УДК 004.4’242

ПРИМЕНЕНИЕ ШАБЛОНОВ ТРЕБОВАНИЙ К ФОРМАЛЬНОЙ СПЕЦИФИКАЦИИ И ВЕРИФИКАЦИИ АВТОМАТНЫХ ПРОГРАММ

А.А. Клебанов, О.Г. Степанов, А.А. Шалыто

Верификация на модели *(model checking*) является глубоко проработанной технологией проверки корректности программного обеспечения, однако при этом она недостаточно широко используется на практике. Одной из причин является сложность составления формальных требований на языке темпоральных логик. В настоящей работе описывается подход для записи верифицируемых требований на подмножестве естественного языка в контексте автоматного программирования. В частности, приводится анализ применимости шаблонов требований к формальной спецификации автоматных программ, а также описывается грамматика для вывода требований.

Ключевые слова: автоматное программирование, верификация на модели, темпоральные логики.

Введение

Автоматное программирование [1] – это метод разработки программного обеспечения (ПО), основанный на расширенной модели конечного автомата. В рамках данного подхода программы представляются системой автоматизированных объектов управления, логика поведения которых задается системой взаимодействующих управляющих автоматов.

В ряде работ [2, 3] показано, что к автоматным программам хорошо применима верификация на модели (*Model Checking*) [4]. Суть такой верификации состоит в проверке соответствия модели с конечным числом состояний (*структуры Крипке*) формальной спецификации, заданной в виде набора формул темпоральной логики. При верификации преимуществом автоматного подхода перед традиционными подходами к разработке ПО является высокая степень автоматизации, так как в автоматных программах модель поведения задается априори. Разработаны методы [5–7], позволяющие автоматически преобразовывать как управляющие автоматы в модель, пригодную для верификации, так и построенный верификатором контрпример в автоматную модель. Однако, как при верификации автоматных программ, так и при верификации программ общего вида, существует следующая проблема – необходимость записи формальных требований в виде формул темпоральных логик, работа с которыми достаточно трудоемка и требует значительной математический подготовки.

В работе [8] эта проблема частично решается использованием контрактов [9]. Хотя контракты являются более простым формализмом (и, как следствие, ошибки в подобной формальной спецификации менее вероятны), рассматриваемый подход имеет несколько недостатков. Во-первых, контракты значительно уступают темпоральным логикам в выразительных возможностях. К записи требований они применимы только в том случае, когда необходимо специфицировать свойства инвариантности, предусловия или постусловия. Во-вторых, специфицирование требований, распространяющихся на группу состояний, может быть достаточно трудоемким процессом, поскольку для каждого состояния из группы потребуется разработать свою спецификацию. Таким образом, описанную выше проблему нельзя считать полностью решенной.

В настоящей работе описывается подход к записи требований, скрывающий сложность темпоральных логик. Предлагается записывать требования на подмножестве естественного языка, заданного приводимой ниже формальной грамматикой. Грамматика основывается на наборе шаблонов требований [10, 11] – обобщенном описании (формальном и на естественном языке) часто встречающихся ограничений на допустимые последовательности состояний в модели системы с конечным числом состояний. Таким образом, для каждого полученного требования существует эквивалентная формальная запись, позволяющая осуществить верификацию.

Актуальность применения шаблонов требований в контексте автоматного программирования отмечается в работе [2]: «… важным является вопрос о шаблонах (структуре) темпоральных свойств, наиболее применимых и адекватных для верификации автоматных программ. Наличие таких шаблонов позволяло бы говорить о классах темпоральных свойств автоматных моделей, что, несомненно, облегчало бы построение технологической схемы проверки автоматных программ на корректность относительно спецификации». Однако в указанной работе выделяется только одно требование, являющееся частным случаем существующего шаблона, и дальше этот вопрос никак не прорабатывается.

В начале настоящей работы кратко описываются шаблоны требований, затем приводится анализ их применимости к спецификации автоматных программ, и, наконец, вводится формальная грамматика для записи требований. В заключении сделаны выводы по работе.

Шаблоны требований

В работах [10, 11] предлагается система шаблонов требований, разработанная на основе спецификаций для программ общего вида. Шаблоны можно классифицировать в соответствии с иерархической структурой, основанной на их семантике. В настоящее время выделено восемь основных шаблонов («Отсутствие», «Существование», «Всеобщность», «Ограниченное существование», «Предшествование»,

«Ответ», «Цепное предшествование», «Цепной ответ»), которые делятся на две группы – «Наличие» и «Порядок». В группу «Наличие» входят шаблоны, описывающие наличие или отсутствие состояний, в которых выполняется заданное требование. В группу «Порядок» – описывающие порядок состояний.

Описание шаблона состоит из его имени (или списка имен), цели, записи на различных формализмах (LTL, CTL и т.п.), примера использования и связи с другими шаблонами.

Каждое требование имеет *ограничение* – ту часть пути исполнения, на котором это требование должно выполняться. Всего выделены пять видов ограничений.

1. Глобально – на всем пути исполнения.
2. До – на пути до заданного состояния.
3. После – на пути после заданного состояния.
4. Между – на пути между двумя заданными состояниями.
5. После-до – аналогично ограничению «Между», однако наличие правой границы интервала не является обязательным.

Отметим, что стандартно для формализмов, ориентированных на состояние, интервал, на котором должно выполняться требование, замкнут на левом конце и открыт на правом.

В табл. 1 приведен пример шаблона «Всеобщность». Оригинальный пример использования, предложенный в работе [10], заменен примером более применимым в контексте автоматного программирования. В этом состоит задача адаптации системы шаблонов для автоматного программирования.

<table>
<tr><td colspan="2">Цель</td><td colspan="2">Используется для описания части пути исполнения системы, в которой содержатся только те состояния, в которых выполняется необходимое требование. Известен также как «Впредь» и «Всегда».</td></tr>
<tr><td rowspan="11">Запись</td><td rowspan="6">LTL</td><td>Ограничение</td><td>Запись</td></tr>
<tr><td>Глобально</td><td>□(P)</td></tr>
<tr><td>До R</td><td>◊R → (P U R)</td></tr>
<tr><td>После Q</td><td>□(Q → □(P))</td></tr>
<tr><td>Между Q и R</td><td>□((Q & !R & ◊R) → (P U R))</td></tr>
<tr><td>После Q до R</td><td>□(Q & !R → (P W R))</td></tr>
<tr><td rowspan="5">CTL</td><td>Ограничение</td><td>Запись</td></tr>
<tr><td>Глобально</td><td>AG(P)</td></tr>
<tr><td>До R</td><td>A[(P | AG(!R)) W R]</td></tr>
<tr><td>После Q</td><td>AG(Q → AG(P))</td></tr>
<tr><td>Между Q и R</td><td>AG(Q & !R → A[(P | AG(!R)) W R])</td></tr>
<tr><td>После Q до R</td><td>AG(Q & !R → A[P W R])</td></tr>
<tr><td colspan="2">Пример использования</td><td colspan="2">Этот шаблон может быть применен для описания общих свойств модели в целом или отдельной группы состояний. Например, в том случае, когда необходимо выразить свойство вида: «Если автомат находится в состоянии s, то верно свойство P».
При использовании ограничения «Глобально» подстановка темпоральных выражений <>f или AF(f) в качестве параметра P позволяет выразить свойство справедливости.</td></tr>
<tr><td colspan="2">Связь с другими шаблонами</td><td colspan="2">Этот шаблон тесно связан с шаблонами «Отсутствие» и «Существование». Наличие состояния, в котором выполняется требование, может рассматриваться как отрицание его отсутствия.</td></tr>
</table>

Таблица 1. Шаблон «Всеобщность»

Применимость шаблонов требований к спецификации автоматных программ

Рассмотрим вопрос применимости шаблонов требований к формальной спецификации автоматных программ. Для этого проанализируем требования к различным программам, разработанным в СПбГУ ИТМО, Ярославском государственном университете, ОАО «Концерн «НПО «АВРОРА» и доступным на сайте [12], и проверим, как они выражаются при помощи шаблонов. Пример организации промежуточных результатов анализа приводится в табл. 2. В столбцах «Требование» и «Исходная формальная запись» приводятся оригинальные требования из источника (столбец «Источник»), записанные на естественном языке и одном из формализмов соответственно. В столбце «Шаблон, Ограничение» приводится запись шаблона, подстановка необходимых утверждений в который, даст эквивалентную исходной формальную запись. В случае необходимости приводится формальное доказательство эквивалентности.

Требование	Исходная формальная запись	Шаблон, Ограничение	Источник
Если произошла поломка нагревателя или одного из клапанов, то кофеварка (основной автомат A_0) обязательно перейдет в состояние 5.	$AG((y_{31} = 4 \mid y_{32} = 4 \mid y_2 = 4) \& y_0 = 2 \rightarrow A(y_0 = 2\ U\ y_0 = 5))$	Ответ (ограниченный), Глобально $AG(P \rightarrow A(S))$, $P: (y_{31} = 4 \mid y_{32} = 4 \mid y_2 = 4) \& y_0 = 2$, $S: y_0 = 2\ U\ y_0 = 5$	2

Таблица 2. Анализ применимости шаблонов требований к спецификации автоматных программ

Всего было рассмотрено 118 требований и их вариантов из 20 источников. Установлено, что 85% требований покрывается пятью шаблонами. Оставшиеся 15% не удается записать при помощи шаблонов из-за ряда причин, таких как ограниченность системы шаблонов и некоторых особенностей конкретной автоматной модели рассматриваемого приложения. Процентное соотношение между использованными шаблонами приведено на рис. 1, а между использованными ограничениями – на рис. 2.

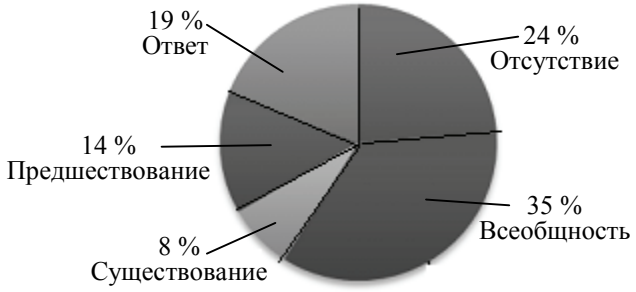

Рис. 1. Процентное соотношение между использованными шаблонами

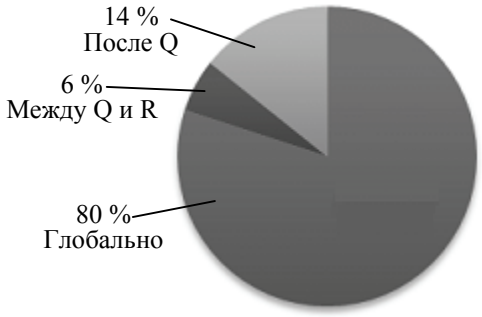

Рис. 2. Процентное соотношение между использованными ограничениями

Метод записи верифицируемых требований на естественном языке

Существуют различные подходы к выделению формальных (верифицируемых) требований из спецификаций, записанных на естественном языке. Отметим два основных направления [13] – синтаксический разбор текстов и использование формальных грамматик, ограничивающих естественный язык.

В настоящей работе предлагается использовать формальную грамматику, которая приводится в табл. 3. Как отмечалось выше, грамматика основывается на основных шаблонах требований и их вариантах. Это позволяет иметь запись требования, как на естественном языке, так и на любом из формализмов, запись на которых разработана для шаблонов. Моноширинным шрифтом выделены указатели места заполнения шаблона реальными требованиями.

<требование>	::= <ограничение> <шаблон>
<ограничение>	::= «Для любого состояния верно, что» | «До состояния, в котором Q, верно что» | «После состояния, в котором Q, верно что» | «Между состоянием, в котором Q, до состояния, в котором R, верно что» | «После состояния, в котором Q, до состояния, в котором R, верно что»
<шаблон>	::= <отсутствие> | <всеобщность> | <существование> | <предшествование> | <ответ> | <ответ на следующем шаге>
<отсутствие>	::= «никогда не выполняется P»
<всеобщность>	::= «всегда выполняется P»
<существование>	::= «когда-нибудь выполнится P»
<предшествование>	::= «всегда верно, что если выполнено P, то до этого было выполнено S»
<ответ>	::= «всегда верно, что если выполнится P, то когда-нибудь выполнится S»
<ответ на следующем шаге>	::= «всегда верно, что если выполнится P, то в следующем состоянии выполнится S»

Таблица 3. Грамматика верифицируемых требований на подмножестве русского языка

В качестве примера рассмотрим реальное требование к системе управления кофеваркой, которое приводится в работе [3]: «Система управления кофеваркой никогда не попадет в такое состояние, в котором она не реагирует ни на события системного таймера, ни на нажатие кнопок «ОК» и «С». В автомат-

ной модели кофеварки требованию «Никак не реагирует ни на события системного таймера, ни на нажатие кнопок «ОК» и «С» соответствует предикат act = end. Наречие «никогда» подсказывает, что должен быть использован шаблон «Отсутствие» с ограничением «Глобально».

Выполним порождение:

<требование> → <ограничение> <шаблон> → Для любого состояния верно, что <шаблон> → Для любого состояния верно, что <отсутствие> → Для любого состояния верно, что никогда не выполняется P

Подставив вместо P реальное требование, получим искомое формальное требования на естественном языке: «Для любого состояния верно, что никогда не выполняется act = end». Этому требованию на языках CTL и LTL соответствуют выражения AG(! act = end) и □(! act = end). Эти выражения совместно с автоматной моделью системы подаются на вход инструментальному средству, осуществляющему проверку на модели.

Заключение

Запись требований в виде формул темпоральной логики является неотъемлемой частью верификации на модели. В настоящей работе в рамках автоматного программирования предложен подход, который, во-первых, значительно упрощает этот процесс, а во-вторых, минимизирует число потенциальных ошибок в самой спецификации.

В качестве приоритетного направления дальнейших исследований можно выделить инструментальную поддержку предложенного подхода. В работе [8] показывается, как система метапрограммирования *JetBrains MPS* [14] может быть использована как для разработки, так и для верификации автоматных программ. На текущий момент требования записываются в виде формул темпоральных логик, что может быть улучшено на основе описанного выше решения. Аналогично работам [13, 15, 16] может быть реализован помощник для интерактивного выбора необходимого ограничения и шаблона. Наконец, остается открытым ряд теоретических вопросов, таких как расширение набора шаблонов или запись существующих шаблонов на новом формализме.

Исследование проводится в рамках Федеральной целевой программы «Научные и научно-педагогические кадры инновационной России на 2009–2013 годы».

Литература

1. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. – СПб: Питер, 2009. – 176 с. [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/books/_book.pdf, своб.
2. Васильева К.А., Кузьмин Е.В. Верификация автоматных программ с использованием LTL // Моделирование и анализ информационных систем. – 2007. – Т. 14. – № 1. – С. 3–14.
3. Кузьмин Е.В., Соколов В.А. Моделирование, спецификация и верификация «автоматных» программ // Программирование. – 2008. – № 1. – С. 38–60.
4. Кларк Э., Грамберг О., Пелед Д. Верификация моделей программ. Model Checking. – М.: Изд-во МЦНМО, 2002. – 416 с.
5. Гуров В.С., Яминов Б.Р. Технология верификации автоматных моделей программ без их трансляции во входной язык верификатора // Тезисы научно-технической конференции «Научное программное обеспечение в образовании и научных исследованиях». – СПбГУ ПУ, 2008. – С. 36–40 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/download/2008-02-24_jaminov_verifikazija.pdf, своб.
6. Лукин М.А., Шалыто А.А. Автоматизация верификации визуальных автоматных программ / Материалы XV Международной научно-методической конференции «Высокие интеллектуальные технологии и инновации в образовании и науке». – СПбГПУ, 2008. – С. 296–297 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/download/2008-02-25_politech_tezis.pdf, своб.
7. Kurbatsky E. Verification of Automata-Based Programs // Proceedings of the Second Spring Young Researchers Colloquium on Software Engineering. – 2008. – V. 2. – P. 15–17 [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/verification/_kurbatsky_syrcse.pdf, своб.
8. Степанов О.Г. Методы реализации автоматных объектно-ориентированных программ. Диссертация на соискание ученой степени кандидата технических наук. – СПбГУ ИТМО, 2009. – Режим доступа: http://is.ifmo.ru/disser/stepanov_disser.pdf, своб.
9. Мейер Б. Объектно-ориентированное конструирование программных систем. – М.: Русская редакция, 2005. – 1204 с.
10. Dwyer M.B., Avrunin G.S., Corbett J.C. Property Specification Patterns for Finite-state Verification // Proceedings of the 2nd Workshop on Formal Methods in Software Practice. – 1998.
11. Dwyer M.B., Avrunin G.S., Corbett J.C. Patterns in Property Specifications for Finite-state Verification // Proceedings of the 21st International Conference on Software Engineering. – 1999.

12. Веб-сайт кафедры «Технологии программирования» [Электронный ресурс]. – Режим доступа: http://is.ifmo.ru/, своб.
13. Konrad S., Cheng B.H.C. Facilitating the Construction of Specification Pattern-based Properties // Proceedings of the IEEE International Requirements Engineering Conference. – 2005.
14. JetBrains Meta Programming System [Электронный ресурс]. – Режим доступа: http://www.jetbrains.com/mps/index.html, своб.
15. Smith R.L., Avrunin G.S., Clarke L.A., Osterweil L.J. PROPEL: An Approach Supporting Property Elucidation // Proceedings of the 24th Int'l. Conference on Software Engineering. – 2002.
16. Mondragon O., Gates A.Q., Roach S. Prospec: Support for Elicitation and Formal Specification of Software Properties // Proceedings of Run-time Verification Workshop. – 2004.

Клебанов Андрей Александрович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, магистрант, klebanov.andrey@gmail.com

Степанов Олег Георгиевич – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, кандидат технических наук, ассистент, oleg.stepanov@gmail.com

Шалыто Анатолий Абрамович – Санкт-Петербургский государственный университет информационных технологий, механики и оптики, доктор технических наук, профессор, заведующий кафедрой, shalyto@mail.ifmo.ru