Method of Symbolic Test Scenarios Automated Concretization

[1]Nikita V. Voinov <voinov@ics2.ecd.spbstu.ru>
[1]Pavel D. Drobintsev <drob@ics2.ecd.spbstu.ru>
[1]Igor V. Nikiforov <igor.nikiforovv@gmail.com>
[1]Vsevolod P. Kotlyarov <vpk@ics2.ecd.spbstu.ru>
[2]Alexander V. Kolchin <shurik@iss.org.ua>
[1]Peter The Great Saint-Petersburg Polytechnic University
Politehnicheskaya str., 29, Saint-Petersburg, Russia
[2]Glushkov Institute of Cybernetics NAS Ukraine
Academika Glushkova avenue, 40, Kiev, Ukraine

Abstract. When providing correctness checking for the models of software systems which include data types with wide range of values, a single symbolic behavioral scenario may cover a set of equivalent scenarios with concrete values. This feature is especially useful for systems with integer data types. However symbolic scenarios cannot be used as executable tests, they shall be concretized prior to execution. On the other hand, modern industrial software projects may contain many thousands of tests with nontrivial dependencies between their parameters. Manual selection and insertion of concrete values is impossible which requires full automation of the process. Besides, the actual experience in modern testing shows that efforts on bugs detection decrease significantly when directed method of selecting values is used instead of random selection of values. Concretization process shall follow a test plan prepared by tester. Such plans shall be flexible and generated based on standard templates or plans modified by user.

Method of symbolic test scenarios automated concretization which resolves mentioned issues is described in the article. It allows to control coverage of boundary test parameters values which increases the quality of developed software.

The developed method was successfully integrated into single complex technology of verification and testing which includes creation of a formal model based on initial requirements, automated symbolic verification, generation and concretization of symbolic behavioral scenarios, generation of test sets based on concretized scenarios and analysis of tests execution verdict. Results of method application within integrated technology are also shown.

Keywords: concretization; symbolic behavior scenario; software testing

DOI: 10.15514/ISPRAS-2015-27(3)-8

For citation: Voinov N.V., Drobintsev P.D., Nikiforov I.V., Kotlyarov V.P., Kolchin A.V. Method of Symbolic Test Scenarios Automated Concretization. *Trudy ISP RAN/Proc. ISP RAS*, vol. 27, issue 3, 2015, pp. 115-124. DOI: 10.15514/ISPRAS-2015-27(3)-8.

1. Introduction

In the scope of software lifecycle the cost of software defects increases dramatically in accordance with development stage [1]. Avoiding defects on the stage of requirements gathering and detecting them on early stages of project lifecycle reduces the amount of corrections in the software and overall cost of development. This makes usage of methods and tools for model-based verification and testing extremely valuable [2,3]. However in the toolsets which mainly resolve problems of model-based approach (automation of requirements formalization, creation of behavioral models, verification of generated model-based scenarios, requirements coverage analysis [4-7]) arises the combinatorial explosion problem of possible behavioral scenarios which shall be tested [8-11].

Methods of symbolic verification are very effective to reduce the behavioral space. It is possible to specify ranges of possible parameters values in symbolic scenario. Each symbolic scenario represents a set of concrete scenarios with equivalent behavior (with same sequence of events). This means that to provide required coverage of complete model behavior it is enough to select several specific scenarios from each group of behavioral equivalence instead of having to check all possible parameters values. This allows to significantly reduce the number of scenarios covering the functionality of application in the scope of selected coverage criteria. However for code generation of executable tests only scenarios with concrete values of parameters are needed. Given that modern industrial software requires many thousands of tests with complex dependencies of parameters values it is impossible to manually count and insert appropriate concrete values based on ranges in symbolic scenario. The concretization process shall be completely automated.

This paper describes the automated concretization process for symbolic test scenarios in the scope of VRS/TAT toolset [12] providing automated generation of test scenarios based on requirements specifications formalized with basic protocol notation [13], which is a representation of Hoare triple [14].

2. Overall Scheme of Concretization

VRS includes symbolic trace generator STG [15] which observes the formal model behavioral space and creates traces – linear sequences of events in the model. Model states are also saved in traces. The mail tool for concretization is called Trace Concretization Tool (Fig. 1). It consists of three modules – Concretizer, ValueCalculator and Concretization View which interact between each other.

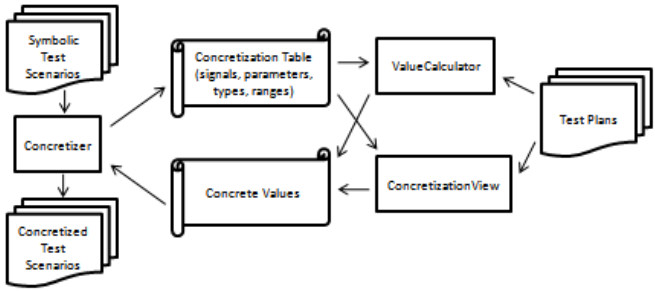

Fig. 1. Scheme of concretization

For each symbolic trace Concretizer generates concretization table with names of parameters, signals, data types and allowed ranges of values. Then while trace bypassing it calls for ValueCalculator to get concrete value for the current parameter. ValueCalculator calculates concrete value based on received commands, test plan and allowed ranges of values and returns it to Concretizer.

The implemented tools Concretizer, ValueCalculator and ConcretizationView are integrated into single concretization process which is a component of industrial software automated testing technology.

3. Steps of Concretization Algorithm

Concretization process is iterative, on each step a single parameter is concretized. The process terminates after concretization of the last parameter in the trace.

Below some definitions are introduced. Transition in the formal model in VRS terms is a basic protocol representing parameterized transition from one model's state into another. Basic protocol *B(x)* is represented by the following expression:

$$\forall x(\alpha(x) \rightarrow < P(x) > \beta(x))$$

where x is a list of protocol's parameters; $\alpha(x), \beta(x)$ – a formula of basic logic language, which are called precondition and postcondition respectively; *P(x)* – a process of basic protocol (in current case – a sequence of parameterized signals in MSC format). Trace parameters are parameters of its signals. Formula of basic language may contain variables and constants, arrays of elements of simple types, functional types. Variables which may change their values during system execution are represented by attributes and attribute expressions.

Trace is a sequence of the following type:

$$S_0 \xrightarrow{B_0(x_0)} S_1 \xrightarrow{B_1(x_1)} \ldots S_n$$

where S are model's states, B – basic protocols, x – lists of their parameters.

The following steps of concretization algorithm can be specified:

- restore of initial symbolic trace

- obtain ranges of allowed values for basic protocol's parameters
- interactive concretization of trace parameters
- save concretized trace.

All steps except interactive concretization are executed automatically by internal means of VRS and hidden from outside. The most interesting for the user are implemented tools of the concretization which provide the control of concretization process and make the technology flexible enough for testing all modes of software functionality.

4. ValueCalculator Tool

This tool implements automatic calculation of concrete values for symbolic parameters within test scenarios. One or several rules can be used for calculation: left value of the range, middle value or right value. Examples of values calculated based on ranges and selected rule are shown in the table below:

Type	Range	Rule	Calculated Value
integer	[1;9]	L	1
integer	[1;9]	M	5
integer	[1;9]	R	9
enumerated	val1,val2,val3,val4	L	Val1
enumerated	val1,val2,val3,val4	M	Val2
enumerated	val1,val2,val3,val4	R	Val4

Possible values for each parameter on each step of behavioral trace are calculated automatically by the means of VRS. Selection of the rule for value calculation is provided by corresponding set of options (Fig. 2):

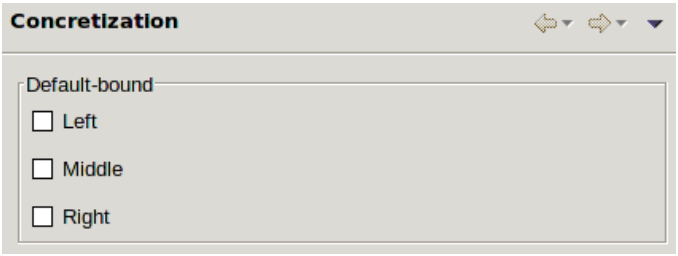

Fig. 2. Options for selecting concretization rule

Based on calculated values of symbolic parameters the STG creates traces with concrete values which can be executed on the model. When two or three rules are selected there will be two or three concretized traces generated for each symbolic scenario.

An example of tool execution is shown below. Test scenario contains a signal which turns on a radio station on the car radio. Radio station number is the signal's parameter (Fig. 3):

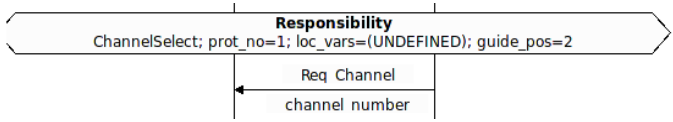

Fig. 3. A part of symbolic test scenario

If overall number of radio stations is 9, ValueCalculator will calculate the following values for the channel_number parameter depending on selected concretization rule: "1" (for the Left rule), "5" (for the Middle rule) and "9" (Right rule). If all three options are selected (Fig. 2), there will be three concretized traces generated with different values of channel_select parameter. A part of concretized trace with Right rule value selection is shown below (Fig. 4):

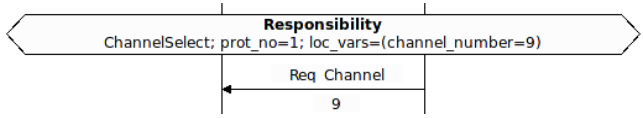

Fig. 4. A part of concretized trace with right value selected

The user can select default concretization rules and repeat generation of concretized traces with corresponding values or use ConcretizationView tool to create own test plan.

5. ConcretizationView Tool

This tool provides the ability to specify any concrete values from the possible range for one, several or all parameters in test scenario. The tool is implemented as a View element in Eclipse IDE. It allows to display the contents of concretization table and specify desired values of symbolic parameters. This is performed by adding "C" symbol on the row with required parameter in the "Rule" column and desired value in the "Value" column.

Continue with the example of turning on a radio station of the car radio. If the range of parameter's possible values varies between 1 and 9, then for example value 7 is neither left, nor middle, nor right value of the range. The only possible way to concretize a trace with this value is to explicitly specify it using ConcretizationView tool (Fig. 5):

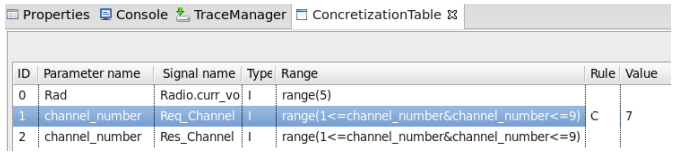

Fig. 5. ConcretizationView user interface

As a result the concretized trace with value 7 will be generated (Fig. 6):

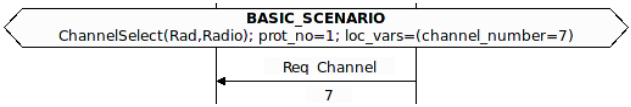

Fig. 6. A part of concretized trace with user-defined value

Applying ValueCalculator and ConcretizationView tools together the user can obtain all tests required to satisfy specific test criteria. For example, a set of tests covering all possible values of one parameter and only boundary values of another parameter. The concretization process terminates when the complete set of tests required for execution is obtained.

6. Results

Created tools were applied for preparing tests in telecom software projects. Symbolic scenarios of possible systems behaviors contained up to several hundred of basic protocols. For testing process all symbolic parameters in generated scenarios shall be concretized which is extremely time consuming without tools of automation. For example, using described approach to concretization in a small project with 11 basic protocols allowed to concretize all traces in 2 minutes. For a project with 151 basic protocols the concretization took about 20 minutes. While manual concretization of such project takes about 3 working days. Clear that in projects with several thousand of basic protocols it is impossible to concretize symbolic scenarios without automation toolset. The table below shows the comparison between manual and automated approaches to concretization:

Number of Basic Protocols in the project	Manual Concretization (staff days)	Automated Concretization (minutes)
11	0,3	2
151	3	20
464	5	25
759	8	28

7. Conclusion

Integration of verification and testing allows to achieve desired level of software quality due to joining results of model static analysis after symbolic verification with number of experimental results after testing which is especially important for testing systems with wide ranges of possible values.

It is also important that symbolic scenarios can not be used for execution on the model. They shall be concretized prior to generating test code for target platform.

Implemented tools which are integrated into single chain of concretization in the scope of test automation technology [16], successfully resolve a very time-consuming problem of symbolic scenarios concretization. Also the technology allows to control coverage of boundary test parameters values which increases the quality of developed software.

References

[1]. Boehm B., Software Engineering Economics, Prentice Hall,Inc.Englewood Cliffs,New Jersey, N.Y. 1981. – 767 p.
[2]. Utting, M. and Legeard, B., Practical Model_Based Testing: A Tools Approach, Morgan_Kaufmann, 2010.
[3]. Burdonov, I., Kosachev, A., Ponomarenko, V., and Shnitman, V., Review of Approaches to Verification of Distributed Systems, M.: ISP RAS, 2006.
[4]. TestOptimal // www.testoptimal.com
[5]. Qtronic // www.conformiq.com
[6]. Test Designer // www.smartesting.com
[7]. Spec Explorer: Microsoft Research // http://research.microsoft.com/specexplorer
[8]. Primeneniye metoda evristik dlya sozdaniya optimalnogo nabora testovykh stsenariyev / N. V. Voinov, V. P. Kotlyarov // Nauchno-tekhnicheskiye vedomosti Sankt-Peterburgskogo gosudarstvennogo politekhnicheskogo universiteta. Informatika. Telekommunikatsii. Upravleniye. – 2010. – T.4 – № 103. – S. 169–174.
[9]. Grindal M. Handling Combinatorial Explosion in Software Testing. Department of Computer and Information Science, Linköpings universitet, 2007.
[10]. C. Nie and H. Leung, “A survey of combinatorial testing,” ACM Comput.Surv., vol. 43, no. 2, pp. 11:1–11:29, Feb. 2011.
[11]. J. McGregor, “Testing a software product line,” in Testing Techniques in Software Engineering. Springer, 2010, vol. 6153, pp. 104–140.
[12]. Baranov S.N., Drobintsev P.D., Kotlyarov V.P., Letichevsky A.A. Implementation of an integrated verification and testing technology in telecommunication project. Proceedings // IEEE Russia Northwest Section. 110 Anniversary of Radio Invention conference. S.Petersburg, 2005. 11 p.
[13]. Letichevsky J., Kapitonova A., Letichevsky Jr., Volkov V., Baranov S., Kotlyarov V., Weigert T. Basic Protocols, Message Sequence Charts, and the Verification of Requirements Specifications // Computer Networks. 2005. 47. P. 662–675.
[14]. Hoare, C.A.R., Communicating Sequential Processes, Prentice Hall, 1985.
[15]. Letichevsky Jr., A. and Kolchin, A., Test scenarios generation based on formal model, Programming Problems, 2010, nos. 2–3, pp. 209–215.

[16]. Drobintsev P. D., Kotlyarov V. P., Nikiforov I. V., Letichevsky A. A., Incremental approach to the technology of test design for industrial projects, Modeling and Analysis of Information Systems, 2014, Volume 21, Number 6, 144–154.

Метод автоматической конкретизации символических тестовых сценариев

[1]Никита Воинов (voinov@ics2.ecd.spbstu.ru),
[1]Павел Дробинцев (drob@ics2.ecd.spbstu.ru),
[1]Игорь Никифоров (igor.nikiforovv@gmail.com),
[1]Всеволод Котляров (vpk@ics2.ecd.spbstu.ru)
[2]Александр Колчин (shurik@iss.org.ua)
[1]Санкт-Петербургский Политехнический Университет Петра Великого,
195251, Россия, г.Санкт-Петербург, ул.Политехническая, 29
[2]Институт кибернетики им.В.М.Глушкова НАН Украины,
03680, Украина, г.Киев, пр.Академика Глушкова, 40

Аннотация. При проверке корректности моделей программных систем с типами данных, включающими большой диапазон значений, один символьный поведенческий сценарий может покрывать множество сценариев с конкретными значениями. Это свойство особенно эффективно используется для систем, использующих числовые типы данных. Однако символьные сценарии в исходном виде непригодны для создания по ним исполнимых тестов, для исполнения тесты должны быть конкретизированы. С другой стороны, в современных промышленных проектах количество тестов исчисляется тысячами, а зависимость между значениями параметров нетривиальна. Осуществлять вручную выбор и подстановку взаимосогласованных конкретных значений практически невозможно, поэтому процесс конкретизации должен быть полностью автоматическим. Кроме того, реальная практика тестирования свидетельствует об уменьшении трудоемкости поиска дефектов при направленном выборе значений по сравнению со случайном выбором. При подстановках необходимо следовать плану тестирования, подготовленному заранее тестировщиком. Подобные планы должны быть гибкими, основная их часть должна генерироваться на основе стандартных шаблонов или переиспользовать отредактированные пользователем планы.

В работе описан полностью автоматизированный метод конкретизации символьных сценариев, решающий упомянутые проблемы. Метод позволяет контролировать покрытие граничных значений параметров теста, что в результате дает возможность повысить качество создаваемого тестового набора.

Разработанный метод был успешно интегрирован в единую технологическую цепочку верификации и тестирования, включающую создание формальной модели системы по исходным требованиям, автоматическую символьную верификацию, генерацию и конкретизацию символьных поведенческих сценариев, генерацию тестовых наборов по конкретизированным сценариям, а также средства анализа результатов исполнения

тестов. Также продемонстрированы результаты применения технологической цепочки с интегрированным методом автоматической конкретизации.

Keywords: concretization; symbolic behavior scenario; software testing

DOI: 10.15514/ISPRAS-2015-27(3)-8

Для цитирования: Воинов Н.В., Дробинцев П.Д., Никифоров И.В., Котляров В.П., Колчин А.В. Метод автоматической конкретизации символических тестовых сценариев. Труды ИСП РАН, том 27, вып. 3, 2015 г., стр. 115-124 (на английском языке). DOI: 10.15514/ISPRAS-2015-27(3)-8.

Список литературы

[1]. Boehm B., Software Engineering Economics, Prentice Hall,Inc.Englewood Cliffs,New Jersey, N.Y. 1981. – 767 p.
[2]. Utting, M. and Legeard, B., Practical Model_Based Testing: A Tools Approach, Morgan_Kaufmann, 2010.
[3]. И.Б.Бурдонов, А.С.Косачев, В.Н.Пономаренко, В.Шнитман. “Обзор подходов к верификации распределенных систем” // ИСП РАН, препринт 16, М., 2006.
[4]. TestOptimal // www.testoptimal.com
[5]. Qtronic // www.conformiq.com
[6]. Test Designer // www.smartesting.com
[7]. Spec Explorer: Microsoft Research // http://research.microsoft.com/specexplorer
[8]. Н.В.Воинов, В.П.Котляров. “Применение метода эвристик для создания оптимального набора тестовых сценариев” // Научно-технические ведомости СПбГПУ. Информатика. Телекоммуникации. Управление. – 2010. – Т.4 – № 103. – с. 169–174.
[9]. Grindal M. Handling Combinatorial Explosion in Software Testing. Department of Computer and Information Science, Linköpings universitet, 2007.
[10]. C. Nie and H. Leung, “A survey of combinatorial testing,” ACM Comput.Surv., vol. 43, no. 2, pp. 11:1–11:29, Feb. 2011.
[11]. J. McGregor, “Testing a software product line,” in Testing Techniques in Software Engineering. Springer, 2010, vol. 6153, pp. 104–140.
[12]. Baranov S.N., Drobintsev P.D., Kotlyarov V.P., Letichevsky A.A. Implementation of an integrated verification and testing technology in telecommunication project. Proceedings // IEEE Russia Northwest Section. 110 Anniversary of Radio Invention conference. S.Petersburg, 2005. 11 p.
[13]. Letichevsky J., Kapitonova A., Letichevsky Jr., Volkov V., Baranov S., Kotlyarov V., Weigert T. Basic Protocols, Message Sequence Charts, and the Verification of Requirements Specifications // Computer Networks. 2005. 47. P. 662–675.
[14]. Hoare, C.A.R., Communicating Sequential Processes, Prentice Hall, 1985.
[15]. Letichevsky Jr., A. and Kolchin, A., Test scenarios generation based on formal model, Programming Problems, 2010, nos. 2–3, pp. 209–215.
[16]. Drobintsev P. D., Kotlyarov V. P., Nikiforov I. V., Letichevsky A. A., Incremental approach to the technology of test design for industrial projects, Modeling and Analysis of Information Systems, 2014, Volume 21, Number 6, 144–154.