Deng CHEN, Yan-duo ZHANG, Wei WEI, Rong-cun WANG, Xiao-lin LI, Wei LIU, Shi-xun WANG, Rui ZHU, 2018. An oversampling approach for mining program specifications. *Frontiers of Information Technology & Electronic Engineering*, 19(6):737-754. https://doi.org/10.1631/FITEE.1601783

An oversampling approach for mining program specifications

Key words: Object usage scenario; API protocol mining; Program temporal specification mining; Oversampling

Corresponding author: Deng CHEN
E-mail: dchen@wit.edu.cn

ORCID: http://orcid.org/ 0000-0001-6359-801X

Motivations

1. Automatic protocol mining techniques require sufficient training data (object usage scenarios) to infer accurate and complete API protocols.

2. To increase the number of object usage scenarios (OUSs), existing approaches may suffer from significant runtime overhead.

3. Inspired by the substitution property proposed by Liskov (1987), we derive an OUS regarding a class from OUSs of its sub-classes.

Main idea

We derive an OUS regarding a class from OUSs of its sub-classes.

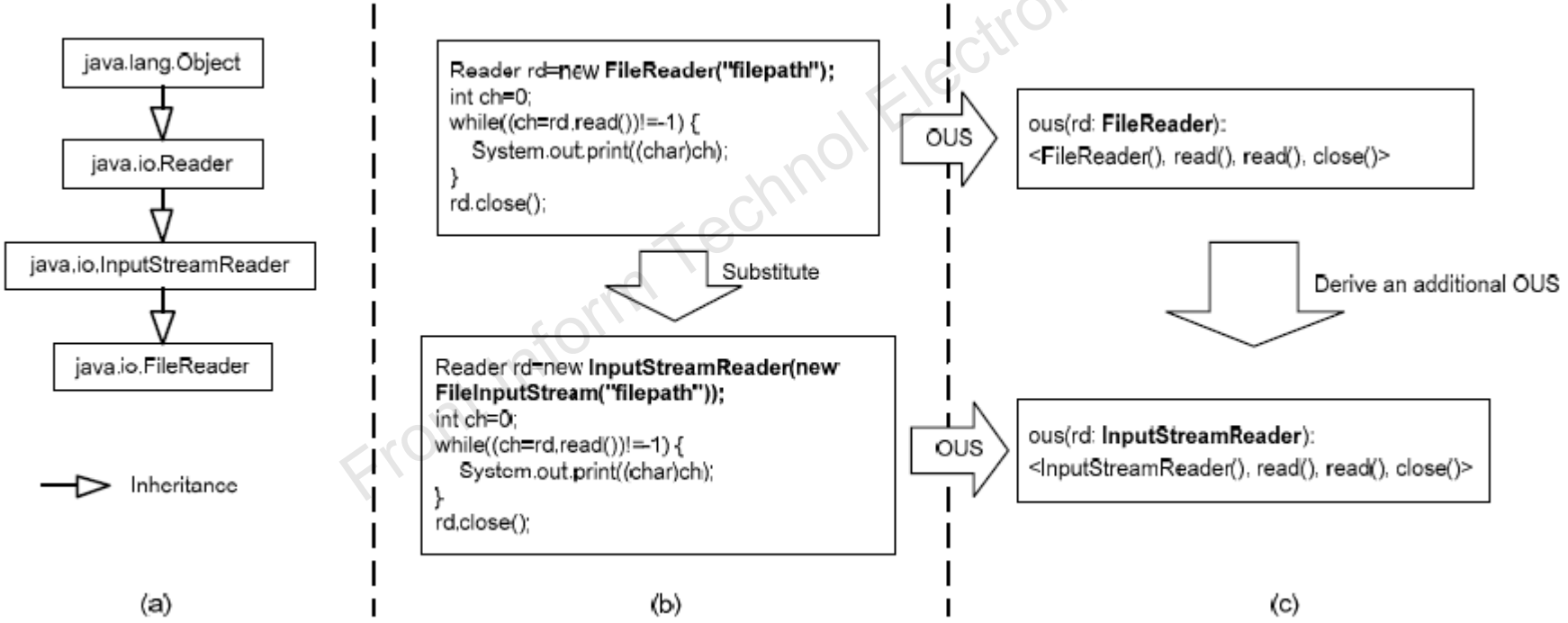


Fig. 5 Motivating example: (a) inheritance relationship; (b) class substitution; (c) OUS derivation

Methods

Heuristic 1 Let M and c be a set of methods and a class, respectively. We use $\rho(M, c)$ to denote the API protocol regarding M imposed by c. Given a sub-class c' of c, we have $\rho(\mathrm{PM}(c), c') \preceq \rho(\mathrm{PM}(c), c)$, where $\preceq$ means that protocol $\rho(\mathrm{PM}(c), c')$ is equivalent to or stricter than $\rho(\mathrm{PM}(c), c)$.

Theorem 1 (Existence of reproducing parent OUSs) Assume that c' is a sub-class of c. PM(c) denotes the set of public methods of class c. For simplicity, we use $\rho(c)$ to denote $\rho(\mathrm{PM}(c), c)$. Given an OUS $u' \in \mathrm{OUS}(c')$ that satisfies $\rho(c')$, there exists a reproducing parent OUS (RP-OUS)

$$u :< m_1, m_2, m_i, \ldots, m_j, \ \ldots >, \quad i, j \in \mathbb{N}, \tag{3}$$

$m_i, m_j \in \mathrm{PM}(c) \wedge < m_i, m_j > \in u'$, such that u satisfies $\rho(c)$.

Algorithm 1 Inheritance-based OUS extraction

Input: t, a PET achieved by the PDA technique.

Output: S, the set of OUSs implicit in t.

Notation: cid, the identifier of a group (OUS); H, a hash table, where the key is the identifier of a group, and the value is a queue used to save method events; E_{rp}, the set of RP method events derived from a general method event.

Methods:

```
1    for each method event e∈t do
2        cid←bkdrhash(e.TID+e.CN+e.OID)
3        Push e into the queue H[cid]
4        if e.MT="G" then
5            E_rp←call RPEventSet(e)
6            Add E_rp to the end of t
7        end if
8    end for
9    Copy data from H into S
10   Output S
```

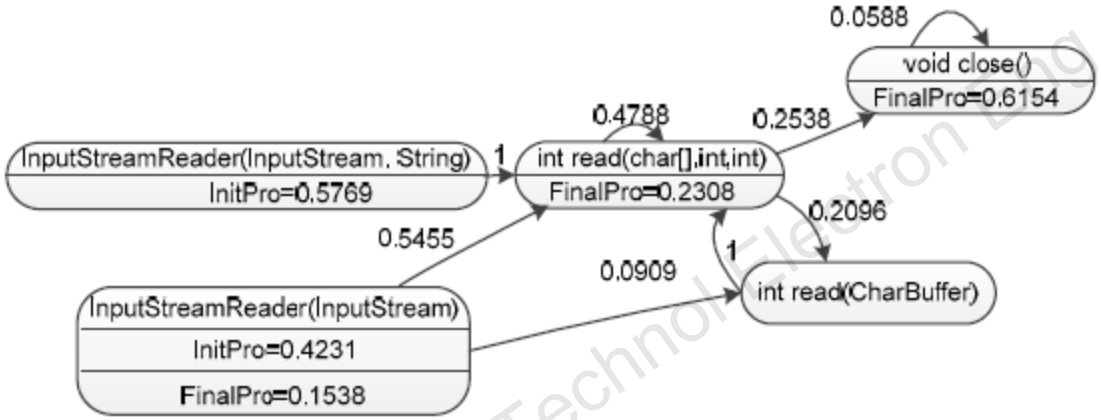
Major results

Table 4 Statistical results of the comparison test on API protocols

No.	Investigated class	OUS number		API protocol*		Comparison**
		ISpecMiner-1	ISpecMiner-2	ISpecMiner-1	ISpecMiner-2	
1	java.io.PushbackInputStream	146	146	√	√	Y
2	java.io.FileInputStream	66	66	√	√	Y
3	java.io.FileOutputStream	26	26	√	√	Y
4	java.io.BufferedReader	46	46	√	√	Y
5	java.io.BufferedWriter	28	28	√	√	Y
6	java.io.DataInputStream	280	280	√	√	Y
7	java.io.DataOutputStream	0	0	×	×	Y
8	java.io.FileReader	21	21	√	√	Y
9	java.io.FileWriter	6	6	√	√	Y
10	java.io.BufferedInputStream	7850	7850	√	√	Y
11	java.io.PrintWriter	31	31	√	√	Y
12	*java.io.FilterInputStream*	0	8192	×	√	N
13	*java.io.InputStreamReader*	26	46	√	√	N
14	*java.io.OutputStreamWriter*	13	19	√	√	N
15	*java.io.FilterOutputStream*	0	0	×	×	Y
16	*java.io.InputStream*	0	8258	×	√	N
17	*java.io.OutputStream*	0	26	×	√	N
18	*java.io.Reader*	0	89	×	√	N
19	*java.io.Writer*	0	78	×	√	N
	Total	8539	25 208	12	17	Y(12), N(7)

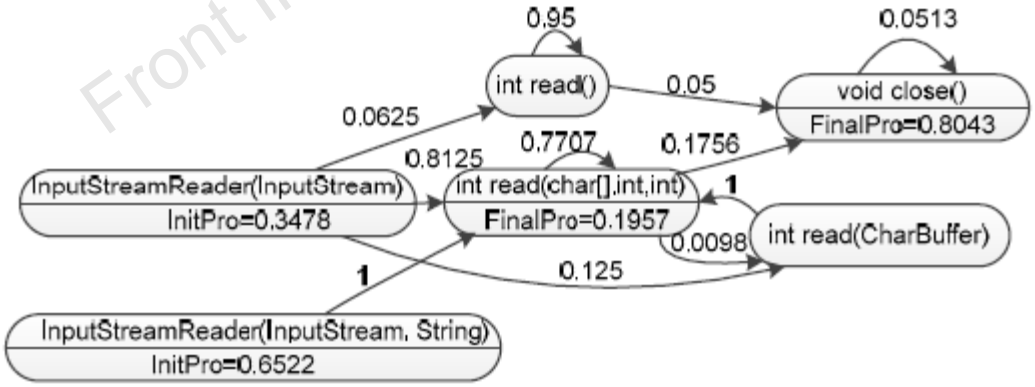
* Whether or not an API protocol was achieved; ** whether or not the API protocols mined by ISpecMiner-1 and ISpecMiner-2 are identical

Comparision of mined API protocols



Markov model of class java.io.InputStreamReader (26, Sun Mar 22 09:57:18 CST 2015)

(a)



Markov model of class java.io.InputStreamReader (46, Sun Mar 22 11:45:15 CST 2015)

(b)

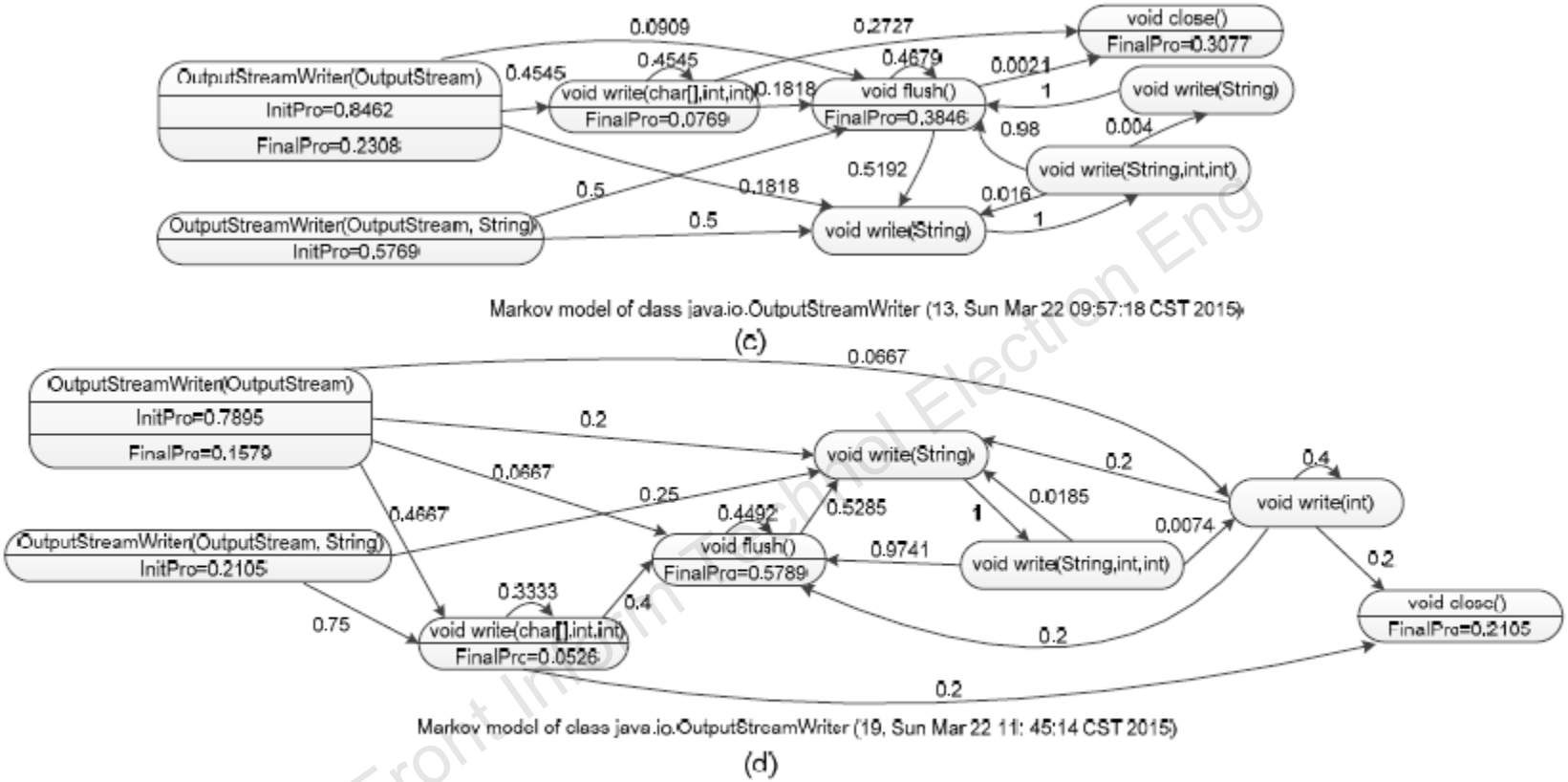


Fig. 9 API protocols achieved in our evaluation: (a) API protocol of InputStreamReader mined by ISpecMiner-1; (b) API protocol of InputStreamReader mined by ISpecMiner-2; (c) API protocol of OutputStreamWriter mined by ISpecMiner-1; (d) API protocol of OutputStreamWriter mined by ISpecMiner-2

Conclusions

1. Different from many existing approaches that increase the number of OUSs by analyzing a large codebase or more programs, we proposed an over-sampling approach IbO.

2. To ease the use of our technique, we integrated IbO with PDA and proposed an OUS-collecting algorithm IbC, which can collect general OUSs and RP-OUSs.

3. Experimental results showed that our technique collected 1.95 times more OUSs than a general approach, and that accurate and complete API protocols were more likely to be achieved.