Go Static: Contextualized Logging Statement Generation

Yichen Li, Yintong Huo, Renyi Zhong, Zhihan Jiang, Jinyang Liu, Junjie Huang,

Jiazhen Gu, Pinjia He, Michael R .Lyu

The Chinese University of Hong Kong

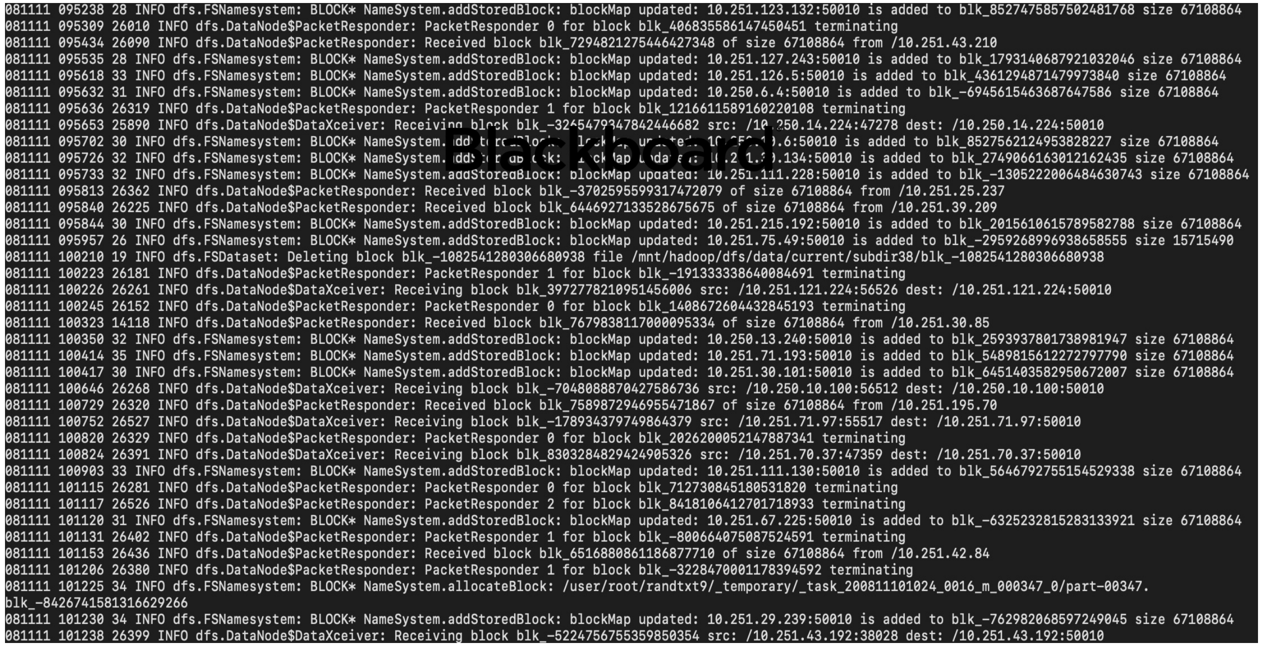
The Chinese University of Hong Kong (Shen Zhen)

ARISE Automated Reliable Intelligent Software Engineering

香港中文大學 The Chinese University of Hong Kong

Background & Motivation

Software systems produce logs to record runtime information

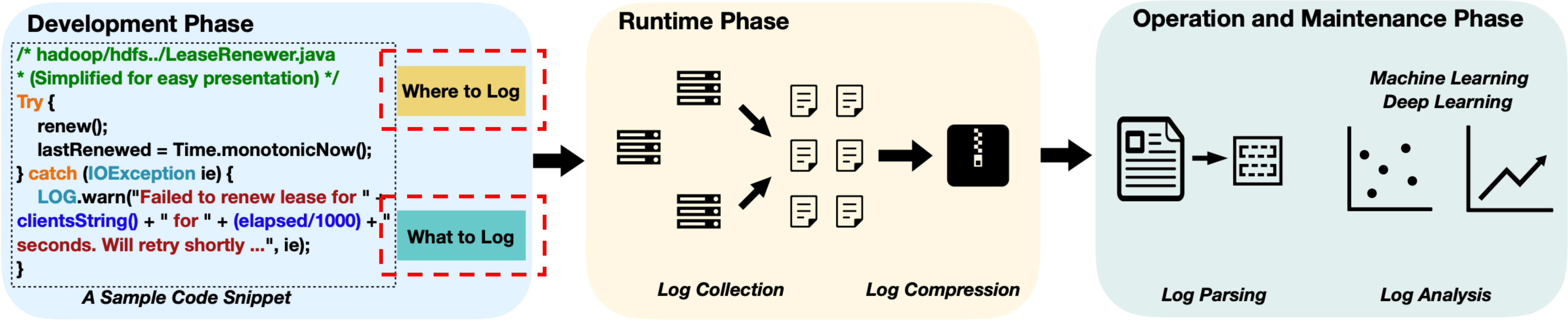


```
081111 095238 28 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.123.132:50010 is added to blk_8527475857502481768 size 67108864
081111 095309 26010 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_406835586147450451 terminating
081111 095434 26090 INFO dfs.DataNode$PacketResponder: Received block blk_7294821275446427348 of size 67108864 from /10.251.43.210
081111 095535 28 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.127.243:50010 is added to blk_1793140687921032046 size 67108864
081111 095618 33 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.126.5:50010 is added to blk_4361294871479973840 size 67108864
081111 095632 31 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.6.4:50010 is added to blk_-6945615463687647586 size 67108864
081111 095636 26319 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_1216611589160220108 terminating
081111 095653 25890 INFO dfs.DataNode$DataXceiver: Receiving block blk_-3265479347842446682 src: /10.250.14.224:47278 dest: /10.250.14.224:50010
081111 095702 30 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.10.6:50010 is added to blk_8527562124953828227 size 67108864
081111 095726 32 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.30.134:50010 is added to blk_2749066163012162435 size 67108864
081111 095733 32 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.111.228:50010 is added to blk_-1305222006484630743 size 67108864
081111 095813 26362 INFO dfs.DataNode$PacketResponder: Received block blk_-3702595599317472079 of size 67108864 from /10.251.25.237
081111 095840 26225 INFO dfs.DataNode$PacketResponder: Received block blk_6446927133528675675 of size 67108864 from /10.251.39.209
081111 095844 30 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.215.192:50010 is added to blk_2015610615789582788 size 67108864
081111 095957 26 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.75.49:50010 is added to blk_-2959268996938658555 size 15715490
081111 100210 19 INFO dfs.FSDataset: Deleting block blk_-1082541280306680938 file /mnt/hadoop/dfs/data/current/subdir38/blk_-1082541280306680938
081111 100223 26181 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_-19133338640084691 terminating
081111 100226 26261 INFO dfs.DataNode$DataXceiver: Receiving block blk_3972778210951456006 src: /10.251.121.224:56526 dest: /10.251.121.224:50010
081111 100245 26152 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_1408672604432845193 terminating
081111 100323 14118 INFO dfs.DataNode$PacketResponder: Received block blk_7679838117000095334 of size 67108864 from /10.251.30.85
081111 100350 32 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.13.240:50010 is added to blk_2593937801738981947 size 67108864
081111 100414 35 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.71.193:50010 is added to blk_5489815612272797790 size 67108864
081111 100417 30 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.30.101:50010 is added to blk_6451403582950672007 size 67108864
081111 100646 26268 INFO dfs.DataNode$DataXceiver: Receiving block blk_-7048088870427586736 src: /10.250.10.100:56512 dest: /10.250.10.100:50010
081111 100729 26320 INFO dfs.DataNode$PacketResponder: Received block blk_7589872946955471867 of size 67108864 from /10.251.195.70
081111 100752 26527 INFO dfs.DataNode$DataXceiver: Receiving block blk_-178934379749864379 src: /10.251.71.97:55517 dest: /10.251.71.97:50010
081111 100820 26329 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_2026200052147887341 terminating
081111 100824 26391 INFO dfs.DataNode$DataXceiver: Receiving block blk_8303284829424905326 src: /10.251.70.37:47359 dest: /10.251.70.37:50010
081111 100903 33 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.111.130:50010 is added to blk_5646792755154529338 size 67108864
081111 101115 26281 INFO dfs.DataNode$PacketResponder: PacketResponder 0 for block blk_712730845180531820 terminating
081111 101117 26526 INFO dfs.DataNode$PacketResponder: PacketResponder 2 for block blk_8418106412701718933 terminating
081111 101120 31 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.67.225:50010 is added to blk_-6325232815283133921 size 67108864
081111 101131 26402 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_-800664075087524591 terminating
081111 101153 26436 INFO dfs.DataNode$PacketResponder: Received block blk_6516880861186877710 of size 67108864 from /10.251.42.84
081111 101206 26380 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_-3228470001178394592 terminating
081111 101225 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.allocateBlock: /user/root/randtxt9/_temporary/_task_200811101024_0016_m_000347_0/part-00347.
blk_-8426741581316629266
081111 101230 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.251.29.239:50010 is added to blk_-762982068597249045 size 67108864
081111 101238 26399 INFO dfs.DataNode$DataXceiver: Receiving block blk_-5224756755359850354 src: /10.251.43.192:38028 dest: /10.251.43.192:50010
```

Blackboard

Background & Motivation

Intelligent solutions for log analysis are now data-driven



Development Phase

```
/* hadoop/hdfs../LeaseRenewer.java
* (Simplified for easy presentation) */
Try {
   renew();
   lastRenewed = Time.monotonicNow();
} catch (IOException ie) {
   LOG.warn("Failed to renew lease for " +
clientsString() + " for " + (elapsed/1000) + "
seconds. Will retry shortly ...", ie);
}
```

A Sample Code Snippet

Framework of Modern Intelligent Log Analytics

Background & Motivation

Logging is critical:

```
if (current_value > max) {
  LOG.info("Update max value to: " + max);
  max = current_value;
}
```

Background & Motivation

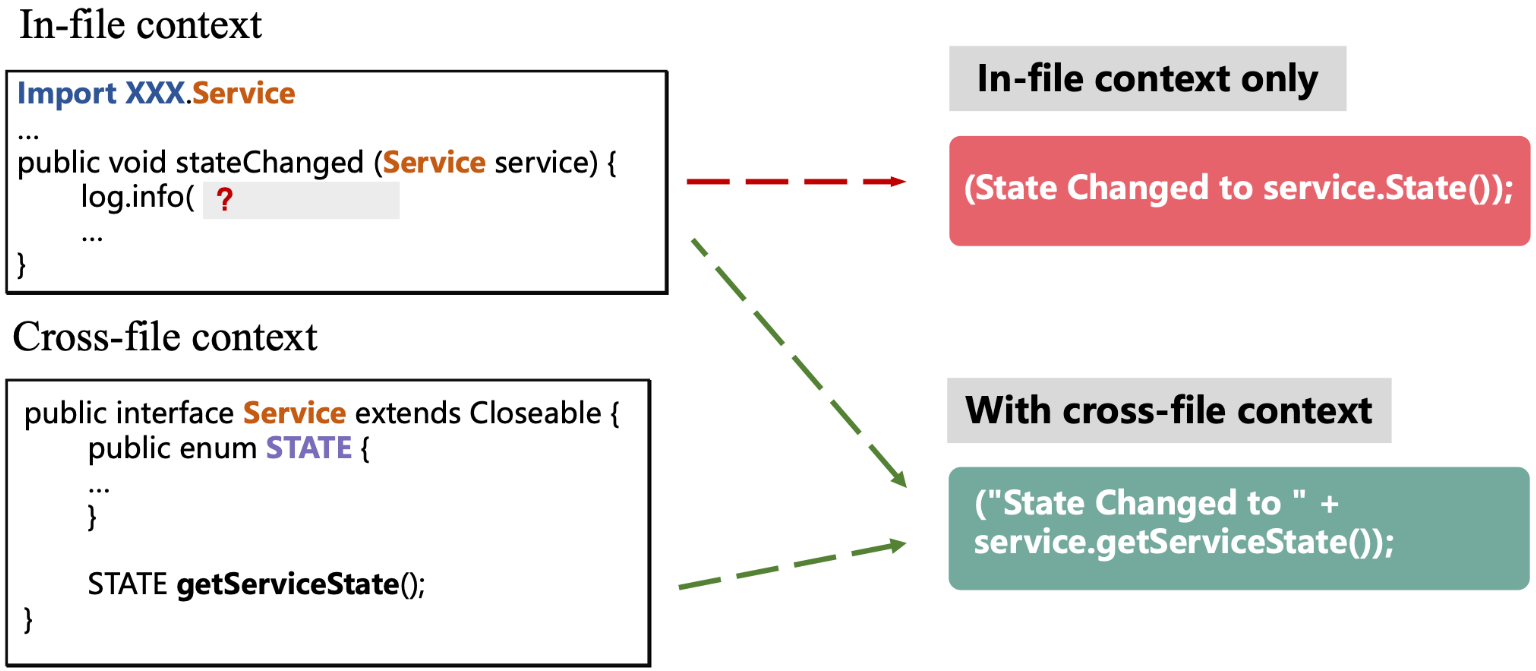
Writing logs is hard:

- In the era of agile development, developers prioritize functional code.
- Dev
- It is hard to find the balance between informativeness and overhead

How to **automatically** generate **appropriate** logging statements

Limitations of Current Inter-Method Approaches

Limited of current methods: single-method context knowledge



In-file context

```
Import XXX.Service
...
public void stateChanged (Service service) {
        log.info(  ?
        ...
}
```

Cross-file context

```
public interface Service extends Closeable {
        public enum STATE {
        ...
        }

        STATE getServiceState();
}
```

In-file context only

```
(State Changed to service.State());
```

With cross-file context

```
("State Changed to " +
service.getServiceState());
```

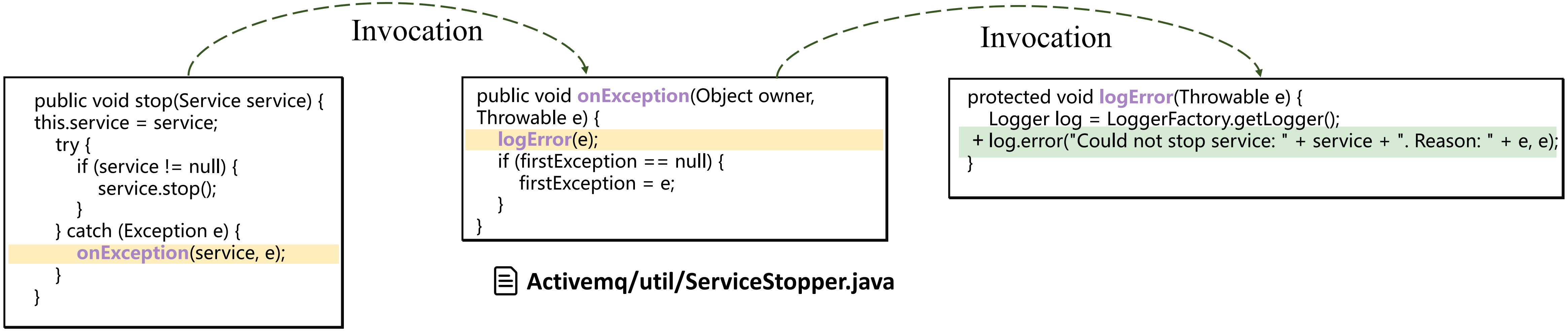
What is the ”service”?

How to use ”service”?

Limitations of Current Inter-Method Approaches

Limited of current methods: single-method context knowledge

- Limited static scope
- Missing type information
- Inconsistent logging style



```
public void stop(Service service) {
this.service = service;
    try {
        if (service != null) {
            service.stop();
        }
    } catch (Exception e) {
        onException(service, e);
    }
}
```

```
public void onException(Object owner,
Throwable e) {
    logError(e);
    if (firstException == null) {
        firstException = e;
    }
}
```

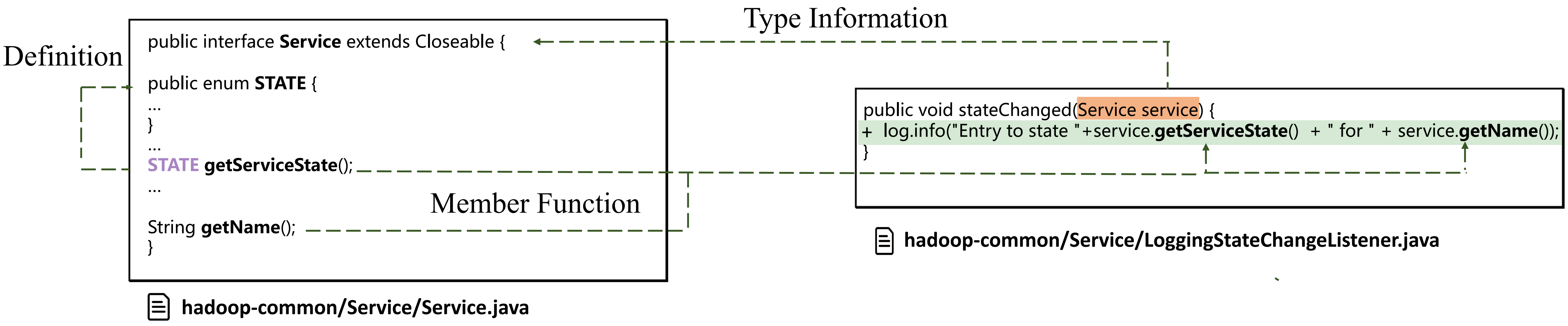
```
protected void logError(Throwable e) {
    Logger log = LoggerFactory.getLogger();
 + log.error("Could not stop service: " + service + ". Reason: " + e, e);
}
```

Activemq/util/ServiceStopper.java

Limitations of Current Inter-Method Approaches

Limited of current methods: single-method context knowledge

- Limited static scope
- Missing type information
- Inconsistent logging style



```
public interface Service extends Closeable {

public enum STATE {
...
}
...
STATE getServiceState();
...

String getName();
}
```

hadoop-common/Service/Service.java

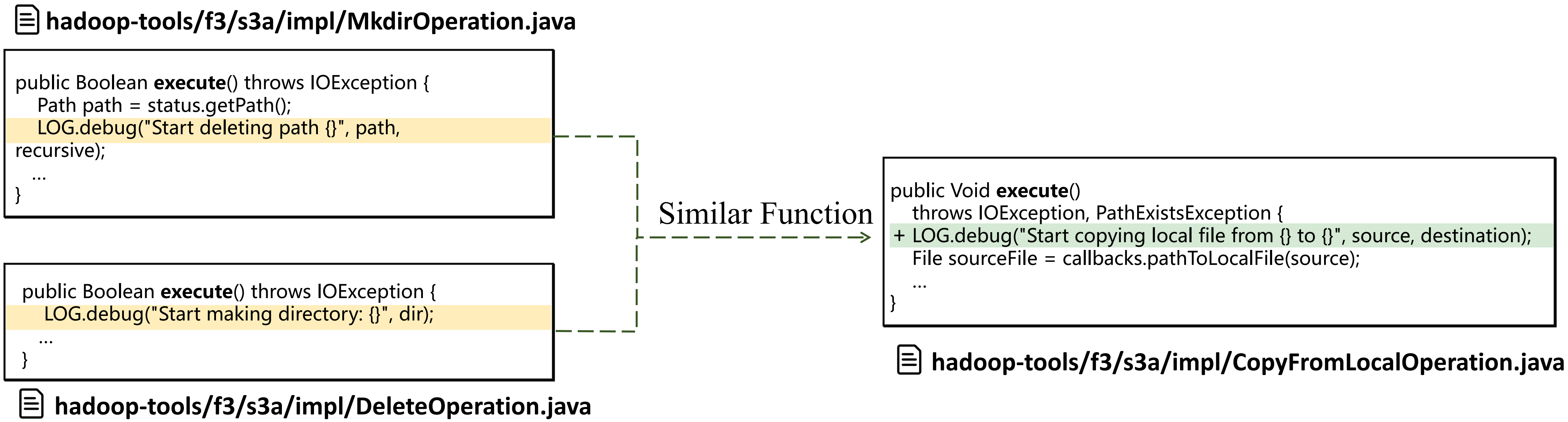
```
public void stateChanged(Service service) {
+  log.info("Entry to state "+service.getServiceState()  + " for " + service.getName());
}
```

hadoop-common/Service/LoggingStateChangeListener.java

Limitations of Current Inter-Method Approaches

Limited of current methods: single-method context knowledge

- Limited static scope
- Missing type information
- Inconsistent logging style



hadoop-tools/f3/s3a/impl/MkdirOperation.java

```
public Boolean execute() throws IOException {
    Path path = status.getPath();
    LOG.debug("Start deleting path {}", path,
recursive);
    ...
}
```

```
public Boolean execute() throws IOException {
    LOG.debug("Start making directory: {}", dir);
    ...
}
```

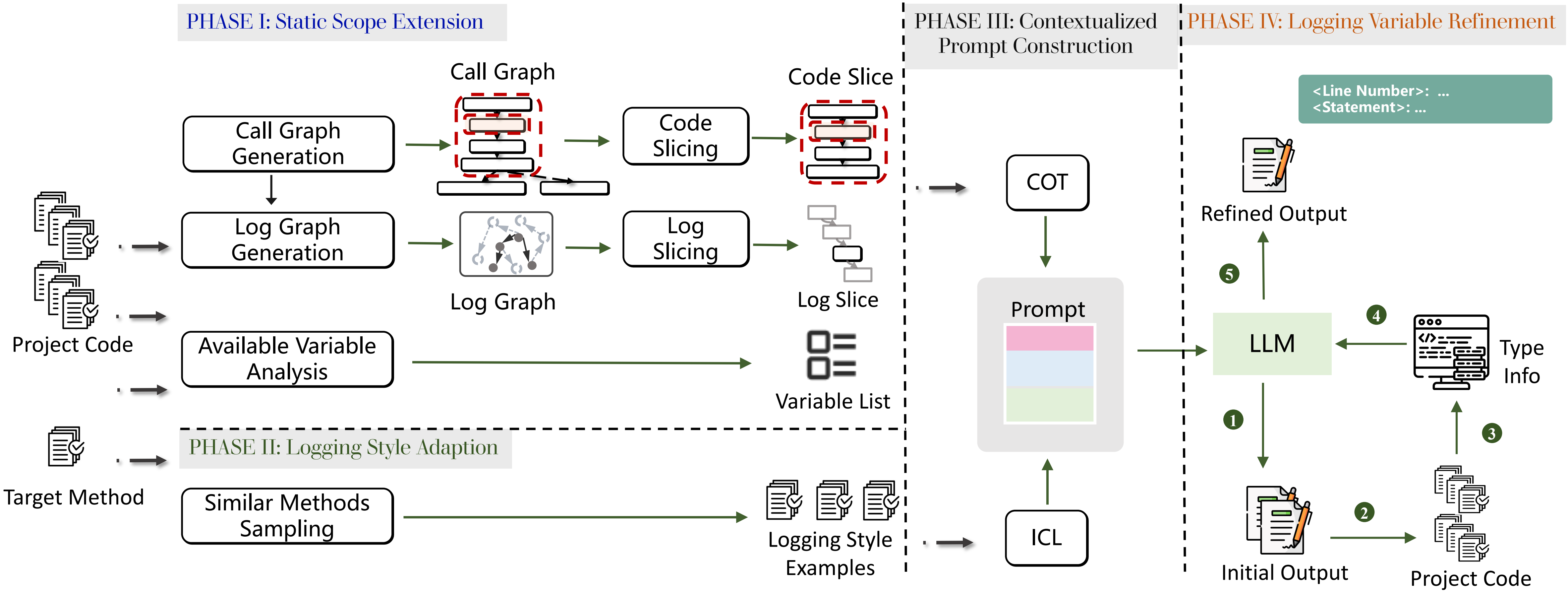
hadoop-tools/f3/s3a/impl/DeleteOperation.java

```
public Void execute()
    throws IOException, PathExistsException {
+   LOG.debug("Start copying local file from {} to {}", source, destination);
    File sourceFile = callbacks.pathToLocalFile(source);
    ...
}
```

hadoop-tools/f3/s3a/impl/CopyFromLocalOperation.java

Our proposal: SCLogger

SCLogger : a logging statement generation approach powered by inter-method Static Contexts

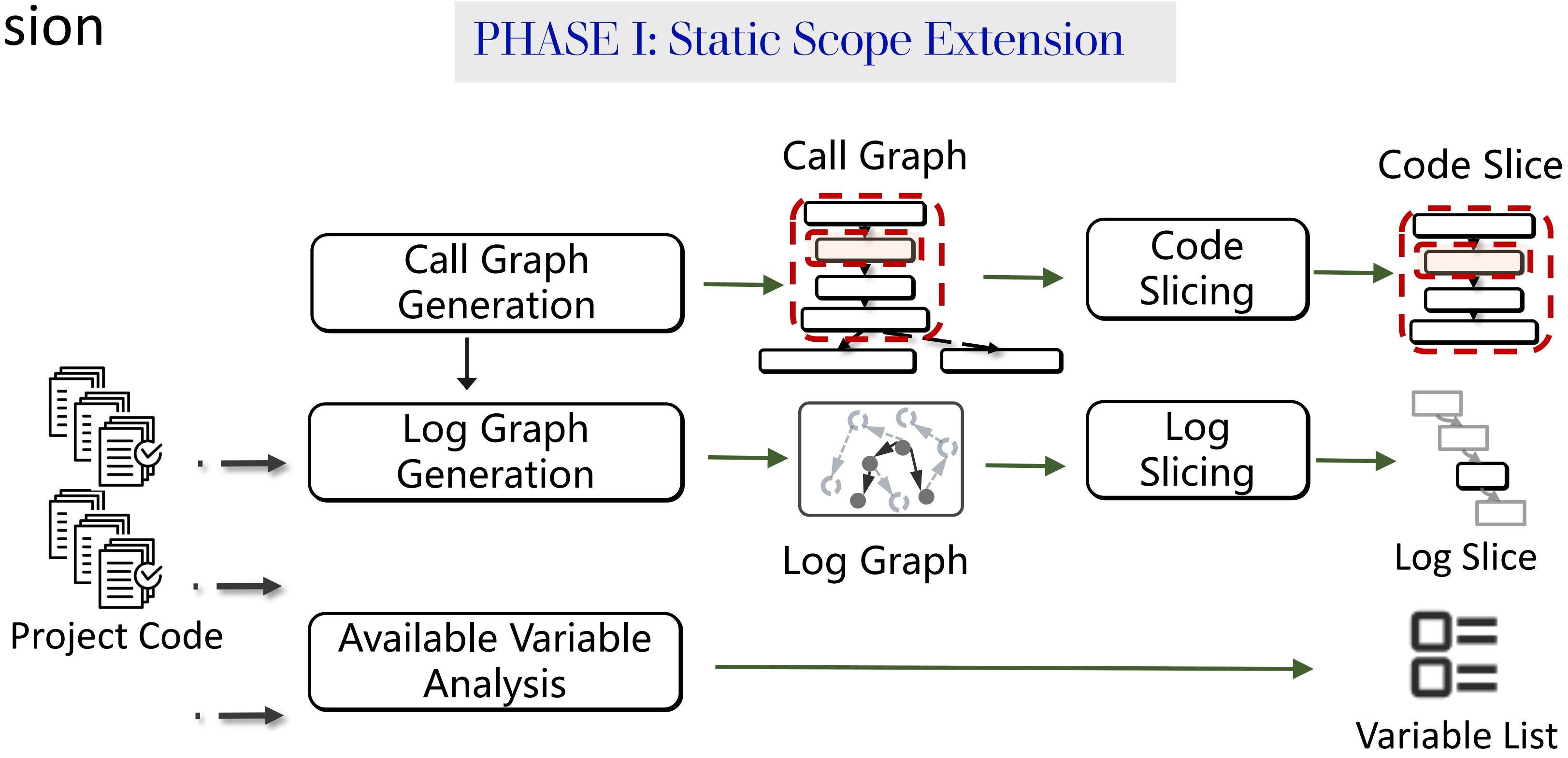


SCLogger: Phase 1

Limitation 1: Limited static scope

Knowledge 1: Static Scope Extension

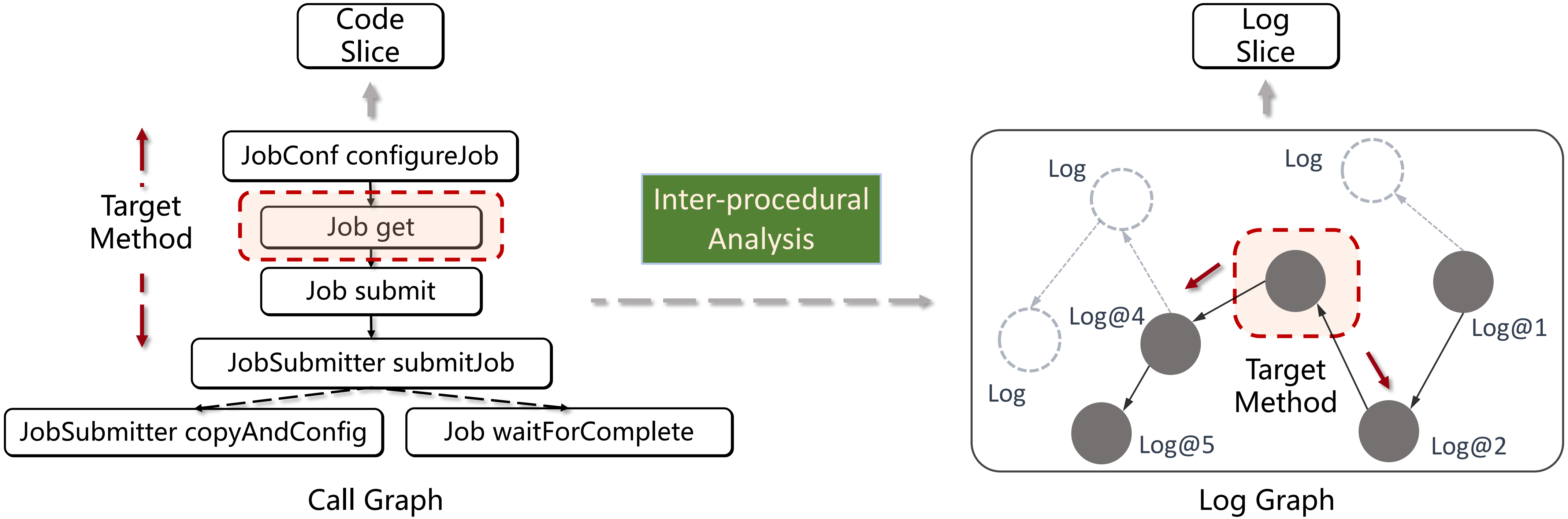
- Code Slicing
- Log Slicing
- Available Variable Analysis



SCLogger: Phase 1

From **Code Slicing** to **Log Slicing**

PHASE I: Static Scope Extension



Code Slice

JobConf configureJob

Job get

Job submit

JobSubmitter submitJob

JobSubmitter copyAndConfig

Job waitForComplete

Target Method

Call Graph

Inter-procedural Analysis

Log Slice

Log

Log

Log

Log@4

Log@1

Log@5

Log@2

Target Method

Log Graph

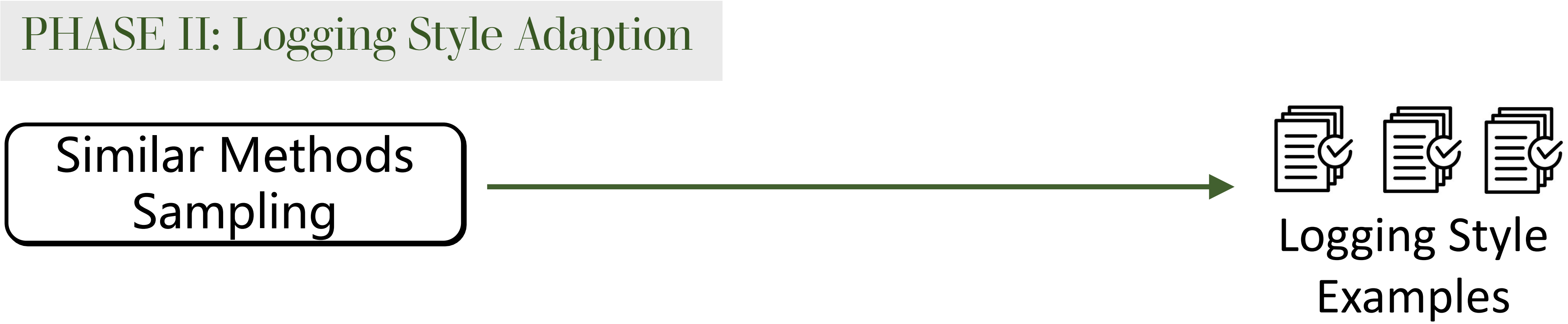
SCLogger: Phase 2

Limitation 2: Inconsistent logging style

Knowledge 2: Logging Style Adaption

- In-context Learning

Finally, keep consistent log style within current project, here are logging examples from current project:
<example>:<label>
<example>:<label>
…



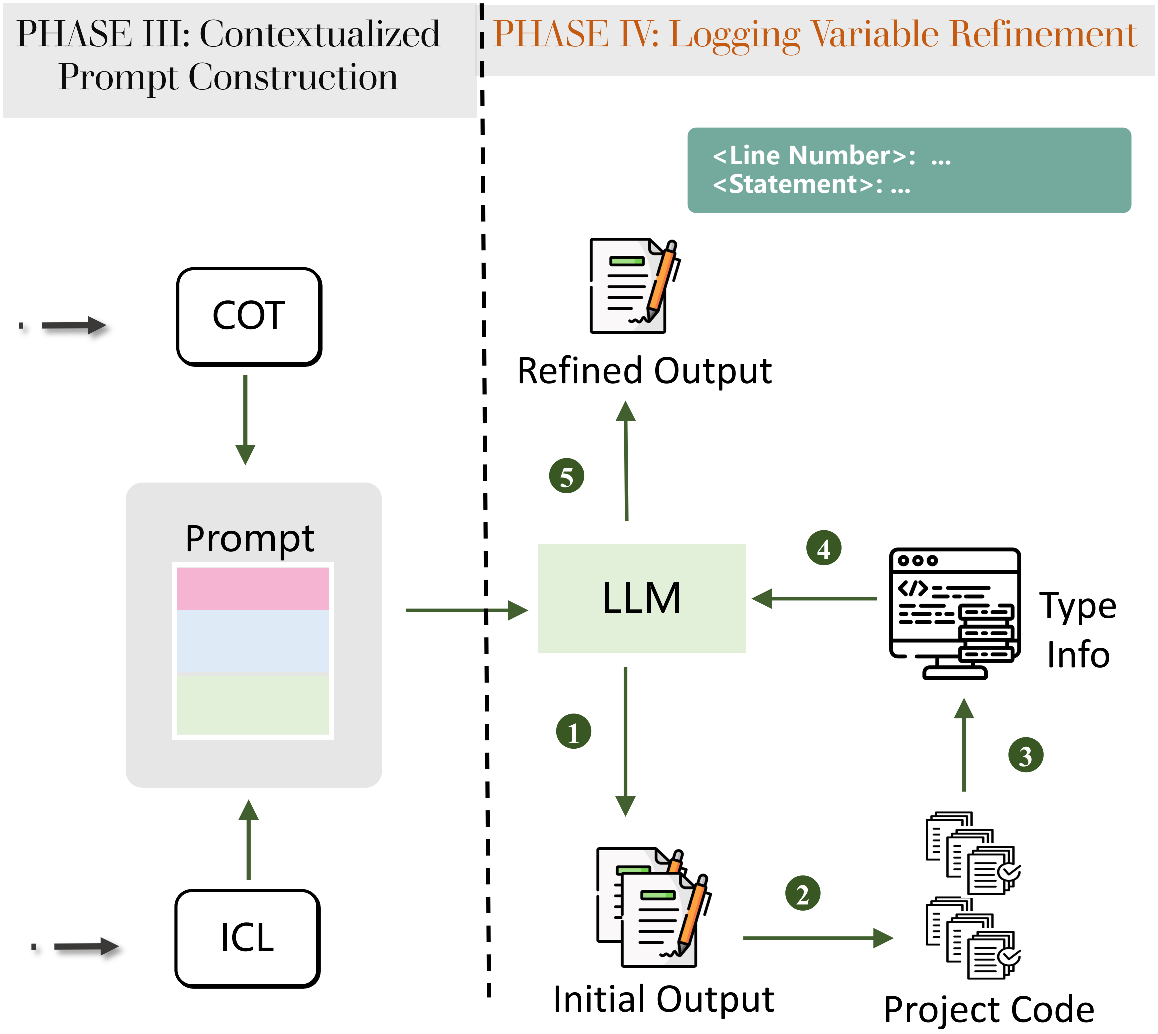
SCLogger: Phase 3, Phase 4

Limitation 3: Missing type information

Knowledge 3: Logging Variable Refinement

Chain of Thought (COT): Let’s think step by step, First you need to...

Logging Variable Refinement: variable type solving



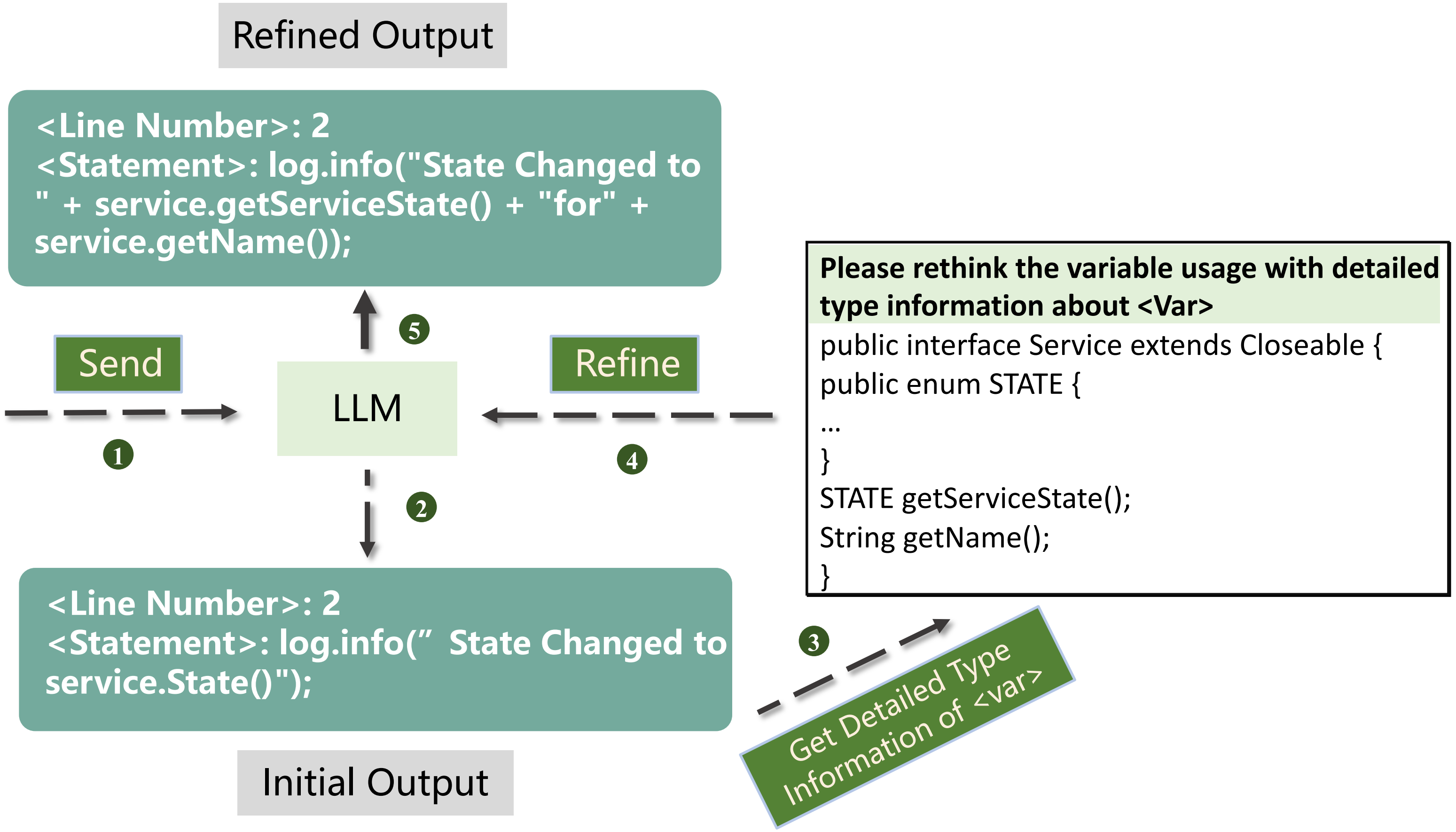
SCLogger: Example Prompt

Example Prompt

Instruction:
Insert a logging statement to the following code using <API>.
With the format <Line Number>: # <Statement>: #

Java Code:
public void stateChanged(Service service) {
…
}

Let's think step by step. First, the given method was called by <method A>, the code of called <methodB> is..
Second, the succeeding and proceeding logs are: <log> <log>…
Third, available variables of this method are <var>, <var>..
Finally, keep consistent log style within current project, here are logging examples from current project:
<example>:<label>
<example>:<label>
…



<Line Number>: 2
<Statement>: log.info("State Changed to " + service.getServiceState() + "for" + service.getName());

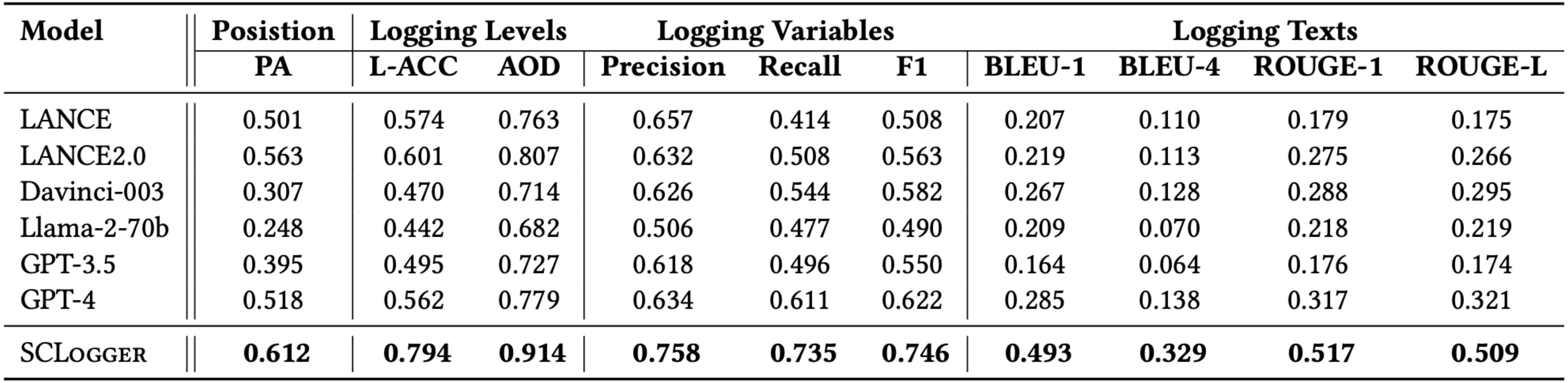
Please rethink the variable usage with detailed type information about <Var>
public interface Service extends Closeable {
public enum STATE {
…
}
STATE getServiceState();
String getName();
}

<Line Number>: 2
<Statement>: log.info(" State Changed to service.State()");

Performance of SCLogger

Both **Where to Log** and **What to Log**

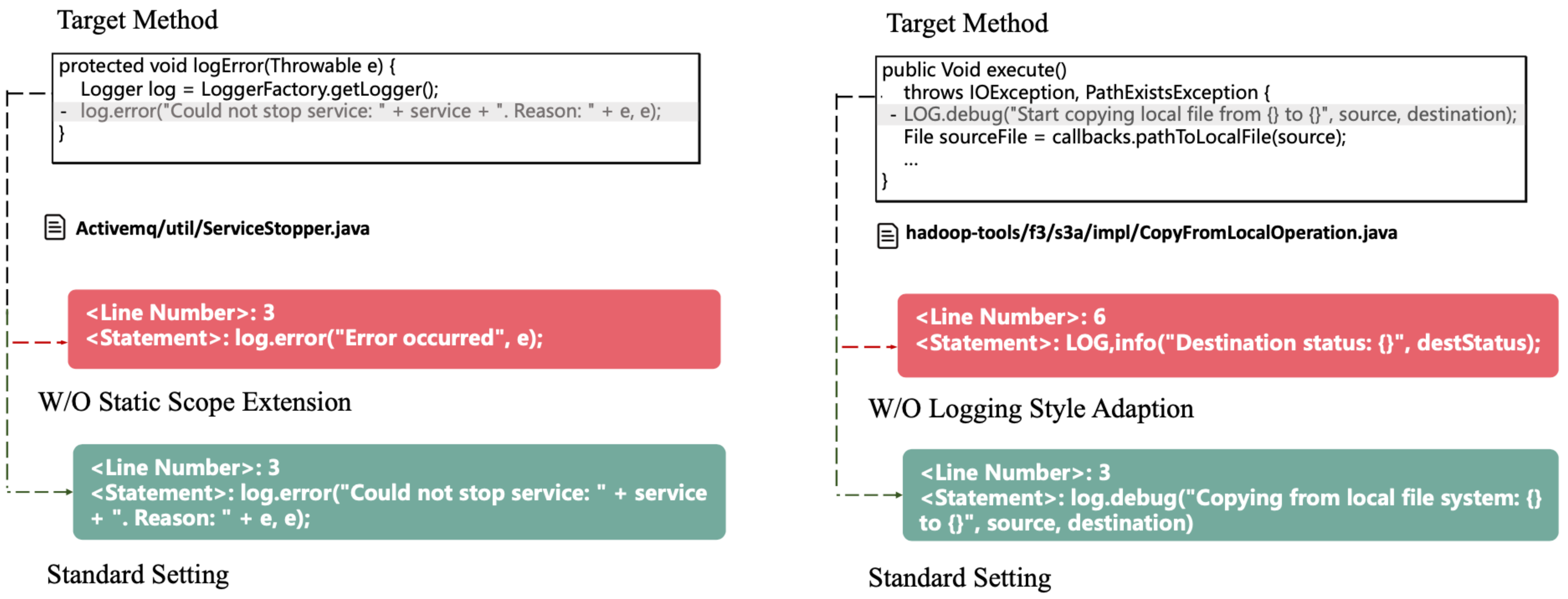
Table 2. Logging statements generation results from both *where-to-log* and *what-to-log* dimensions.



Model	Posistion	Logging Levels		Logging Variables			Logging Texts			
	PA	L-ACC	AOD	Precision	Recall	F1	BLEU-1	BLEU-4	ROUGE-1	ROUGE-L
LANCE	0.501	0.574	0.763	0.657	0.414	0.508	0.207	0.110	0.179	0.175
LANCE2.0	0.563	0.601	0.807	0.632	0.508	0.563	0.219	0.113	0.275	0.266
Davinci-003	0.307	0.470	0.714	0.626	0.544	0.582	0.267	0.128	0.288	0.295
Llama-2-70b	0.248	0.442	0.682	0.506	0.477	0.490	0.209	0.070	0.218	0.219
GPT-3.5	0.395	0.495	0.727	0.618	0.496	0.550	0.164	0.064	0.176	0.174
GPT-4	0.518	0.562	0.779	0.634	0.611	0.622	0.285	0.138	0.317	0.321
SCLogger	**0.612**	**0.794**	**0.914**	**0.758**	**0.735**	**0.746**	**0.493**	**0.329**	**0.517**	**0.509**

10% improvement in *where to log* and a 42% average enhancement across all metrics in *what to log*.

Ablation Study: Case Study



Target Method

```
protected void logError(Throwable e) {
    Logger log = LoggerFactory.getLogger();
-   log.error("Could not stop service: " + service + ". Reason: " + e, e);
}
```

Activemq/util/ServiceStopper.java

<Line Number>: 3
<Statement>: log.error("Error occurred", e);

W/O Static Scope Extension

<Line Number>: 3
<Statement>: log.error("Could not stop service: " + service + ". Reason: " + e, e);

Standard Setting

Target Method

```
public Void execute()
    throws IOException, PathExistsException {
-   LOG.debug("Start copying local file from {} to {}", source, destination);
    File sourceFile = callbacks.pathToLocalFile(source);
    ...
}
```

hadoop-tools/f3/s3a/impl/CopyFromLocalOperation.java

<Line Number>: 6
<Statement>: LOG,info("Destination status: {}", destStatus);

W/O Logging Style Adaption

<Line Number>: 3
<Statement>: log.debug("Copying from local file system: {} to {}", source, destination)

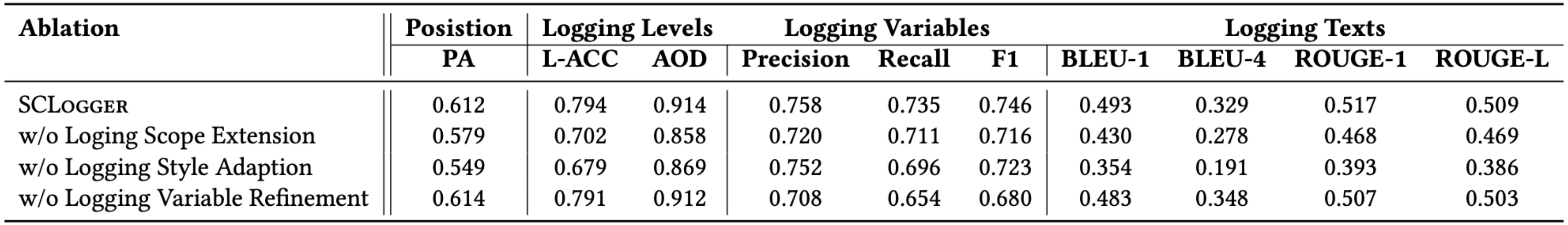
Standard Setting

(a) Removing the phase of static scope extension.

(b) Removing the phase of logging style adaption.

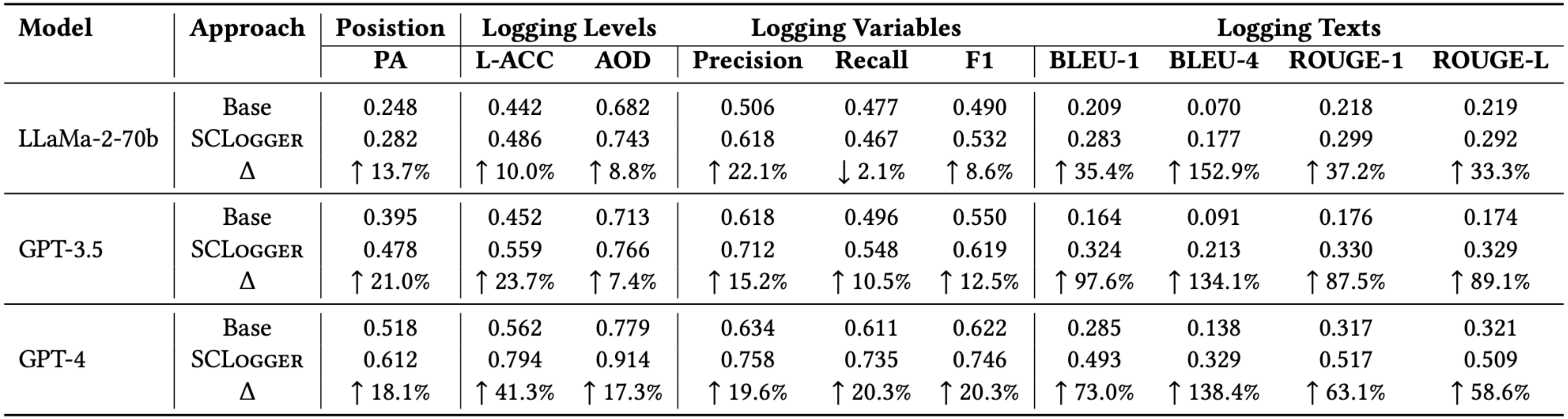
Ablation Study and Generality

Table 3. Ablation Study of SCLogger.



Ablation	Posistion	Logging Levels		Logging Variables			Logging Texts			
	PA	L-ACC	AOD	Precision	Recall	F1	BLEU-1	BLEU-4	ROUGE-1	ROUGE-L
SCLogger	0.612	0.794	0.914	0.758	0.735	0.746	0.493	0.329	0.517	0.509
w/o Loging Scope Extension	0.579	0.702	0.858	0.720	0.711	0.716	0.430	0.278	0.468	0.469
w/o Logging Style Adaption	0.549	0.679	0.869	0.752	0.696	0.723	0.354	0.191	0.393	0.386
w/o Logging Variable Refinement	0.614	0.791	0.912	0.708	0.654	0.680	0.483	0.348	0.507	0.503

Table 4. The performance of SCLogger with different backbone models.



Model	Approach	Posistion	Logging Levels		Logging Variables			Logging Texts			
		PA	L-ACC	AOD	Precision	Recall	F1	BLEU-1	BLEU-4	ROUGE-1	ROUGE-L
LLaMa-2-70b	Base	0.248	0.442	0.682	0.506	0.477	0.490	0.209	0.070	0.218	0.219
	SCLogger	0.282	0.486	0.743	0.618	0.467	0.532	0.283	0.177	0.299	0.292
	Δ	↑ 13.7%	↑ 10.0%	↑ 8.8%	↑ 22.1%	↓ 2.1%	↑ 8.6%	↑ 35.4%	↑ 152.9%	↑ 37.2%	↑ 33.3%
GPT-3.5	Base	0.395	0.452	0.713	0.618	0.496	0.550	0.164	0.091	0.176	0.174
	SCLogger	0.478	0.559	0.766	0.712	0.548	0.619	0.324	0.213	0.330	0.329
	Δ	↑ 21.0%	↑ 23.7%	↑ 7.4%	↑ 15.2%	↑ 10.5%	↑ 12.5%	↑ 97.6%	↑ 134.1%	↑ 87.5%	↑ 89.1%
GPT-4	Base	0.518	0.562	0.779	0.634	0.611	0.622	0.285	0.138	0.317	0.321
	SCLogger	0.612	0.794	0.914	0.758	0.735	0.746	0.493	0.329	0.517	0.509
	Δ	↑ 18.1%	↑ 41.3%	↑ 17.3%	↑ 19.6%	↑ 20.3%	↑ 20.3%	↑ 73.0%	↑ 138.4%	↑ 63.1%	↑ 58.6%

Practicality Discussion

Cost Reduction

SCLogger **only** extracts and isolates the context related to logging ✓

Taking all the cross-file contexts as input ✗

SCLogger **only** takes type information of chosen logging variables using two-staged Strategy ✓

Taking type information of all available variables from the initial input ✗

IDE Integration

SCLogger can be **easily integrated into** well-established Integrated Development Environments (IDEs)

Built-in static analysis capabilities of IDEs

- Invocation analysis
- Type inference
- Linting feedback
- ….

Conclusion

- Current approaches struggle to generate appropriate logging statement for complex project instances due to limited intra-method context.

- The proposed **SCLogger** analyzes inter-method static contexts for contextualized logging statement generation.

- SCLogger is effective and generalized, which provides insights for enhancing system runtime information.