International Journal of Software Engineering and Its Applications
Vol.8, No.2 (2014), pp.385-390
http://dx.doi.org/10.14257/ijseia.2014.8.2.37

An Efficient Algorithm of Simplest LTS using Strong Equivalence for White Box Approach

Byungho Park[1], Yong B. Park[2] and Robert Youngchul Kim[1]

[1]*Dept. of CIC, Hongik University, Sejong Campus, 339-701, Korea*
{bhpark, bob}@selab.hongik.ac.kr
[2]*Dept. of Computer Science, Dankook University, 330-714, Korea*
ybpark@dankook.ac.kr

Abstract

In a large system, it is the most important work to consider the verification of the overall source codes and also to test minimal test cases for the maximal test coverage. In the current testing situations, if a complicated and huge system is developed, it may be difficult to test all the source codes. How could it extract small-sized test cases and satisfy high coverage if possible? For this, our previous research worked a simplest LTS using two steps of the transformation algorithm. Bur this paper suggests an efficient algorithm to transform original LTS into Simplest LTS at one time to extract minimal test cases for the maximal test coverage. It can expect an advantage to decrease computer sources and testing time epochally because it tests errors after excluding the recursion of software source codes with Strong Equivalence and optimizing them.

Keywords: *Test Case, Strong Equivalence, Simplest LTS, Optimization*

1. Introduction

As the size and complexity of a critical system such as nuclear energy and express trains has expanded with the use of computer, it has become very important to remove any danger inherent in a computer. Therefore, the importance of testing in software development has been more and more increasing. The verification of the overall source codes in a large system is the most ideal but the real system is so wide that it is general to test only a few test cases by the maximal test coverage.

Here, the coverage rate indicates how many coverage criteria it has achieved to be tested. Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. This activity results in the actually expected difference between their effects. In a word, testing executes a system to identify any gap, error or missing requirement in contrast to the actual desires or requirements [1].

According to ANSI/IEEE 1059 standard, Testing can be defined as "A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item" [2]. The most effective software testing is to remove errors and recursions through verification in all the sources so as to minimize the numbers of bugs inside the program, and to satisfy the function and accuracy required by users. However, if the system to be developed is large-sized, it may be difficult to test all of the source codes. So it is universal to try to extract small-sized test cases and satisfy high coverage if possible. For this, our previous research worked a simplest LTS using two steps of the transformation algorithm.

ISSN: 1738-9984 IJSEIA

But this paper suggests an algorithm which can transform original LTS into Simplt LTS at one time. It can expect an advantage to decrease computer sources and testing time epochally because it tests errors after excluding the recursion of software source codes with Strong Equivalence and optimizing them.

This paper is organized as follows. Section 2 look at the related studies and Section 3 explained the suggested Improved Simplest Algorithm Transformation. Section 4 describes the conclusion and the future research directions.

2. Preliminaries

2.1. Related Work

Generally, the equivalence for processes is based on observations. The idea of the equivalence is that when an observer inspects behaviors of the two processes from the environment, they are equal if the observer cannot distinguish these behaviors [3]. Processes can develop software systems which are composed of the subset processes.

The process can represent semantics by the Labeled Transition System (LTS)[3].

Definition 2.1 A labeled transition system[3]

Sys is a 4-tuple <S, A, →, s_0 >, where

(1) *S is a non-empty set of state.*

(2) *A is a set of actions.*

(3) *→ S X A X S and is a set of transition relations.*

(4) *s_0 ∈ S is the initial state of Sys.* □

Therefore, we can say

$$\rightarrow = \bigcup_{a \in A} \{ \xrightarrow{a} \mid \xrightarrow{a} S X S' \}$$

Definition 2.2 The transition relation between the processes is referred to by the following rules [3].

$$\frac{}{a;P \xrightarrow{a} p'} \qquad \frac{P \xrightarrow{a} p'}{P \,[]\, Q \xrightarrow{a} p'} \qquad \frac{Q \xrightarrow{a} Q'}{P \,[]\, Q \xrightarrow{a} p'}$$ □

The equivalence for the processes is based on observations. The idea of equivalence is that when an observer inspects behaviors of two processes from environment, they are equal if the observer cannot distinguish these behaviors [4].

In the following, we introduce the concept of strong equivalence.

Definition 2.3 A relation **R** over processes[3] is a strong bisimulation if P **R** Q implies, for all a ∈ A,

(1) *whenever $P \xrightarrow{a} p'$ then, for some Q', $Q \xrightarrow{a} Q'$ and P' **R** Q'*

(2) *whenever $Q \xrightarrow{a} Q'$ then, for some P', $P \xrightarrow{a} p'$ and P' **R** Q'* □

For processes P and Q, if there exists a strong bisimulation R such that P, Q ∈ **R**, then P and Q are strongly equivalent, written as P ~Q. The strong equivalence does not consider the internal actions which cannot be observed from the external environment.

In this paper, we deal only with observable actions because the strong equivalence is enough for our purpose.

2.2. Smallest Labeled Transition System

A process which has an equivalent relation with any P but whose expression is different with P generally exists infinitely. We introduce an algorithm [4] which can derive the process of only one form by deriving the process of only one form to be the smallest LTS among processes which are strong equivalence to P exclusive of commutative law.

Definition 2.5 The smallest LTS are composed of four-tuples [4].

Sys = < *S, A, T, s*$_0$ >, *where*

(1) $S = \{p' \mid p(\rightarrow)^n P', n \geq 0\}$,

(2) $A = \{a \mid P \xrightarrow{a} P',\ P, P' \in S\}$,

(3) $T = \{\xrightarrow{a} \mid P \xrightarrow{a} P',\ P, P' \in S\}$,

(4) $s_0 = P$ □

$p(\rightarrow)^n P'$ means that there exists two types of processes as non-recursive LTS and recursive LTS. In this paper, we just limited the range to non-recursive LTS.

2.3. Simplest Labeled Transition System

We already proposed an algorithm to translate from the smallest LTS to the simplest one [5]. The smallest LTS has the smallest number of states and nodes.

Even if an original LTS has a tree structured form, the corresponding LTS is not tree-structured but a graph, in general. In order to compare two systems using each distinct layer, the minimal LTS must be changed to a tree structured LTS, in other words, the simplest LTS [5].

Definition 2.6 The simplest LTS is defined as follows [5].

Let P and Sys be a process and its LTS, respectively. Sys is simplest LTS if each state has only a mother state corresponding to the process which is strong equivalence to P. □

Figure 1 shows an example of Simplest LTS via Smallest LTS from Original LTS [5].

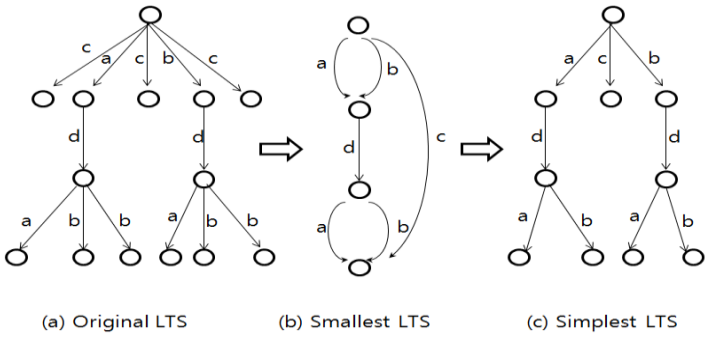

Figure 1. Simplest LTS via Smallest LTS from Original LTS

3. Improved Simplest Algorithm Transformation

The above algorithm needs to be performed twice even though it can make the simplest LTS. In other words, as shown in the figure, leaf nodes need to have only one edge in order to

compare the simplest LTS optimization contraction and the equivalence of each layer between the two LTS's, and also, each node has to go through the transition process to the simplest LTS to transform so that it can only one parent. It requires to be performed twice for improvements.

Therefore, this paper suggests an improved algorithm, which can be transformed into Original LTS and Simplest LTS based on Strong Equivalence.

The suggested algorithm:

If For $P, P' \in S$, Suppose $P \xrightarrow{a} P'$, then for some Q', $Q \xrightarrow{a} Q'$ and P' ***R*** Q',

For $Q, Q' \in S$, Suppose $Q \xrightarrow{a} Q'$, then for some P', $P \xrightarrow{a} P'$ and P' ***R*** Q',

Then $P \sim Q$.

This algorithm will be performed from bottom-up method.

For some P'', $a,b \in S$, $P \xrightarrow{(a)*} P'' \xrightarrow{(a)+} P'$.

Step 1. Reduce branches if having the same branch(trace) in case of over two times.

If $|P'' \xrightarrow{(a)+} P^m| \leq 2$, $m \leq 2$ then $\left|P'' \xrightarrow{(a)+} P^n\right| = 1$, $n=1$

Step 2. Avoid redundancy if having the same sub tree(process) in case of over two times.

If $|P'' \xrightarrow{(a,b)+} P'| \leq 2$, $a \neq b$ then $\left|P'' \xrightarrow{(a,b)+} P'\right| = 1$.

Step 3. Step up and continue (Step 1) and (Step 2)* to s_0 (root node) $\in S$.

Here, + : one or more, * : zero or more

Proof:

Strong equivalence will be proved by the method of Induction here.

Suppose $P \xrightarrow{a} P'$, $P, P' \in S$ then for some Q', $Q \xrightarrow{a} Q'$ and P' ***R*** Q'

There is Strong Equivalence between the systems by Def. 2.1.

For all $P, P', P'', P''', Q, Q', Q'', Q''' \in S$ and $\xrightarrow{a}, \xrightarrow{dept\ n-1}, \xrightarrow{dept\ n} \in T$

i) In case of depth n=1,

if $P \xrightarrow{a} P'$, then for some Q', $Q \xrightarrow{a} Q'$

It is an obvious strong equivalence because the next state of each system is transited by only one and the same transition.

ii) In case of depth n=n-1,

If it is supposed to be the same action and the same size for transition to depth n-1, then $P \xrightarrow{dept\ hn-1} P''$, for some Q'', $Q \xrightarrow{dept\ n-1} Q''$

The above two systems are strong equivalence.

iii) In case of depth n=n and more over, each system is transited to the next state only by the same action respectively.

$P \xrightarrow{dept\ n} P'''$, for some Q''', $Q \xrightarrow{dept\ n} Q'''$

Therefore, Def. 2.1,shows that P, Q is satisfied by the strong equivalence. □

An example below shows that it changed from original LTS to Simplest LTS using Strong Equivalence algorithm.

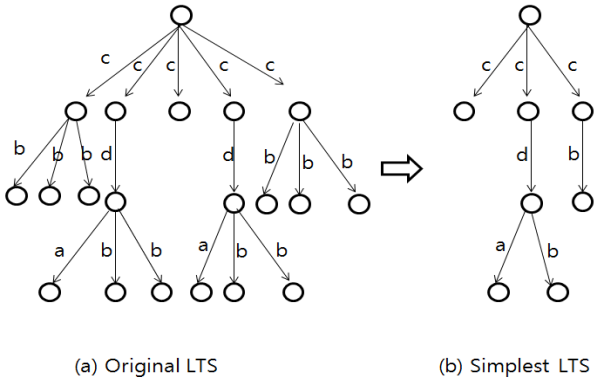

Figure 2. The Simplest LTS from the Original LTS at Once

Figure 2 shows the expected effects after using and optimizing this algorithm.

As shown in Table 1, it is recognized that if Simplest LTS is used, test case is generated in this method. It is also useful because computer resource and testing time can be decreased in a large scale.

Table 1 shows the result of reducing the number of nodes, edges and branches in Figure 2 with the suggested algorithm.

Table 1. Comparison of Original LTS to Simplest LTS for Test Case

	original	simplest	result
node	20	8	-12
edge	19	7	-12
branch(choice)	5	2	-3

4. Conclusion

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.

In the current testing situations, if a complicated and huge system is developed, it may be difficult to test all the source codes.

So, it is universal to try to extract small-sized test cases and satisfy high coverage if possible. For this, our previous research worked a simplest LTS using two steps of the transformation algorithm.

But this paper suggests an algorithm which can transform original LTS into Simplest LTS at one time. It can expect an advantage to decrease computer sources and testing time epochally because it tests errors after excluding the recursion of software source codes with Strong Equivalence and optimizing them.

Future studies are expected to find out a method to exclude recursion about the same type existing in the optimized LTS.

Acknowledgements

This work was supported by the IT R&D Program of MKE/KEIT [10035708, "The Development of CPS(Cyber-Physical Systems) Core Technologies for High Confidential Autonomic Control Software"] and Basic Science Research Program through the National

Research Foundation of Korea(NRF) funded by the Ministry of Education (2013R1A1A2011601)

References

[1] Tutorialspoint.com,"Software Testing Tutorial".
[2] ANSI/IEEE 1059 standard.
[3] M. Robin, "Communication and Concurrency", Prentice hall, **(1989)**.
[4] N. Shiratori, H. Kaminaga, K. Takahashi and S. Noguchi, "A Verification Method for LOTOS Specifications and its Application", Protocol Specification, Testing and Verification IX, North-Holland, **(1990)**, pp. 59-70.
[5] B. Park, S. Kimura, E. Lee and N. Shiratori, "Process Error Detection of Education Support Environment for LOTOS", IEICE Trans., vol. J80-D-II, no. 4, **(1997)**, pp. 961-971.
[6] B. Park, D. Kim, Y. Park, H. Son and R. Y. Kim, "A Method for Establishing Simplest LTS based on Strong Equivalence", SMA2013, Malaysia, **(2013)**, pp. 277-280.
[7] G. J. Myers, "The Art of Software Testing, Second Edition", Word Association, Inc., **(2004)**.
[8] ISO: Information Processing Systems Open Systems Interconnection LOTOS A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, ISO 8807, **(1989)**.

Authors

Byungho Park received the M.S. degree and the Ph.D. degree in Computer Science from Tohoku University, Japan in 1995, 1999 respectively. He also worked as an officer in KMND of Korean government. He joined the Korea Military Academy as a professor. He is currently a visiting professor in Hongik University. His research interests are Formal Method, Software Test and Software Security.

Yong B. **Park** received the M.S. degree and the Ph.D. degree in Computer Science from N.Y. Polytechnic University(NYU-Poly), USA in 1987, 1991 respectively. He is currently a professor in Dankook University. His research interests are Appling AI technology to SE, Information Architecture, and Secure Industrial Control Software.

R. YoungChul Kim received the B.S. degree in Computer Science from Hongik University, Korea in 1985, and the Ph.D. degree in Illinois Institute of Technology (IIT), USA in 2000. He also worked as a primary manager in LG industrial Research Center. He is currently a professor in Hongik University. His research interests are test case generation, test maturity model(TMM), and modeling based testing(MBT).

