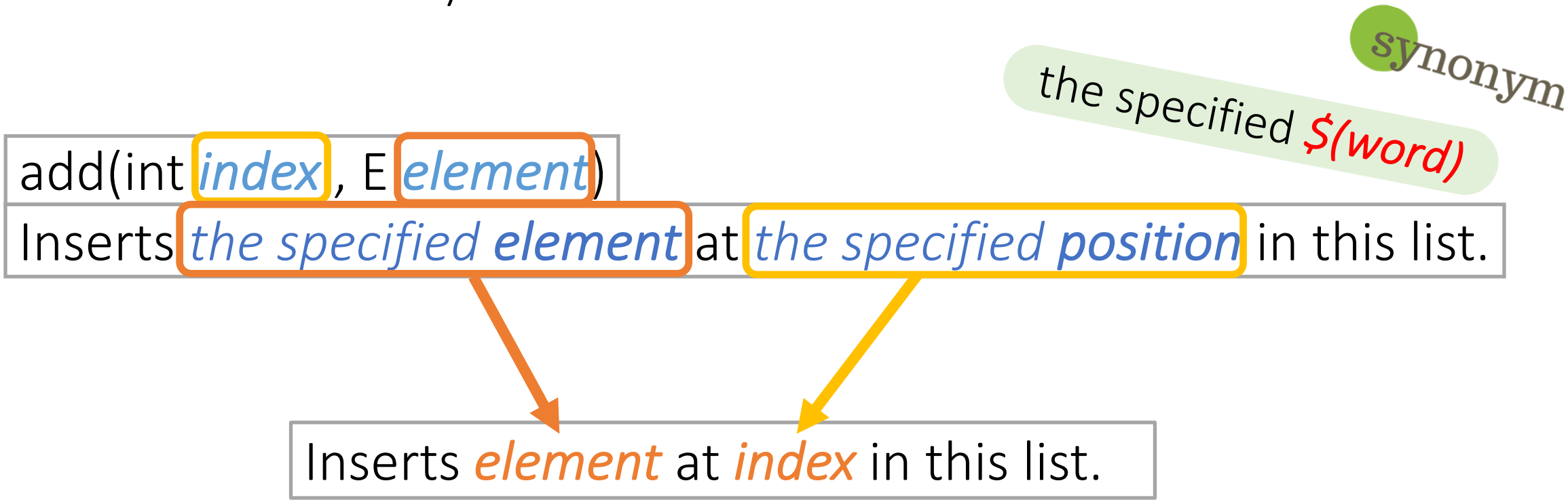
Design: Generators



Variables, Structures, Operations

Design: IR Generator -- Variables

- Internal variables: any names
- *Parameters*: identify from documents



Design: IR Generator -- Program structures

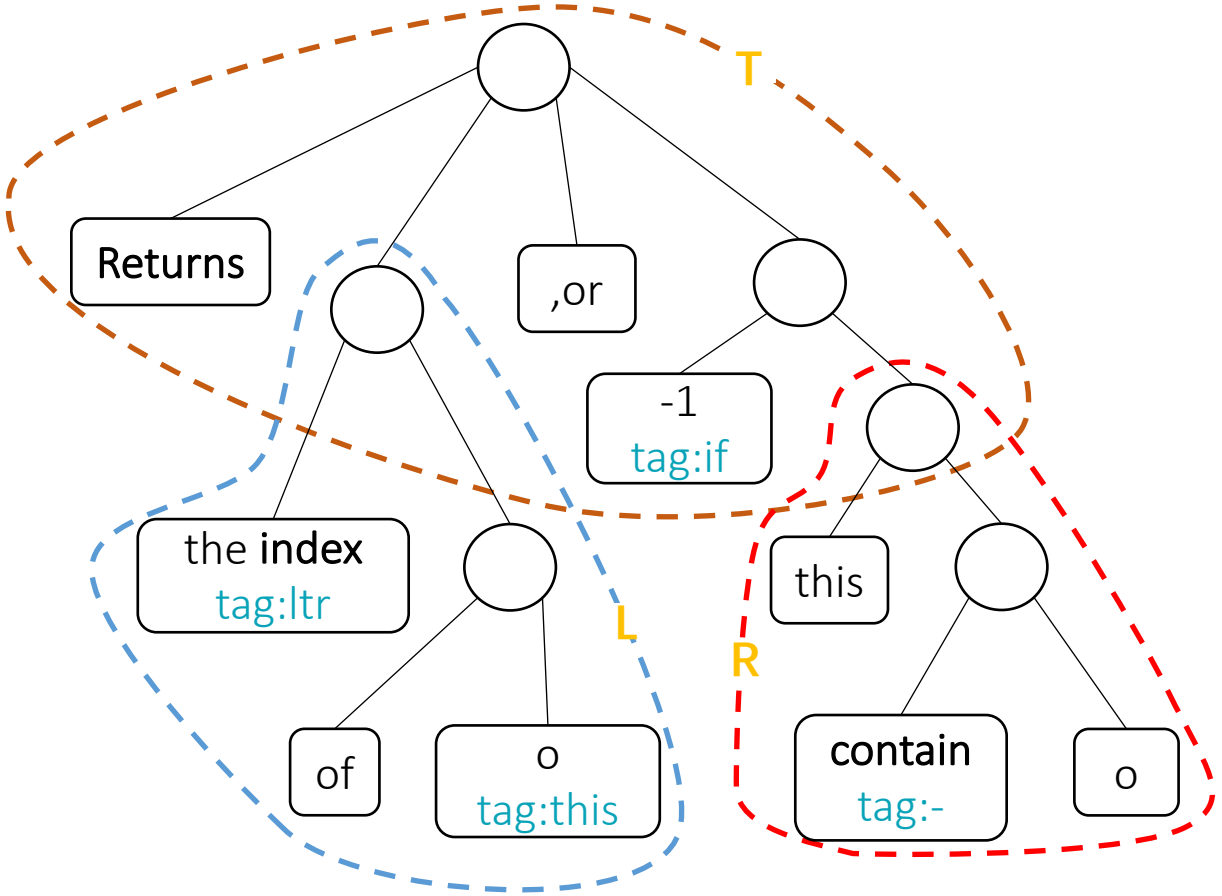
- Sequential
  - Default

- Loop structure
  - plurals
  - singular nouns modified by “each”
  - the first/last occurrence → indicate the loop iteration order

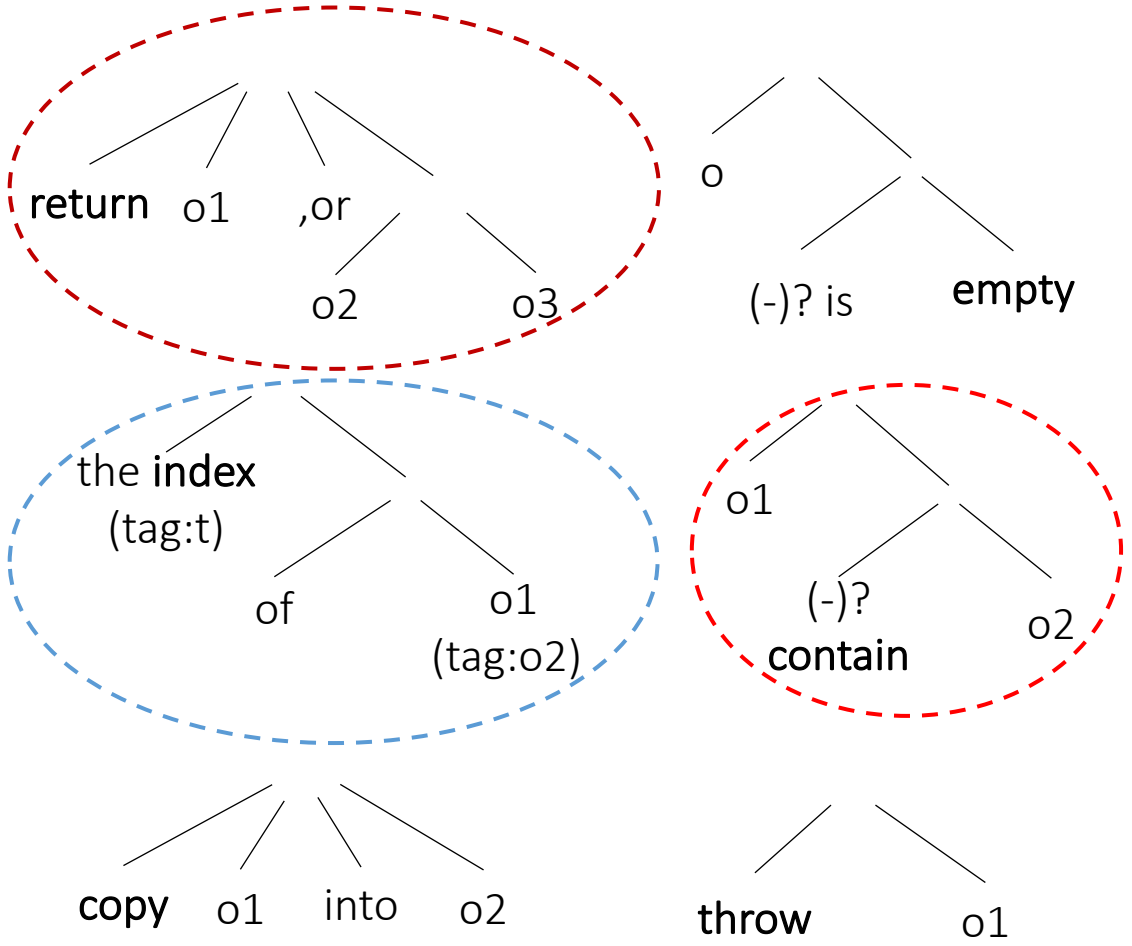
- Conditional structure
  - if/when
  - otherwise

Design: Model Generator

- Tile the IR (tree)

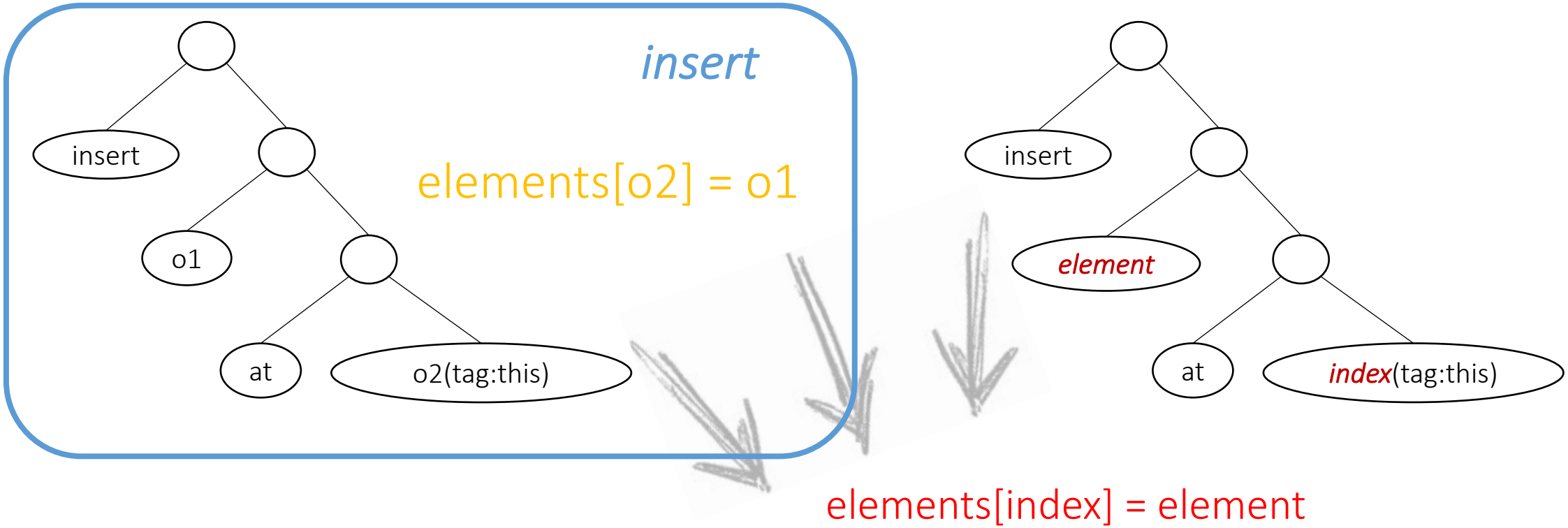


Tree Pattern



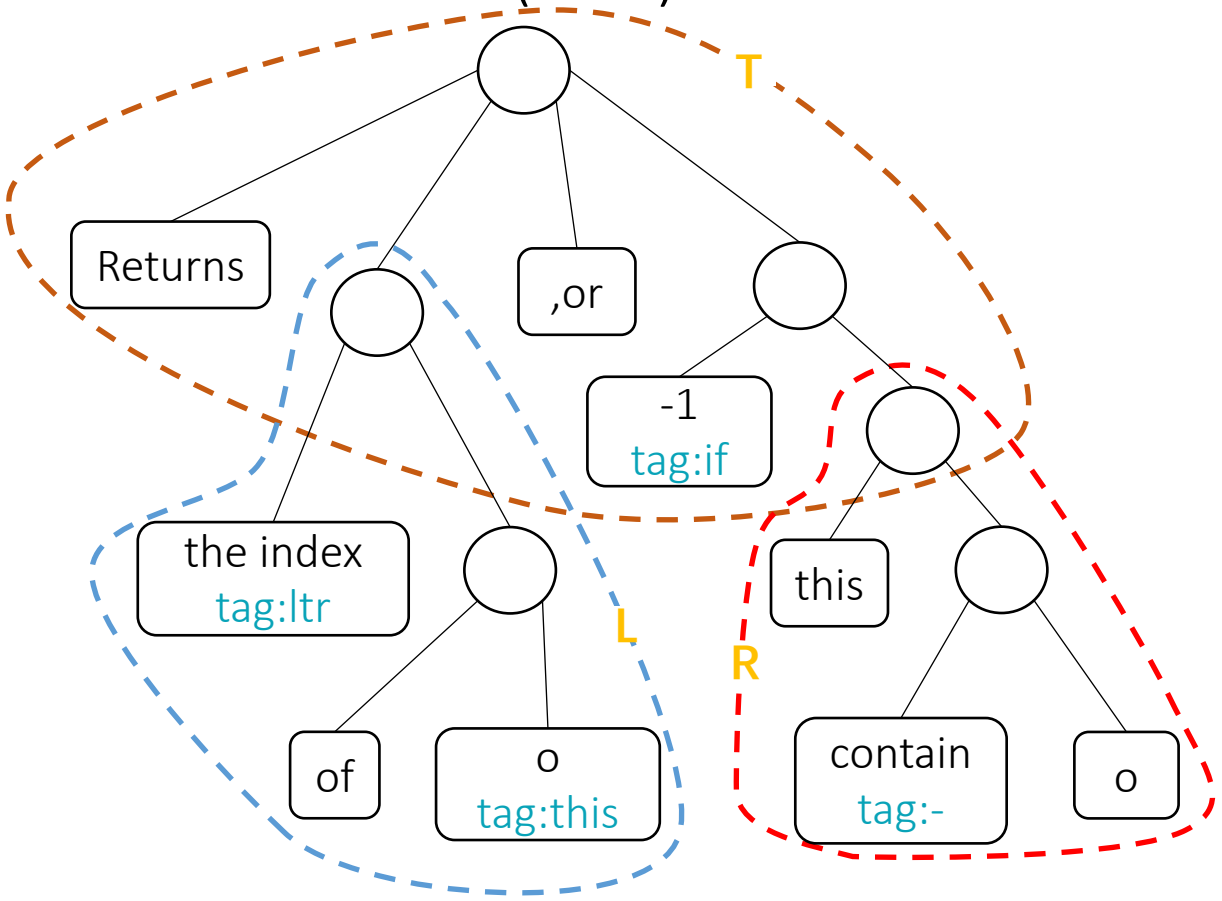
Design: Model Generator

- Unified data structure model: one-dimensional array
- Primitive: tree pattern corresponds to a piece of code template



Design: Model Generator

- Tile the IR (tree)




```
int index1 = -1;
if(o == null){
   for(int i=0; i<size; i++){
        if(element[i] == null){
            index1 = i;
            break;
        }
   }
}else{
  for(int i=0; i<size; i++){
       if(o.equals(element[i])){
            index1 = i;
            break;
       }
  }
}
```

L

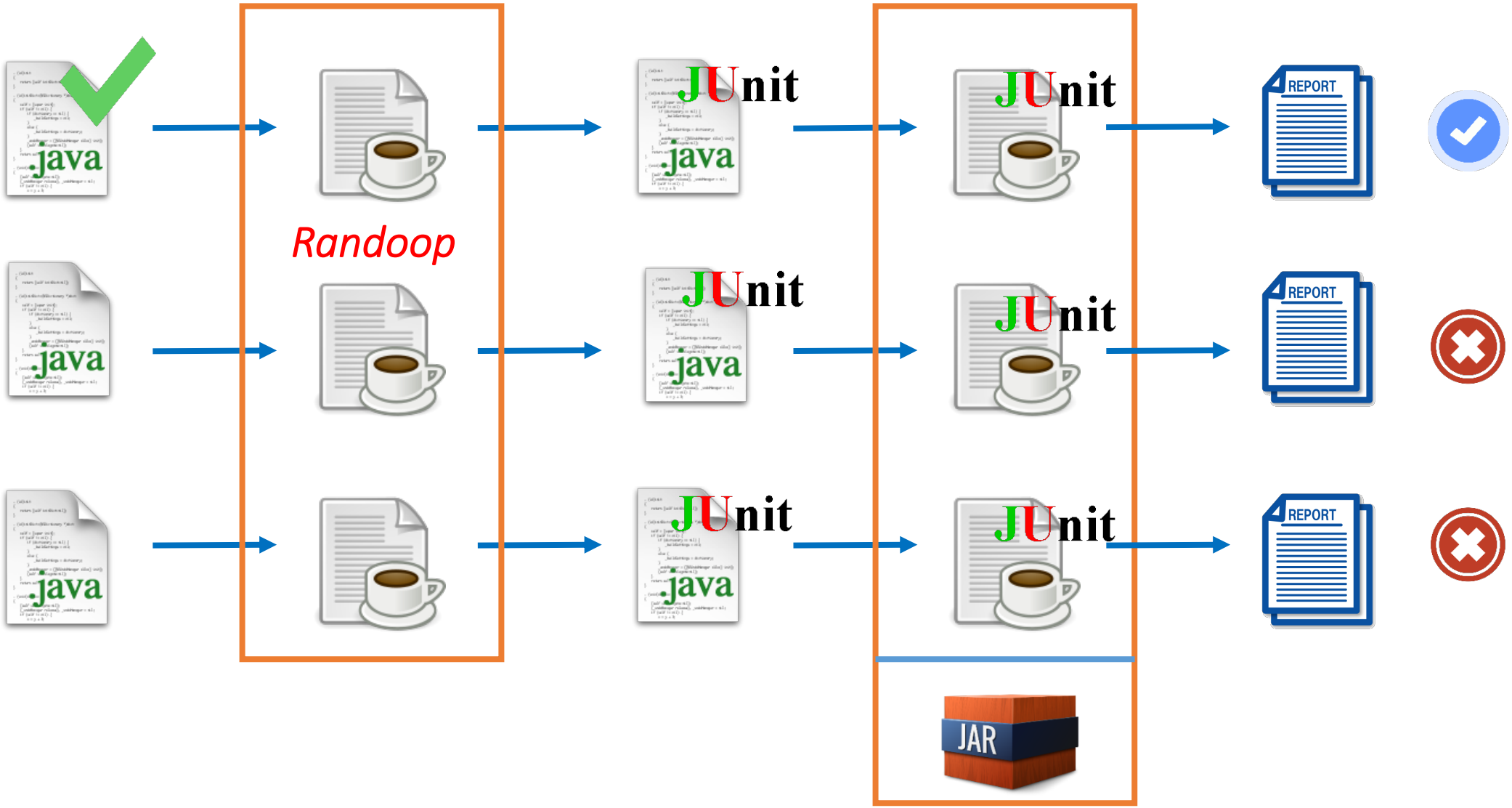
```
int index2 = -1;
if(o == null){
   for(int i=0; i<size; i++){
        if(element[i] == null){
            index2 = i;
            break;
        }
   }
}else{
  for(int i=0; i<size; i++){
       if(o.equals(element[i])){
            index2 = i;
            break;
       }
  }
}
```

R

```
if(index2 == -1) return -1;
else return index1;
```

T

Design: Model Validator



Evaluation Setup

- Hardware
  - CPU: Intel® i7-3770
  - RAM: 8GB
- Operating system
  - Ubuntu 12.04

Evaluation: Overall Result

Class	# Total Methods	# Modeled Methods	%
ArrayList	34	29	85.29%
Vector	54	46	85.19%
Stack	6	5	83.33%
ArrayDeque	36	35	97.22%
LinkedList	42	41	97.62%
HashMap	28	23	82.14%
LinkedHashMap	15	14	93.33%
HashSet	13	12	92.31%
LinkedHashSet	5	4	80.00%
AttributeList	15	11	73.33%
RoleList	14	9	64.29%
RoleUnresolvedList	14	9	64.29%
StringBuffer	54	40	74.07%
StringBuilder	54	40	74.07%
Summary	397	326	82.12%

Evaluation: Cases that Cannot be Handled

- Incompleteness of API documents
  - *add(int index, Object element)* in *AttributeList*
    lack of descriptions about the *IndexOutOfBoundsException*

- Describe one primitive behavior with several sentences
  - *insert(int index, Char[] str, int offset, int len)* in *StringBuffer*
    Inserts the string representation of a subarray of the str array argument into this sequence.
    The subarray begins at the specified offset and extends len chars.
    The characters of the subarray are inserted into this sequence atthe position indicated by index

Evaluation: Static Taint Analysis

- Undesirable information flow
- Compare paths found by using *our model* V.S. *official JDK*
- Set up
  - Android: 96 apps
  - Sources: User input
  - Sinks: Internet, Log

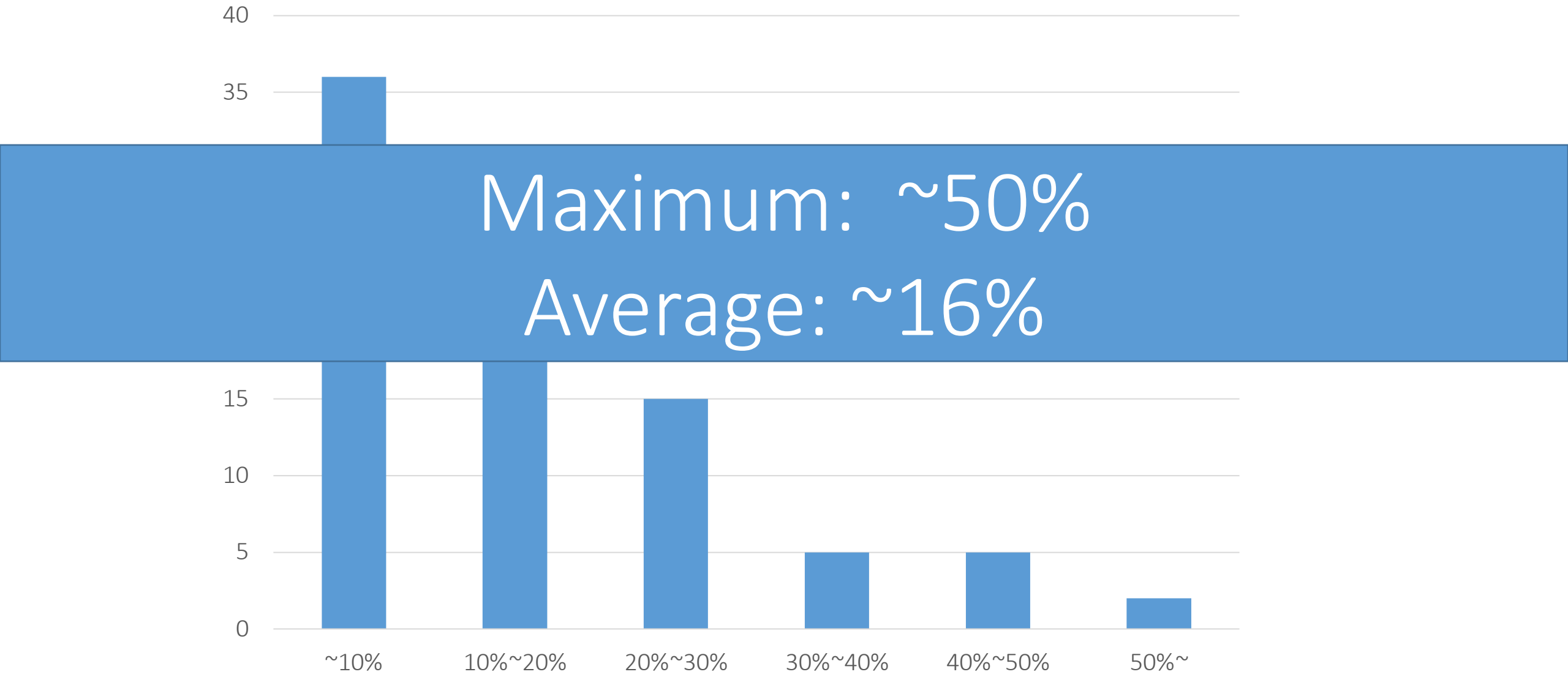
Evaluation: Static Taint Analysis

- Results
  - the same set of information leak warnings for both versions for almost all apps
  - except app com.yes123.mobile

- Case: *com.123yes.mobil*
  - Our model - 16 paths V.S. JDK – 14 paths
  - Java Native Interface (JNI) function call: toArray(object[]) -> System.arrayCopy()

Evaluation: Static Taint Analysis

- Efficiency improvement distribution



Maximum: ~50%

Average: ~16%

Related Work

- **Documentation Analysis**
  - Sarah[ICSE ‘16], Zhong [OOPSLA ’13, ASE ’09], Tan [ICST ’12, ICSE ’11, SOSP ’07] , Pandita [ICSE ’12], Sun [ICSE ’10], Sinha [ICST ’10], Runeson [ICSE ’07] , Henkel [TSE ’07]

- **Environment Modeling**
  - Jeon[ICSE ‘16], Merwe [SEN ’15], Ceccarello [SEN ’14], Palepu [ASE ’13], Qi [WCRE ’12], Cadar [OSDI ’08], Tkachuk [ASE ’03]

Conclusion

- Idea: modeling Java library from Java API documents
  - A combination of NLP and auto-testing
- Advantages
  - Expected behaviors with simpler code (no JNI code, no other languages etc.)
  - Helps many program analysis techniques

Thank You