SBES

(**S**earch-**B**ased synthesis of **E**quivalent method **S**equences)

SBES is a search-based technique to synthesize sequences of method invocations that are equivalent to a target method within a finite set of execution scenarios. The experimental results obtained on 47 methods from 7 classes (`java.util.Stack` and 6 from `org.graphstream`) show that the proposed approach correctly identifies equivalent method sequences in the majority of the cases where redundancy was known to exist, with few false positives.

For more information please contact: Andrea Mattavelli (http://www.inf.usi.ch/phd/mattavelli)

Index

Replication Package

The artifact is structured as follows:

- `common.mk` configuration file to set environment and SBES-specific properties.
- `experiments` is the main folder of the evaluation, and it is structured as follows:
    - `graphstream/` contains all the experiments on the graphstream classes.
    - `java/` contains all the experiments on the Stack class.
    - `Makefile` is the script used to execute all the experiments.
- `libraries` : folder containing all the case studies dependencies.

- `tools` : folder containing SBES, all its dependencies and the scripts to extract the salient data from the execution logs of the experiments.

This section explains how to replicate our experimental evaluation of SBES.

Requirements

To replicate our experiments, you need a Unix-like operating system (Linux or Mac OS X). You also need:

- JDK 1.7
- GNU make
- Python 2.7

Step by step Experiments Replication

This step-by-step tutorial focuses on the evaluation of our technique.
Note that a make command can be parallelized using the `-j` option (see Make documentation on parallelization (http://www.gnu.org/software/make/manual/make.html#Parallel)).

1. Download the package: SBES_Evaluation.zip (http://star.inf.usi.ch/star/software/sbes/SBES_Evaluation.zip). Extract the archive and go to the proper directory.

```
unzip SBES_Evaluation.zip
cd SBES_Evaluation/
```

2. Configure the file `common.mk` with your Java home path.

```
emacs common.mk

[...]
##
## JAVA
##
JAVAROOT=/tools/jdk1.7.0_05
```

3. Go to the experiments directory of the replication package:

```
cd ~/SBES_Evaluation/experiments
```

4. Now you can run different types of experiments (each one includes running 30 times SBES on each target method):
   - one single method at a time;

     ```
     make CLASS=java/stack TARGET_METHODS=stack.util.Stack.addElement\(Object\)
     ```

   - one single class at a time (with all its methods);

     ```
     make CLASS=java/stack
     ```

   - the whole set of experiments. This command invokes make data at the end of the execution.

     ```
     make fse2014
     ```

   The replication of the entire evaluation process requires several hours (for example, it requires up to 20 hours on a server with 16 parallel jobs, that is with `-j16` option). If you want to limit the number of iterations, you can run all the aforementioned commands with the option ITER.

   ```
   make fse2014 ITER=10
   ```

5. To clean the experiments you can either erase all data

   ```
   make veryclean
   ```

   or erase the data of a particular class.

   ```
   make clean CLASS=java/stack
   ```

6. Now you can invoke the scripts to generate the reports (if you invoke the whole execution of the experiments you do not need to invoke this command).

   ```
   make data
   ```

Checking the Results of the Experiments

Effectiveness After the invocation of `make data` , in the `experiments` folder there should be two files named `graphstream_es.txt` and `java_es.txt` .

Each file reports, for each triple of class-method-iteration, the list of synthesized equivalent sequences. For example:

```
cat graphstream_es.txt
========== graphstream/edge/org.graphstream.graph.implementations.AbstractEdge.addAtt
ribute(String,Object)/1/log
EqSeq1 clone.setAttribute(p0, p1);
EqSeq2 clone.changeAttribute(p0, p1); [...]
```

The manual inspection of those two files allows the generation of the data to fill both Table 1 and Table 2 (RQ1 and RQ2), and Table 4 (RQ4) in the paper.

Efficiency SBES stores the time required to perform each step of the technique in a CSV file named `output.csv` . This file is stored in a directory with the same name as the current method under investigation (for example, stack.util.Stack.addElement(Object)) inside the iteration directory. The CSV file is structured as follows:

- **process** reports the overall computation time (ms)
- **scenario** reports the time spent to parse the execution scenario (ms)
- **synthesis** reports the time spent to synthesize an equivalent sequence (ms)
- **counterexample** reports the time spent to generate a counterexample (ms)
- **iteration** reports the time spent for each iteration, that is the time required to create the stubs, to synthesize an equivalent sequence and to generate a counterexample (ms)

The make data command leads to the generation of two CSV files for each class XYZ:

1. `max_synthesis_time_XYZ.csv` contains two different columns: first, an identifier composed of the pair ; second, the corresponding maximum time required to synthesize an equivalent sequence (in seconds).
2. `max_counterexample_time_XYZ.csv` contains two different columns: first, an identifier composed of the pair ; second, the corresponding maximum time required to generate a counterexample (in seconds).

To fill Table 3 (RQ3) it is sufficient to open each CSV file and to compute the maximum and the median time of all the reported experiments.

Evaluation

In this section we report the complete and extensive set of results obtained by applying SBES on 47 methods from 7 classes.

Table 2: RQ1, RQ2: Effectiveness of the Approach

- **FP** - Total number of false positives on 30 runs
- **Min** - Minimum number of equivalent sequences synthesized (true positives) on 1 run
- **Max_T** - Maximum number of equivalent sequences synthesized (true positives) on 30 runs
- **Max_R** - Maximum number of equivalent sequences synthesized (true positives) on 1 run
- **AVG** - Average number of equivalent sequences synthesized (true positives)
- **Prec** - Precision
- **Rec** - Recall

Case Study	Equivalent Sequences		Synthesized Equivalences						
	Original	Equivalent	FP	Min	Max_T	Max_R	AVG	Prec	Rec
	add(int index, E item)	insertElementAt(item, index)	0	0	2	2	0.43	1.00	1.00
		addAll(index, collection)							
	add(E item)	addAll(size(), new Collection<E>(item))	0	1	3	3	2.10	1.00	0.50
		addAll(index, collection)							
		addElement(item)							
		add(size(), item)							
		insertElementAt(item, size())							
		push(item)							
		addAll(size(), new Collection<E>(item))							
		addAll(index, collection)							
		add(item)							
		add(size(), item)							

Case Study	Equivalent Sequences		Synthesized Equivalences						
	Original	Equivalent	FP	Min	Max	Max	AVG	Prec	Rec
java.util.Stack	addElement(E item)	insertElementAt(item, size())	0	2	4	3	2.17	1.00	0.70
		push(item)							
	clear()	removeAllElements()	1	1	3	3	2.77	0.99	1.00
		retainAll(empty collection)							
		setSize(0)							
	elementAt(index)	get(index)	7	0	1	1	0.90	0.79	1.00
	firstElement()	get(0)	6	0	2	2	1.57	0.89	1.00
		elementAt(0)							
	get(index)	elementAt(index)	6	0	1	1	0.80	0.80	1.00
	indexOf(Object item)	indexOf(item, 0)	4	0	2	2	1.70	0.93	1.00
		lastIndexOf(item)							
	lastElement()	peek()	1	1	2	2	1.17	0.97	0.50
		elementAt(size()-1)							
		get(size()-1)							
		obj=pop(); push(obj)							
	peek()	lastElement()	1	1	2	2	1.23	0.97	1.00
		obj=pop(); push(obj)							
	pop()	peek(); removeElementAt(size()-1)	0	0	2	2	0.60	1.00	1.00
		remove(size()-1)							

Case Study	Equivalent Sequences		Synthesized Equivalences						
	Original	Equivalent	FP	Min	Max	Max	AVG	Prec	Rec
	push(E item)	addAll(size(), new Collection<E>(item))	0	2	2	2	2.00	1.00	1.00
		addAll(index, collection)							
		add(item)							
		add(size(), item)							
		insertElementAt(item, size())							
		addElement(item)							
	remove(E element)	removeElement(E element)	2	0	2	1	0.80	0.92	0.50
		removeElementAt(lastIndexOf(E element))							
		removeElementAt(indexOf(E element))							
		removeAll(Collection(E element))							
	remove(int index)	removeElementAt(int index)	0	0	1	1	0.60	1.00	1.00
	set(int index, E element)	remove(index); add(index, element)	8	0	2	2	0.50	0.65	1.00
		setElementAt(element, index)							
graphstream.Path	getEdgeCount()	getEdgeSet().size()	20	0	2	2	1.80	0.73	1.00
		getEdgePath().size()							
	getNodeCount()	getNodePath().size()	30	2	3	3	2.77	0.73	1.00
		getNodeSet().size()							
		size()							
		getSourceNode()							

Case Study	Equivalent Sequences		Synthesized Equivalences						
	Original	Equivalent	FP	Min	Max	Max	AVG	Prec	Rec
	getNode0()		0	2	2	2	2.00	1.00	1.00
graphstream.Edge		node=getTargetNode(); getOpposite(node)							
	getSourceNode()	getNode0()	0	11	2	2	2.00	1.00	0.85
		node=getTargetNode(); getOpposite(node)							
	getNode1()	getTargetNode()	0	1	2	2	1.97	1.00	1.00
		node=getSourceNode(); getOpposite(node)							
	getTargetNode()	getNode1()	15	2	2	2	2.00	0.80	1.00
		node=getSourceNode(); getOpposite(node)							
	changeAttribute(String attribute, Object... values)	setAttribute(String attribute, Object... values)	0	2	2	2	2.00	1.00	1.00
		addAttribute(String attribute, Object… values)							
	setAttribute(String attribute, Object... values)	addAttribute(String attribute, Object… values)	0	2	2	2	2.00	1.00	1.00
		changeAttribute(String attribute, Object... values)							
	addAttribute(String attribute, Object... values)	changeAttribute(String attribute, Object... values)	0	2	2	2	2.00	1.00	1.00
		setAttribute(String attribute, Object... values)							
		getFirstAttributeOf(String...)							
		getFirstAttributeOf(Class<?>, String...)							

Case Study	Equivalent Sequences		Synthesized Equivalences						
	Original	Equivalent	FP	Min	Max	Max	AVG	Prec	Rec
	getAttribute(String)	getAttribute(String,Class<?>)	0	0	3	3	1.57	1.00	1.00
	getFirstAttributeOf(String...)	getAttribute(String)	0	1	3	3	1.80	1.00	1.00
		getFirstAttributeOf(Class<?>, String...)							
		getAttribute(String,Class<?>)							
graphstream.SingleNode	changeAttribute(String attribute, Object... values)	setAttribute(String attribute, Object... values)	0	2	2	2	2.00	1.00	1.00
		addAttribute(String attribute, Object… values)							
	setAttribute(String attribute, Object... values)	addAttribute(String attribute, Object… values)	0	2	2	2	2.00	1.00	1.00
		changeAttribute(String attribute, Object... values)							
	addAttribute(String attribute, Object... values)	changeAttribute(String attribute, Object... values)	0	2	2	2	2.00	1.00	1.00
		setAttribute(String attribute, Object... values)							
	getAttribute(String)	getFirstAttributeOf(String...)	0	0	3	3	0.63	1.00	1.00
		getFirstAttributeOf(Class<?>, String...)							
		getAttribute(String,Class<?>)							
	getFirstAttributeOf(String...)	getAttribute(String)	0	0	3	3	1.50	1.00	1.00
		getFirstAttributeOf(Class<?>, String...)							
		getAttribute(String,Class<?>)							

Case Study	Equivalent Sequences		Synthesized Equivalences						
	Original	Equivalent	FP	Min	Max	Max	AVG	Prec	Rec
graphstream.MultiNode	changeAttribute(String attribute, Object... values)	setAttribute(String attribute, Object... values)	0	2	2	2	2.00	1.00	1.00
		addAttribute(String attribute, Object… values)							
	setAttribute(String attribute, Object... values)	addAttribute(String attribute, Object… values)	0	2	2	2	2.00	1.00	1.00
		changeAttribute(String attribute, Object... values)							
	addAttribute(String attribute, Object... values)	changeAttribute(String attribute, Object... values)	0	2	2	2	2.00	1.00	1.00
		setAttribute(String attribute, Object... values)							
	getAttribute(String)	getFirstAttributeOf(String...)	0	1	3	3	1.20	1.00	1.00
		getFirstAttributeOf(Class<?>, String...)							
		getAttribute(String,Class<?>)							
	getFirstAttributeOf(String...)	getAttribute(String)	0	0	3	3	1.17	1.00	1.00
		getFirstAttributeOf(Class<?>, String...)							
		getAttribute(String,Class<?>)							
	x()	at(0)	7	0	1	1	0.23	0.50	1.00
	y()	at(1)	0	0	1	1	0.27	1.00	1.00
		set(0, x); set(1, y)							
		v=new Vector2(x, y); copy(v)							

Case Study	Equivalent Sequences			Synthesized Equivalences					
	Original	Equivalent	FP	Min	Max	Max	AVG	Prec	Rec
	set(double x, double y)	p=new Point2(x, y); copy(p)	0	0	5	5	1.37	1.00	1.00
		v=new Vector2(x, y); add(v)							
		p=new Point2(); p.moveTo(x, y); copy(p)							
graphstream.Vector2	fill(double value)	p=new Point2(); p.setX(value); p.setY(value); copy(p)	3	0	10	7	3.87	0.97	1.00
		v=new Vector2(value, value); copy(value)							
		v=new Vector2(value, value); add(value)							
		p=new Point2(value, value); set(p.x, p.y)							
		p=new Point2(value, value); copy(p)							
		p=new Point2(); p.move(value, value); copy(p)							
		p=new Point2(); p.moveTo(value, value); copy(p)							
		p=new Point2(); p.make(value, value); copy(p)							
		scalarAdd(value)							
		scalarSub(value); scalarDiv(-1)							
		set(v.x(), v.y())							
		add(v)							
		fill(1); mult(v)							

Case Study	Equivalent Sequences		Synthesized Equivalences						
	Original	Equivalent	FP	Min	Max	Max	AVG	Prec	Rec
	copy(Vector2 v)	set(v.x, v.y)	3	0	4	3	1.23	0.93	1.00
graphstream.Vector3	x()	at(0)	0	0	1	1	0.10	1.00	1.00
	y()	at(1)	1	0	1	1	0.27	0.89	1.00
	z()	at(2)	1	0	1	1	0.20	0.86	1.00
	set(double x, double y, double z)	set(0, x); set(1, y); set (2, z)	0	0	5	3	0.40	1.00	1.00
		v=new Vector3(x, y, z); copy(v)							
		p=new Point3(x, y, z); copy(p)							
		v=new Vector3(x, y, z); add(v)							
		p=new Point3(); p.moveTo(x, y, z); copy(p)							
	fill(double value)	p=new Point3(); p.setX(value); p.setY(value); p.setZ(value); copy(p)	37	0	10	6	0.14	0.10	1.00
		v=new Vector3(value, value, value); copy(value)							
		v=new Vector3(value, value, value); add(value)							
		p=new Point3(value, value, value); set(p.x, p.y, p.z)							
		p=new Point3(value, value, value); copy(p)							
		p=new Point3(); p.move(value, value, value; copy(p)							

Case Study	Equivalent Sequences			Synthesized Equivalences					
	Original	Equivalent	FP	Min	Max	Max	AVG	Prec	Rec
		p=new Point3(); p.moveTo(value, value, value); copy(p)							
		p=new Point3(); p.make(value, value, value); copy(p)							
		scalarAdd(value)							
		scalarSub(value); scalarDiv(-1)							
	copy(Vector3 v)	set(v.x(), v.y(), v.z())	1	0	4	4	2.27	0.99	1.00
		add(v)							
		fill(1); mult(v)							
		set(v.x, v.y, v.z)							

Table 3: RQ3: Efficiency of the Approach

Case Study		Synthesis Time		Counterexample Time	
		AVG	Median	AVG	Median
	add(int index, E item)	19.4s	12.0s	16.0s	12.0s
	add(E item)	19.6s	17.0s	-	-
	addElement(E item)	19.7s	15.0s	-	-
	clear()	19.8s	14.0s	14.6s	12.0s
	elementAt(index)	47.2s	24.0s	12.5s	11.0s
	firstElement()	43.0s	19.0s	9.2s	9.0s
	get(index)	39.0s	24.5s	11.5s	12.0s

Case Study		Synthesis Time		Counterexample Time	
java.util.Stack		AVG	Median	AVG	Median
	indexOf(Object item)	40.6s	27.0s	10.1s	9.0s
	lastElement()	18.8s	12.0s	9.0s	9.0s
	peek()	20.7s	11.5s	13.7s	10.0s
	pop()	26.2s	20.0s	22.2s	16.0s
	push(E item)	19.2s	19.0s	-	-
	remove(E element)	21.2s	22.0s	17.8s	20.0s
	remove(index)	36.6s	21.5s	21.0s	12.0s
	set(int index, E element)	84.2s	62.0s	12.5s	12.0s
graphstream.Path	getEdgeCount()	19.7s	16.0s	18.4s	15.0s
	getNodeCount()	35.9s	28.0s	12.0s	12.0s
graphstream.Edge	getNode0()	19.1s	19.0s	6.0s	6.0s
	getSourceNode()	82.8s	80.5s	6.0s	6.0s
	getNode1()	15.8s	15.0s	6.2s	6.0s
	getTargetNode()	70.3s	36.0s	5.7s	6.0s
	changeAttribute(String attribute, Object values)	15.0s	14.0s	5.7s	6.0s
	setAttribute(String attribute, Object values)	15.2s	14.0s	5.8s	6.0s
	addAttribute(String attribute, ObjectÉ values)	14.0s	13.0s	5.6s	6.0s
	getAttribute(String)	18.0s	15.0s	5.6s	6.0s
	getFirstAttributeOf(String...)	21.5s	16.5s	5.8s	6.0s

Case Study		Synthesis Time		Counterexample Time	
		AVG	Median	AVG	Median
graphstream.SingleNode	changeAttribute(String attribute, Object values)	18.0s	17.5s	6.2s	6.0s
	setAttribute(String attribute, Object values)	20.9s	16.0s	5.8s	6.0s
	addAttribute(String attribute, Object values)	18.2s	17.5s	6.2s	6.0s
	getAttribute(String)	15.6s	16.0s	5.3	5.0s
	getFirstAttributeOf(String...)	12.9s	12.0s	5.1s	5.0s
graphstream.MultiNode	changeAttribute(String attribute, Object values)	25.8s	23.5s	8.2s	8.0s
	setAttribute(String attribute, Object values)	24.8s	21.5s	8.2s	8.0s
	addAttribute(String attribute, ObjectÉ values)	22.4s	21.5s	8.2s	8.0s
	getAttribute(String)	16.0s	15.0s	5.8s	6.0s
	getFirstAttributeOf(String...)	16.1s	14.0s	5.8s	5.0s
graphstream.Vector2	x()	59.1s	35.0s	6.4s	6.0s
	y()	61.7s	32.0s	20.0s	20.0s
	set(x, y)	32.1s	12.0s	9.2s	10.0s
	fill(double value)	13.6s	15.0s	9.6s	7.0s
	copy(Vector2 v)	18.8s	11.0s	5.2s	5.0s
	x()	25.8s	16.5s	5.0s	5.0s
	y()	49.1s	26.0s	6.5s	6.5s

Case Study		Synthesis Time		Counterexample Time	
		AVG	Median	AVG	Median
graphstream.Vector3	z()	110.8s	106.5s	12.5s	12.5s
	set(x, y, z)	95.8s	56.0s	6.7s	5.0s
	fill(double value)	75.5s	10.5s	8.8s	10.0s
	copy(Vector3 v)	80.1s	33.0s	7.1s	6.0s

Table 4: RQ4: Effectiveness of the Counterexamples

Case Study		False Positives	Discarded	Efficiency
java.util.Stack	add(int index, E item)	0	4	100%
	add(E item)	0	0	-
	addElement(E item)	0	0	-
	clear()	1	52	98.11%
	elementAt(index)	7	22	75.86%
	firstElement()	6	12	66.66%
	get(index)	6	9	60%
	indexOf(Object item)	4	18	81.81%
	lastElement()	1	7	87.50%
	peek()	1	21	95.45%
	pop()	0	0	-
	push(E item)	0	0	-
	remove(element)	2	3	60%