Folding Repeated Instructions for Improving Token-based Code Clone Detection

○Hiroaki Murakami, Keisuke Hotta, Yoshiki Higo,
Hiroshi Igaki, Shinji Kusumoto

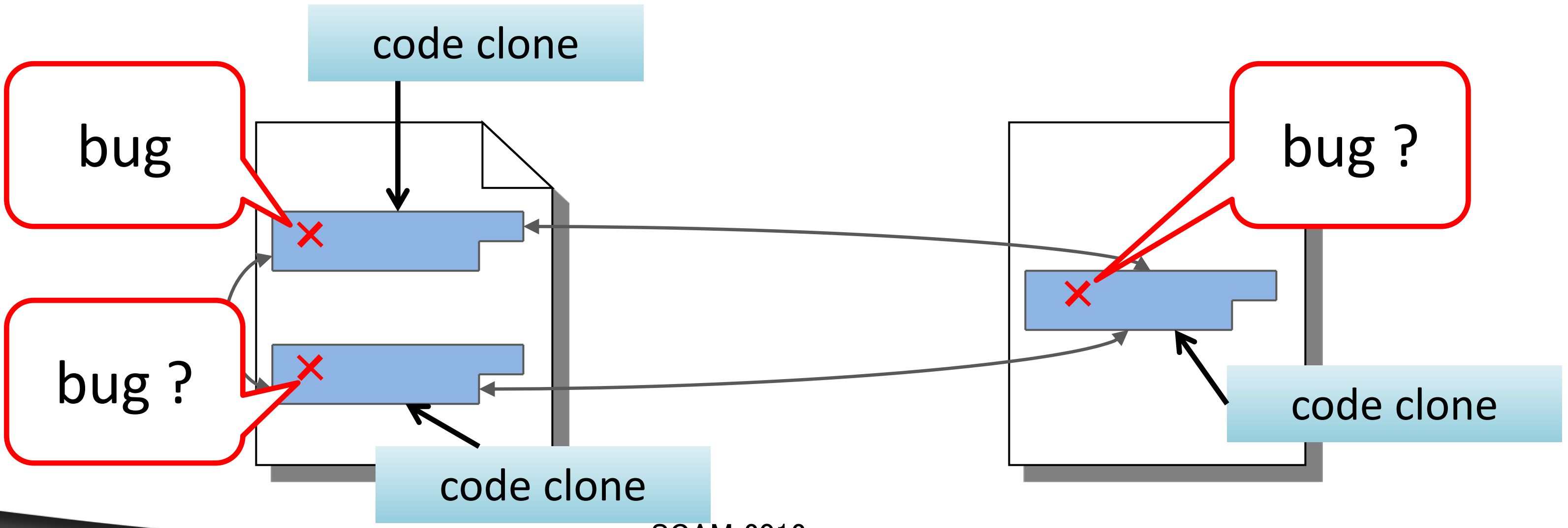
Graduate School of Information Science and Technology
Osaka University

Outline

- Background
- Problem of existing methods
- Proposed method
- Experiments
- Conclusion

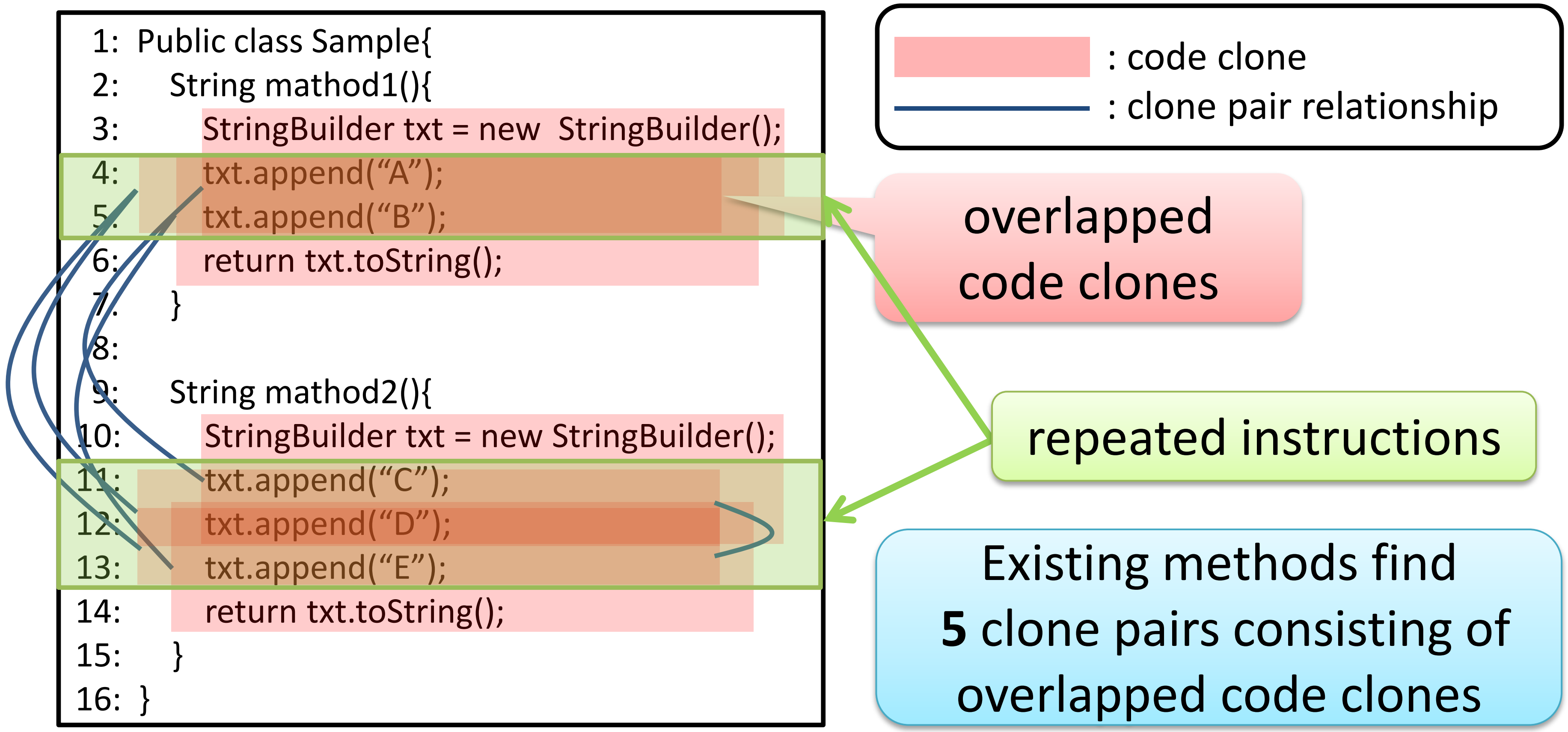
What is code clone ?

- A code clone is a code fragment in source files that is identical or similar to another.
- To detect code clones automatically, a variety of detection method has been proposed.



Overlapped code clones

- Line-based/Token-based detections report many overlapped code clones.



```
Public class Sample{
    String mathod1(){
        StringBuilder txt = new  StringBuilder();
        txt.append("A");
        txt.append("B");
        return txt.toString();
    }

    String mathod2(){
        StringBuilder txt = new StringBuilder();
        txt.append("C");
        txt.append("D");
        txt.append("E");
        return txt.toString();
    }
}
```

🟥 : code clone
— : clone pair relationship

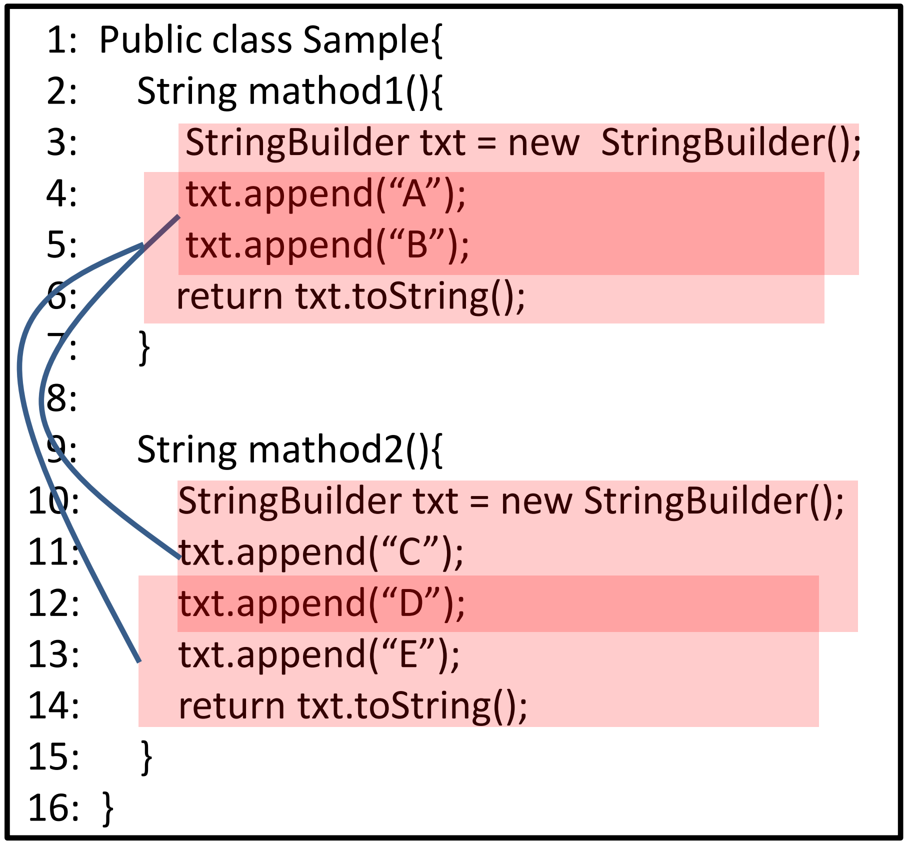
overlapped code clones

repeated instructions

Existing methods find **5** clone pairs consisting of overlapped code clones

Characteristics of overlapped code clones

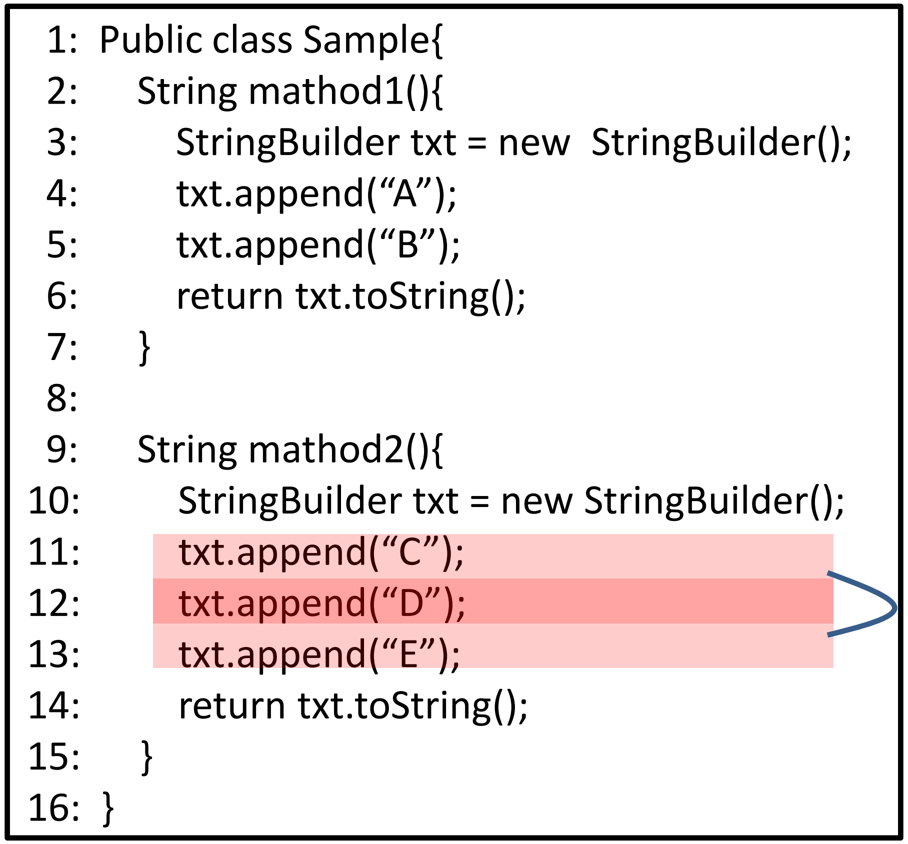
(1) almost the same location



```
Public class Sample{
    String mathod1(){
        StringBuilder txt = new  StringBuilder();
        txt.append(“A”);
        txt.append(“B”);
        return txt.toString();
    }

    String mathod2(){
        StringBuilder txt = new StringBuilder();
        txt.append(“C”);
        txt.append(“D”);
        txt.append(“E”);
        return txt.toString();
    }
}
```

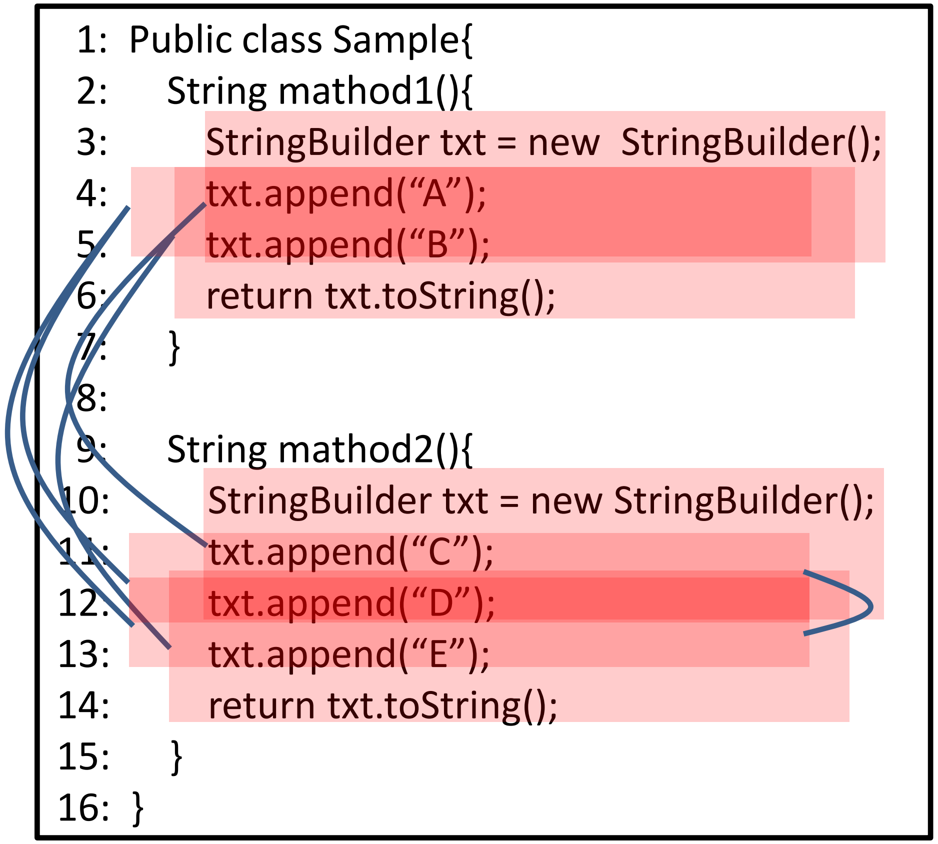
(2) self-overlapping



```
Public class Sample{
    String mathod1(){
        StringBuilder txt = new  StringBuilder();
        txt.append(“A”);
        txt.append(“B”);
        return txt.toString();
    }

    String mathod2(){
        StringBuilder txt = new StringBuilder();
        txt.append(“C”);
        txt.append(“D”);
        txt.append(“E”);
        return txt.toString();
    }
}
```

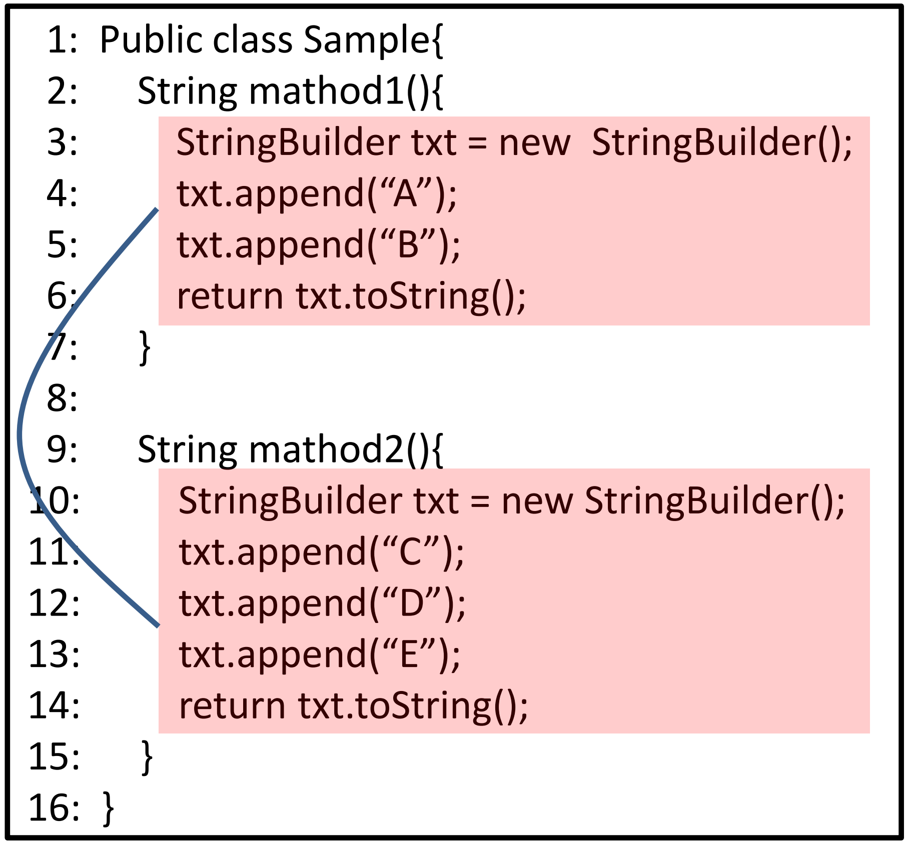
Desirable detection result



```
Public class Sample{
   String mathod1(){
      StringBuilder txt = new  StringBuilder();
      txt.append("A");
      txt.append("B");
      return txt.toString();
   }

   String mathod2(){
      StringBuilder txt = new StringBuilder();
      txt.append("C");
      txt.append("D");
      txt.append("E");
      return txt.toString();
   }
}
```

✖ 5 clone pairs

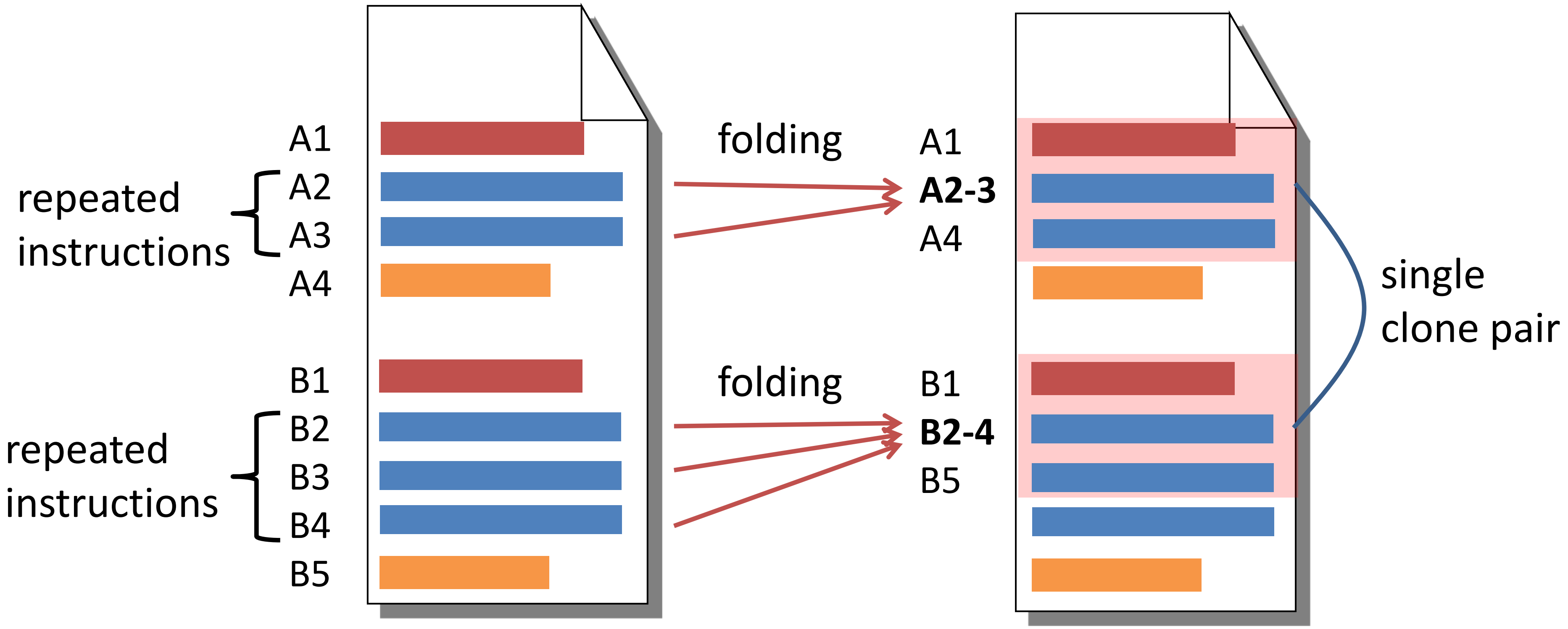


```
Public class Sample{
   String mathod1(){
      StringBuilder txt = new  StringBuilder();
      txt.append("A");
      txt.append("B");
      return txt.toString();
   }

   String mathod2(){
      StringBuilder txt = new StringBuilder();
      txt.append("C");
      txt.append("D");
      txt.append("E");
      return txt.toString();
   }
}
```

○ single clone pair

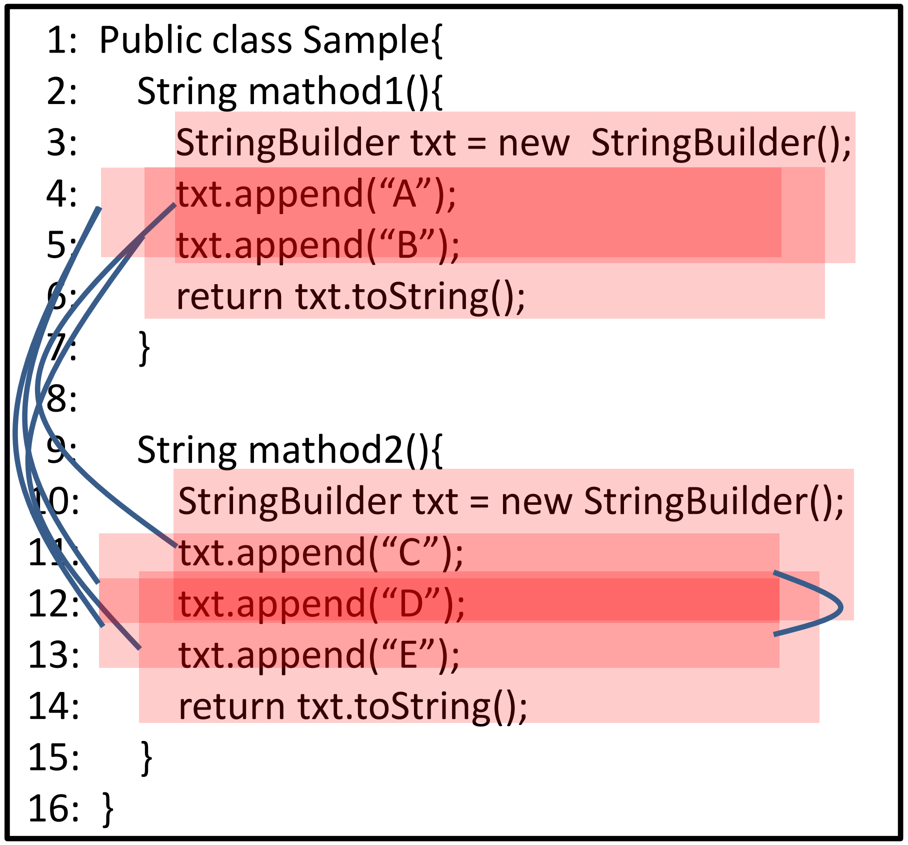
Proposed method



If repeated instructions are folded,
the problem of overlapped code clones can be solved.

Example detection of proposed method

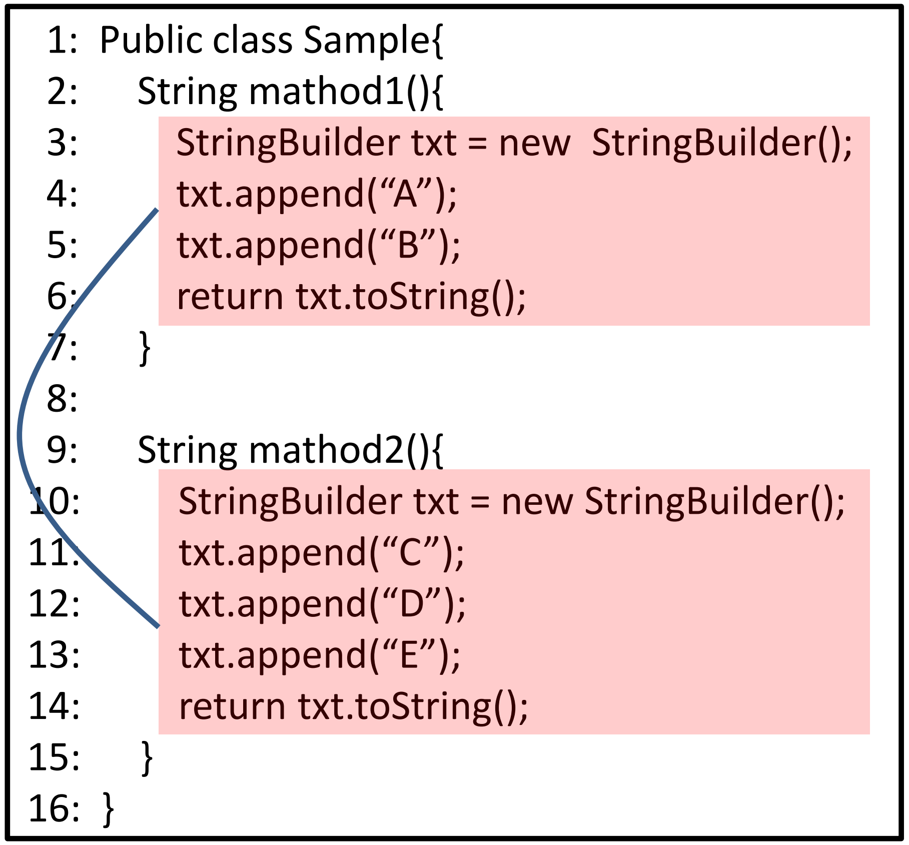
Result of existing method



```
Public class Sample{
    String mathod1(){
        StringBuilder txt = new  StringBuilder();
        txt.append("A");
        txt.append("B");
        return txt.toString();
    }

    String mathod2(){
        StringBuilder txt = new StringBuilder();
        txt.append("C");
        txt.append("D");
        txt.append("E");
        return txt.toString();
    }
}
```

Result of the proposed method



```
Public class Sample{
    String mathod1(){
        StringBuilder txt = new  StringBuilder();
        txt.append("A");
        txt.append("B");
        return txt.toString();
    }

    String mathod2(){
        StringBuilder txt = new StringBuilder();
        txt.append("C");
        txt.append("D");
        txt.append("E");
        return txt.toString();
    }
}
```

Our proposed method prevents
overlapped code clones from being detected

How to evaluate proposed method ?

- We implemented the proposed method as a tool, **FRISC**.
- Targets are 8 open source software systems.
- We compare *precision* and *recall* of **FRISC** with those of code clone detection tools.
  - Clone references are given by Bellon's experiments[1].

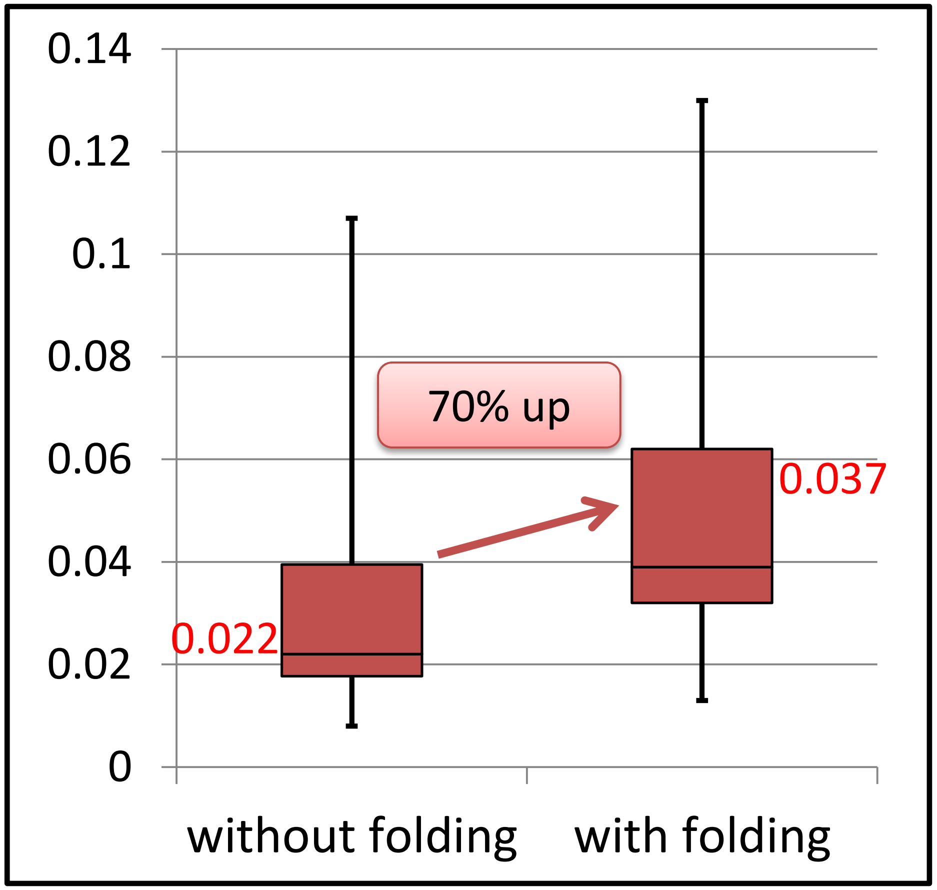
software	lang.	LOC	software	lang.	LOC
netbeans	Java	14,360	weltab	C	11,460
ant	Java	34,744	cook	C	70,008
jdtcore	Java	147,634	snns	C	93,867
swing	Java	204,037	postgresql	C	201,686

detectors	detection method
CloneDR	AST-based
CLAN	metrics-based
CCFinder	token-based
Dup	token-based
Duploc	line-based
Duplix	PDG-based
Nicad	token-based

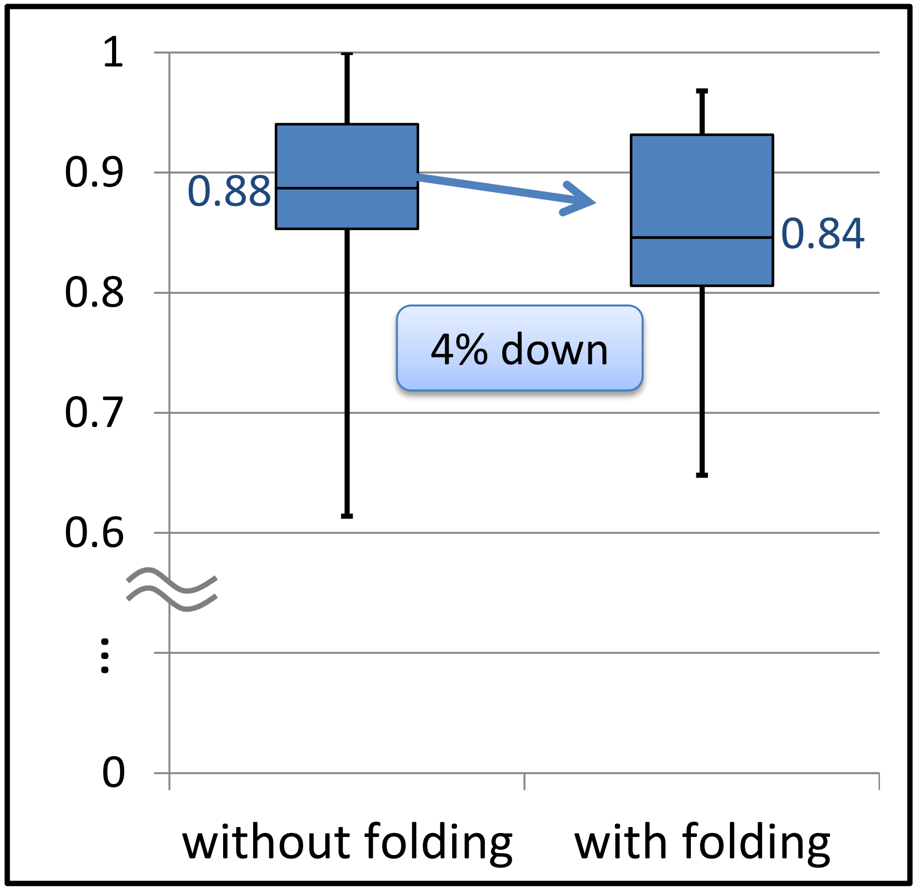
[1] S. Bellon, R. Koschke, G. Antniol, J. Krinke, and E. Merlo. Comparison and evaluation of clone detection tools. *IEEE Trans. Software Engineering*, Vol. 31, No. 10, pp.804-818, Oct. 2007.

Precision and Recall without and with folding

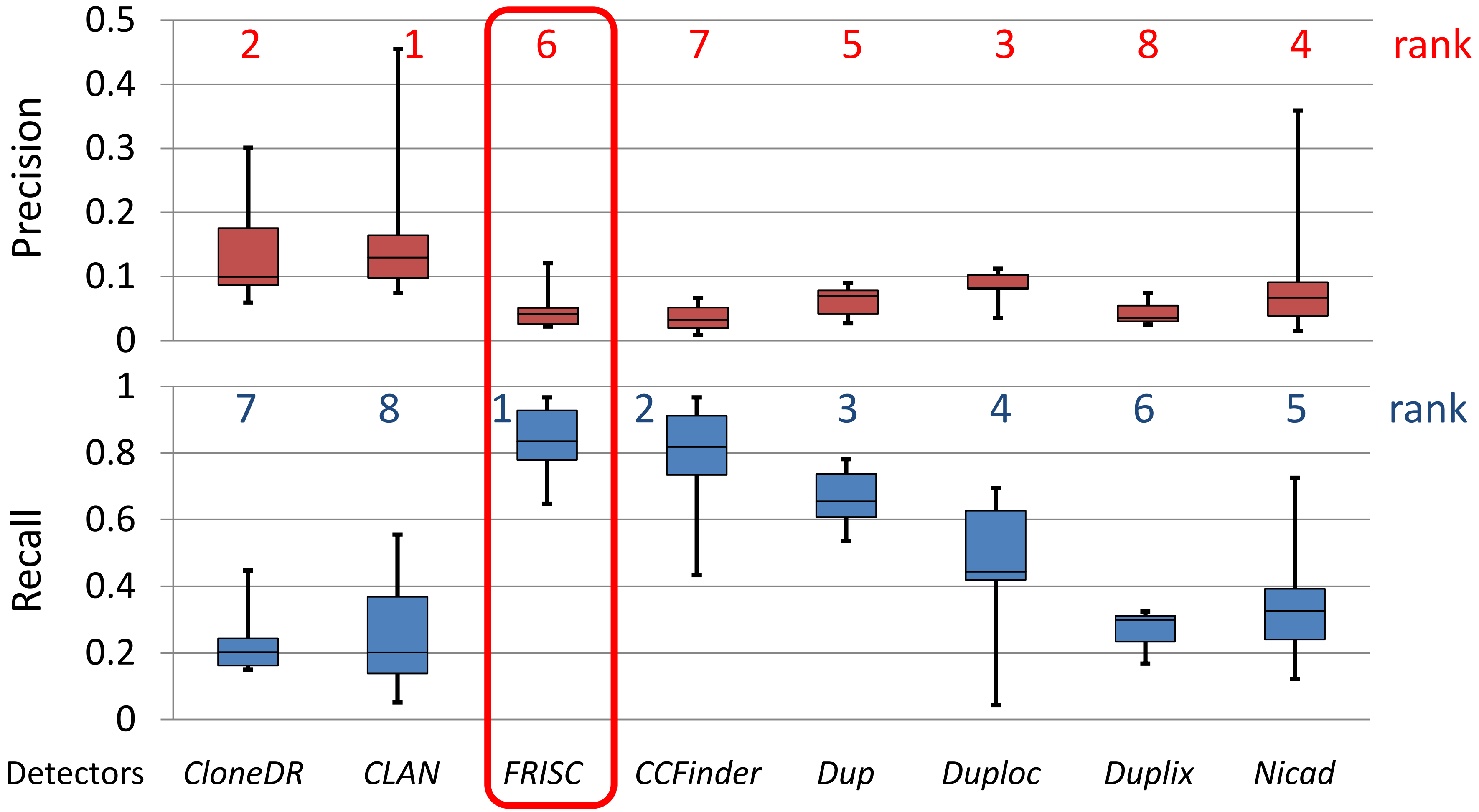
Precision



Recall

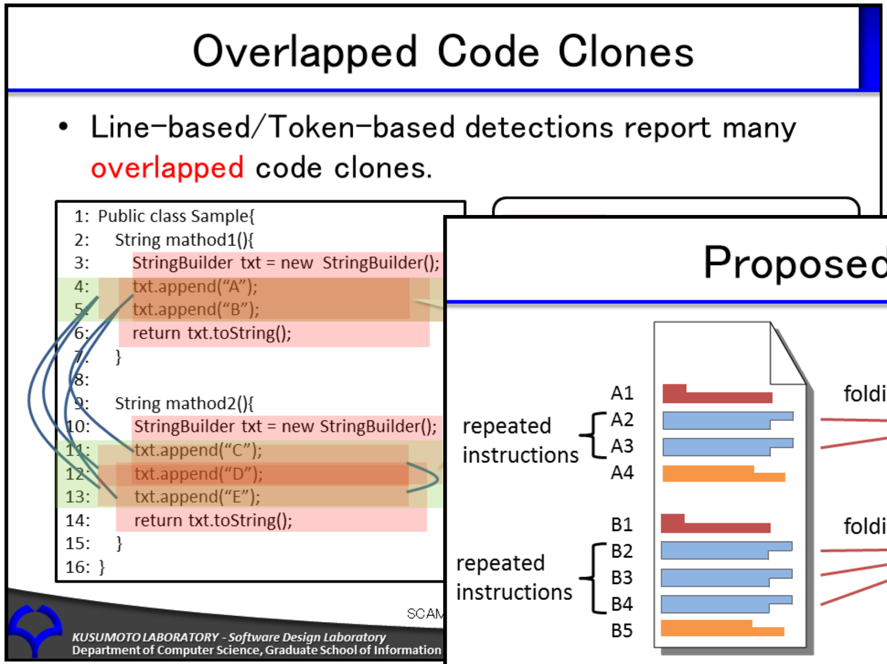


Precision and Recall of clone detectors



Detection method	AST	Metrics	Token	Token	Token	Line	PDG	Token

Conclusion



Overlapped Code Clones

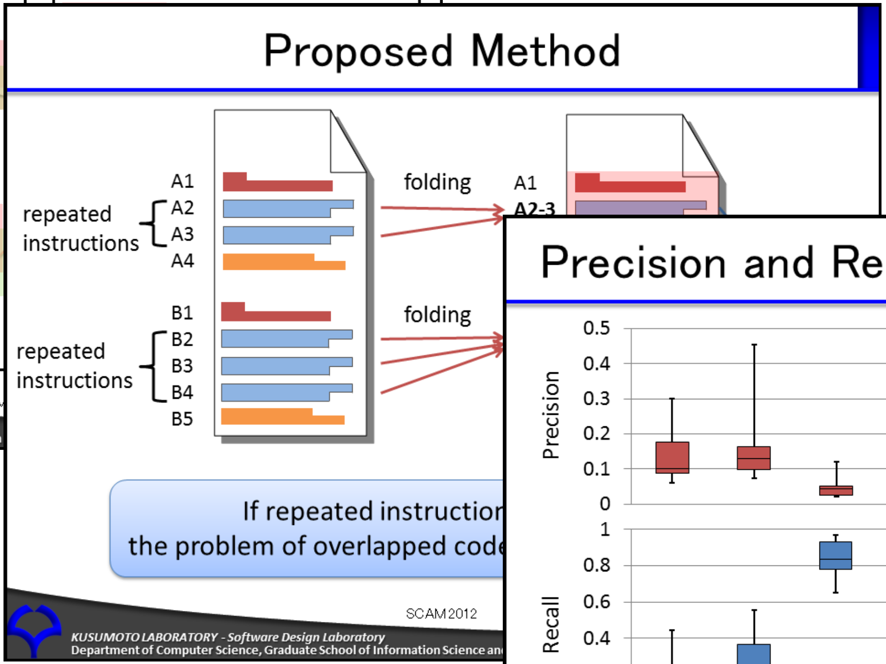
- Line-based/Token-based detections report many overlapped code clones.

```
Public class Sample{
    String mathod1(){
        StringBuilder txt = new  StringBuilder();
        txt.append("A");
        txt.append("B");
        return txt.toString();
    }

    String mathod2(){
        StringBuilder txt = new StringBuilder();
        txt.append("C");
        txt.append("D");
        txt.append("E");
        return txt.toString();
    }
}
```

SCAM

KUSUMOTO LABORATORY - Software Design Laboratory
Department of Computer Science, Graduate School of Information

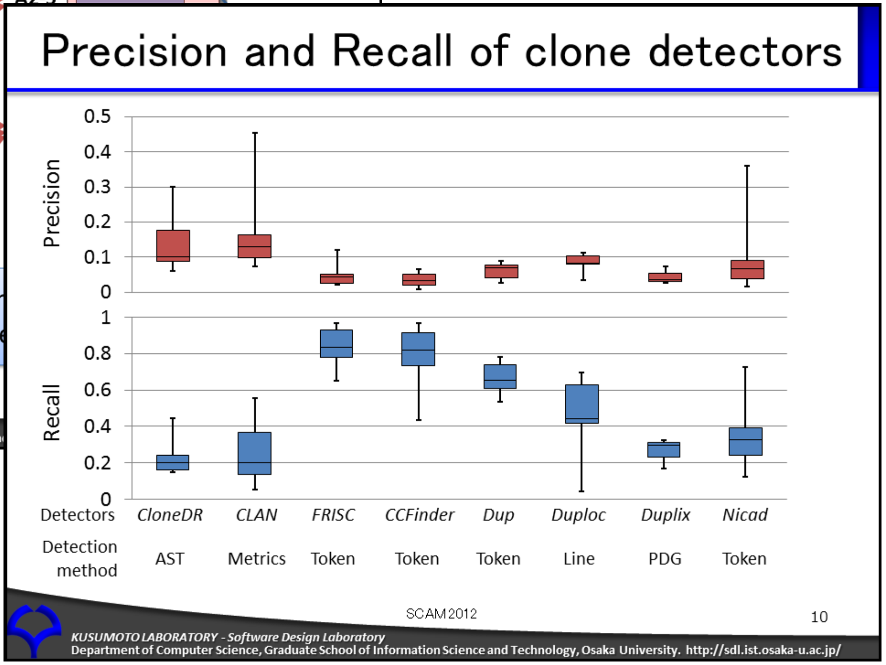


Proposed Method

If repeated instruction
the problem of overlapped code

SCAM2012

KUSUMOTO LABORATORY - Software Design Laboratory
Department of Computer Science, Graduate School of Information Science an



Precision and Recall of clone detectors

Detectors	CloneDR	CLAN	FRISC	CCFinder	Dup	Duploc	Duplix	Nicad
Detection method	AST	Metrics	Token	Token	Token	Line	PDG	Token

SCAM2012 10

KUSUMOTO LABORATORY - Software Design Laboratory
Department of Computer Science, Graduate School of Information Science and Technology, Osaka University. http://sdl.ist.osaka-u.ac.jp/